Chapter 12

Applications

In this chapter we provide some applications to illustrate the power of mathematical programming combined with a tool, such as GAMS, that allows one to solve, in an efficient and easy way, the stated problems. We start in Section 12.1, giving an application to neural and functional networks. In Section 12.2 we present an automatic mesh generation method that can be useful in finite element methods and in all methods that require describing surfaces by means of triangular elements. In Section 12.3 we give some applications to probability. In Section 12.4 we show how to solve some regression models, that are difficult to solve using standard regression techniques. In Section 12.5 we present some applications to continuous optimization problems, as the braquistochrone, the hanging cable, the problems of optimal control of hitting a target and of an harmonic oscillator, and the transportation and the unit commitment problems. In Section 12.6 we provide applications to transportation systems. Finally, in Section 12.7 an application to power systems is provided.

12.1 Applications to Artificial Intelligence

In this section we show how mathematical programming can be applied to deal with neural and functional network problems. This application is based on Castillo et al. [20].

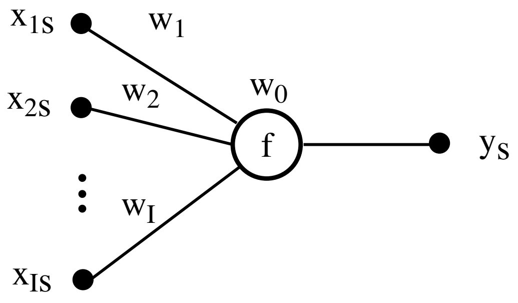

Figure 12.1: A one-layer neural network with its neural function.

Neural networks consist of one or several layers of neurons connected by links. Each neuron computes a scalar output from a linear combination of inputs, coming from the previous layer, using a given scalar function.

Consider the one-layer neural network in Figure 12.1, where the neural function f is assumed to be invertible.

The typical problem of neural networks consists of learning the thresholds and weights given a set of data

$$D \equiv \{(x_{1s}, x_{2s}, \ldots, x_{Is}, y_s) | s = 1, 2, \ldots, S\}$$

which contains the set of inputs $(x_{1s}, x_{2s}, \ldots, x_{Is})$ and the corresponding outputs y_s for a collection of S measurements.

The set of equations giving the outputs as a function of the inputs is

$$y_s = f(w_0 + \sum_{i=1}^{I} w_i x_{is}) + \delta_s; \quad s = 1, 2, \ldots, S \tag{12.1}$$

where w_0 and $w_i; i = 1, 2, \ldots, I$ are the neuron threshold values and weights, respectively, and δ_s measures the error associated with the output y_s.

Three possibilities for learning the thresholds and weights are given below:

Option 1. Minimize

$$Q_1 = \sum_{s=1}^{S} \delta_s^2 = \sum_{s=1}^{S} \left(y_s - f(w_0 + \sum_{i=1}^{I} w_i x_{is}) \right)^2 \tag{12.2}$$

which is the sum of squared errors measured in the output scale (the units of the y_s). Unfortunately, this leads to a nonlinear problem, due to the presence of the usually nonlinear neural function f. Thus, Q_1 is not guaranteed to have a global optima. In fact, it usually has many local optima.

Option 2. Minimize the sum of squared errors:

$$Q_2 = \sum_{s=1}^{S} \epsilon_s^2 = \sum_{s=1}^{S} \left(w_0 + \sum_{i=1}^{I} w_i x_{is} - f^{-1}(y_s) \right)^2 , \tag{12.3}$$

This option measures the errors in terms of the input scale (units of the x_{is}). The advantage is that we have a standard least-squares minimization, that can be written as a system of linear equations (linear problem).

Option 3. Alternatively, we can minimize the maximum absolute error:

$$\min_{w} \left\{ \max_{s} \left| w_0 + \sum_{i=1}^{I} w_i x_{is} - f^{-1}(y_s) \right| \right\} \tag{12.4}$$

which can be stated as the following linear programming problem. Minimize ϵ subject to

$$\left. \begin{array}{rcl} w_0 + \sum\limits_{i=1}^{I} w_i x_{is} - \epsilon & \leq & f^{-1}(y_s) \\ -w_0 - \sum\limits_{i=1}^{I} w_i x_{is} - \epsilon & \leq & -f^{-1}(y_s) \end{array} \right\}; \quad s = 1, 2, \ldots, S \tag{12.5}$$

which global optimum can be easily obtained using linear programming techniques.

12.1.1 Learning the Neural Functions

An important improvement is obtained if we learn the f^{-1} function instead of assuming that it is known. When the neural functions are learned instead of being given, we say that we are "in front" of a functional network. More precisely, f^{-1} can be assumed to be a convex combination of a set $\{\phi_1(x), \phi_2(x), , \ldots, \phi_R(x)\}$ of invertible basic functions:

$$f^{-1}(x) = \sum_{r=1}^{R} \alpha_r \phi_r(x) \tag{12.6}$$

where $\{\alpha_r;\ r = 1, 2, \ldots, R\}$ is the corresponding set of coefficients, which has to be chosen for the resulting function f^{-1} to be invertible. Without loss of generality, it can be assumed to be increasing.

Thus, we can consider two more options:

Option 4. Minimize, with respect to $w_i; i = 0, 1, \ldots, I$ and $\alpha_r; r = 1, 2, \ldots, R$, the function

$$Q_4 = \sum_{s=1}^{S} \epsilon_s^2 \quad = \quad \sum_{s=1}^{S} \left(w_0 + \sum_{i=1}^{I} w_i x_{is} - \sum_{r=1}^{R} \alpha_r \phi_r(y_s) \right)^2 \tag{12.7}$$

subject to

$$\sum_{r=1}^{R} \alpha_r \phi_r(y_{s_1}) \quad \leq \quad \sum_{r=1}^{R} \alpha_r \phi_r(y_{s_2}); \ \ \forall y_{s_1} < y_{s_2} \tag{12.8}$$

which forces the candidate function f^{-1} to increase in the corresponding interval, and

$$\sum_{r=1}^{R} \alpha_r \phi_r(y_0) \quad = \quad 1 \tag{12.9}$$

which avoids the zero function as the optimum.

Option 5. Alternatively, we can minimize the maximum absolute error:

$$Q_5 = \min_{w,\alpha} \left\{ \max_s \left| w_0 + \sum_{i=1}^{I} w_i x_{is} - \sum_{r=1}^{R} \alpha_r \phi_r(y_s) \right| \right\} \tag{12.10}$$

which can be stated as the following linear programming problem. Minimize ϵ subject to

$$\begin{array}{rcl} w_0 + \sum\limits_{i=1}^{I} w_i x_{is} - \sum\limits_{r=1}^{R} \alpha_r \phi_r(y_s) - \epsilon & \leq & 0 \\ -w_0 - \sum\limits_{i=1}^{I} w_i x_{is} + \sum\limits_{r=1}^{R} \alpha_r \phi_r(y_s) - \epsilon & \leq & 0 \\ \sum\limits_{r=1}^{R} \alpha_r \phi_r(y_{s_1}) & \leq & \sum\limits_{r=1}^{R} \alpha_r \phi_r(y_{s_2}); \quad \forall y_{s_1} < y_{s_2} \\ \sum\limits_{r=1}^{R} \alpha_r \phi_r(y_0) & = & 1 \end{array} \tag{12.11}$$

which has a global optimum easily obtainable.

To use the network, that is, to calculate the outputs for given inputs, it is necessary to know $f(x)$. However, this method only gives the function f^{-1}. One possible and efficient way to obtain $f(x)$ when $f^{-1}()$ is given is the bisection method.

The four main elements of the neural and functional networks problem are

1. **Data.**

   S: number of data vectors

   I: dimension of data input vectors

   x_{is}: vectors of input data $(x_{1s}, x_{2s}, \ldots, x_{Is}); s = 1, 2, \ldots, S$

   y_s: output data $y_s; s = 1, 2, \ldots, S$

2. **Variables.**

   w_0: neuron threshold value

   w_i: neuron weight for the component i of the input vector

   ϵ: maximum error (it is a nonnegative variable that has sense only for option 3)

3. **Constraints.** For options 1 and 2 there are no constraints. For Option 3 the constraints are

$$\left.\begin{array}{rcl} w_0+\sum\limits_{i=1}^{I} w_i x_{is}-\epsilon & \leq & f^{-1}(y_s) \\ -w_0-\sum\limits_{i=1}^{I} w_i x_{is}-\epsilon & \leq & -f^{-1}(y_s) \end{array}\right\}; \quad s=1,\ldots,S \tag{12.12}$$

In the case of option 4 we have

$$\begin{array}{rcl} \sum\limits_{r=1}^{R} \alpha_r\phi_r(y_{s_1}) & \leq & \sum\limits_{r=1}^{R} \alpha_r\phi_r(y_{s_2});\ \ \forall y_{s_1}<y_{s_2} \\ \sum\limits_{r=1}^{R} \alpha_r\phi_r(y_0) & = & 1 \end{array}$$

and for option 5, to the last two constraints we must add

$$\begin{array}{rcl} w_0+\sum\limits_{i=1}^{I} w_i x_{is}-\sum\limits_{r=1}^{R} \alpha_r\phi_r(y_s)-\epsilon & \leq & 0 \\ -w_0-\sum\limits_{i=1}^{I} w_i x_{is}+\sum\limits_{r=1}^{R} \alpha_r\phi_r(y_s)-\epsilon & \leq & 0 \end{array}$$

4. **Function to be optimized.** We have the following cases:

Option 1. Minimize

$$Q_1=\sum_{s=1}^{S}\delta_s^2=\sum_{s=1}^{S}\left(y_s-f(w_0+\sum_{i=1}^{I} w_i x_{is})\right)^2 \tag{12.13}$$

Option 2. Minimize

$$Q_2=\sum_{s=1}^{S}\epsilon_s^2 \quad = \quad \sum_{s=1}^{S}\left(w_0+\sum_{i=1}^{I} w_i x_{is}-f^{-1}(y_s)\right)^2 \tag{12.14}$$

Option 3 and 5. Minimize ϵ.

Option 2. Minimize

$$Q_4=\sum_{s=1}^{S}\epsilon_s^2 \quad = \quad \sum_{s=1}^{S}\left(w_0+\sum_{i=1}^{I} w_i x_{is}-\sum_{r=1}^{R}\alpha_r\phi_r(y_s)\right)^2$$

Once these four elements have been identified, we are ready to solve the problem.

Example 12.1 (Neural network). Consider de data in the first three columns of Table 12.1, which gives the outputs y_s obtained for the corresponding inputs $\{x_{1s}, x_{2s}\}$ for $s=1,\ldots,30$. The data have been simulated from the model

$$y_s=(0.3+0.3x_{1s}+0.7x_{2s})^2+\epsilon_s$$

where $x_{is} \sim U(0, 0.5)$ and $\epsilon_s \sim U(-0.005, 0.005)$.

With the aim of learning the relation $y_s = q(x_1, x_2)$, from these data, we decide to use a one-layer neural network with two inputs $\{x_1, x_2\}$ and one output y, and use the three options with known neuron function, and the two options where the neuron function is learned, described above. For the sake of illustration, a neural function $f(x) = \arctan(x)$, and the following basic functions have been used:

$$\{\phi_1(x), \phi_2(x), \phi_3(x)\} = \{\sqrt{x}, x, x^2\}$$

The following input GAMS code was used to solve this problem. Note that five different models with five different objective functions have been used. Note also the implementation of the bisection method.

```
$title Neural networks (neural2)

SETS
S number of data vectors/1*30/
I dimension of data input vectors/1*2/
J number of different learning functions/1*5/
R number of basic functions/1*3/;

ALIAS(S,S1)

PARAMETERS
X(I,S) input data vectors
Y(S) output data
powers(R) exponents of x in the basic functions
/1 0.5
2  1
3  2/;

X(I,S)=uniform(0,1)*0.5;
Y(S)=sqr(0.3+0.3*X('1',S)+0.7*X('2',S))+(uniform(0,1)-0.5)*0.01;

PARAMETERS
aux auxiliary parameter
x1 auxiliary parameter
x2 auxiliary parameter
x3 auxiliary parameter
f1 auxiliary parameter
f2 auxiliary parameter
f3 auxiliary parameter
ymin minimum value of Y(S)
ymax maximum value of Y(S)
acterror maximum prediction error
maxerror maximum allowed error for the bisection method;

maxerror=0.00001;

VARIABLES
z1 function1 to be optimized
z2 function2 to be optimized
z3 function3 to be optimized
z4 function21 to be optimized
z5 function21 to be optimized
```

```
W0 threshold value
W(I) weight associated with component i of input
alfa(R) coefficients of the neural function;

POSITIVE VARIABLES
epsilon error;

EQUATIONS
Q1 definition of z1
Q2 definition of z2
Q3 definition of z3
Q4 definition of z4
Q5 definition of z5
const1(S) upper error bound
const2(S) lower error bound
const3(S) upper error bound
const4(S) lower error bound
normalized normalized values
increasing(S,S1) the f function must be increasing;

Q1..z1=e=sum(S,sqr(Y(S)-arctan(W0+sum(I,W(I)*X(I,S)))));
Q2..z2=e=sum(S,sqr(W0+sum(I,W(I)*X(I,S))-sin(Y(S))/cos(Y(S))));
Q3..z3=e=epsilon;
Q4..z4=e=sum(S,sqr(W0+sum(I,W(I)*X(I,S))
            -sum(R,alfa(R)*Y(S)**(powers(R)))));
Q5..z5=e=epsilon;
const1(S)..W0+sum(I,W(I)*X(I,S))-epsilon=l=sin(Y(S))/cos(Y(S));
const2(S)..-W0-sum(I,W(I)*X(I,S))-epsilon=l=-sin(Y(S))/cos(Y(S));
const3(S)..W0+sum(I,W(I)*X(I,S))
      -sum(R,alfa(R)*Y(S)**(powers(R)))-epsilon=l=0.0;
const4(S)..-W0-sum(I,W(I)*X(I,S))
      +sum(R,alfa(R)*Y(S)**(powers(R)))-epsilon=l=0.0;
normalized..sum(R,alfa(R)*Y('1')**(powers(R)))=e=1;
increasing(S,S1)$(Y(S)<Y(S1))..sum(R,alfa(R)*Y(S)**(powers(R)))
      =l=sum(R,alfa(R)*Y(S1)**(powers(R)));;

MODEL reg1/Q1/;
MODEL reg2/Q2/;
MODEL reg3/Q3,const1,const2/;
MODEL reg4/Q4,normalized,increasing/;
MODEL reg5/Q5,const3,const4,normalized,increasing/;

file out/neural3.out/;
put out;

put "Data"/;
ymin=0.0;
ymax=0.0;
loop(S,
  loop(I,put X(I,S):12:3," & ";);
  put Y(S):12:3,"\\"/;
  if(ymin<Y(S),ymin=Y(S));
  if(ymax>Y(S),ymax=Y(S));
);
loop(J,
  if(ord(J) eq 1, SOLVE reg1 USING nlp MINIMIZING z1;
        put "z1=",z1.l:12:9/;);
  if(ord(J) eq 2, SOLVE reg2 USING nlp MINIMIZING z2;
        put "z2=",z2.l:12:9/;);
  if(ord(J) eq 3, SOLVE reg3 USING  lp MINIMIZING z3;
        put "z3=",z3.l:12:9/;);
  if(ord(J) eq 4, SOLVE reg4 USING nlp MINIMIZING z4;
        put "z4=",z4.l:12:9/;);
  if(ord(J) eq 5, SOLVE reg5 USING  lp MINIMIZING z5;
        put "z5=",z5.l:12:9/;);
  put "Weights:"/;
  put W0.l:12:3;
  loop(I,put " & ",W.l(I):12:3;);
  put " "/;
  if(((ord(J) eq 4) or (ord(J) eq 5)),
    put "Alfas:"/;
    loop(R,put " & ",alfa.l(R):12:3;);
    put " "/;
  );
  acterror=-10000;
  put "Data and fitted values:"//;

  loop(S,
    aux=W0.l+sum(I,W.l(I)*X(I,S));
    if(ord(J) le 3,
      f1=arctan(aux);
    else
      x1=ymin-(ymax-ymin)*0.1;
      x2=ymax+(ymax-ymin)*0.1;
      f1=sum(R,alfa.l(R)*abs(x1)**(powers(R)))-aux;
      f2=sum(R,alfa.l(R)*abs(x2)**(powers(R)))-aux;
      if(f1*f2>0.0,
        put "ERROR IN BISECTION "," S=",S.tl:3," aux=",
                aux:12:3," f1=",f1:12:3," f2=",f2:12:3/;
      else
        while(abs(x1-x2)>maxerror,
          x3=(x1+x2)*0.5;
          f3=sum(R,alfa.l(R)*x3**(powers(R)))-aux;
          if(f3*f1>0.0,
            x1=x3;f1=f3;
          else
            x2=x3;f2=f3;
          );
        );
      );
      f1=((x1+x2)*0.5);
    );
    loop(I,put X(I,S):12:3," & ";);
    aux=(f1-Y(S));
    if(abs(aux)>acterror,acterror=abs(aux));
    put Y(S):12:3," & ",aux:12:3"\\"/;
  );
  put "Maximum error=",acterror:12:9/;
);
```

The corresponding errors are shown in Table 12.1. Finally, Table 12.2 shows the threshold values, the weights, and the α coefficients (only for options 4 and 5), together with the optimal values of the corresponding objective functions.

Table 12.1: Input and output data for a one-layer neural network

Input		Output	ϵ_s Errors				
x_{1s}	x_{2s}	y_s	Q_1	Q_2	Q_3	Q_4	Q_5
0.086	0.055	0.129	-0.023	-0.024	-0.020	0.009	0.004
0.422	0.251	0.360	0.005	0.007	0.012	0.007	0.008
0.275	0.080	0.193	-0.002	-0.003	0.002	-0.004	-0.004
0.151	0.436	0.426	-0.015	-0.014	-0.006	-0.002	-0.008
0.146	0.133	0.187	0.000	-0.001	0.005	-0.001	-0.003
0.112	0.143	0.183	0.000	-0.001	0.005	0.001	-0.001
0.175	0.297	0.315	0.006	0.006	0.013	-0.002	-0.002
0.428	0.361	0.466	-0.024	-0.022	-0.016	-0.004	-0.008
0.034	0.314	0.280	0.008	0.008	0.015	-0.003	-0.005
0.250	0.232	0.287	0.010	0.010	0.016	-0.001	0.000
0.499	0.207	0.351	0.007	0.008	0.013	0.006	0.009
0.289	0.059	0.181	-0.002	-0.002	0.002	0.000	-0.001
0.496	0.157	0.308	0.013	0.014	0.018	0.005	0.009
0.381	0.023	0.190	-0.007	-0.008	-0.004	-0.007	-0.006
0.065	0.169	0.191	-0.002	-0.003	0.003	-0.003	-0.006
0.320	0.091	0.214	0.000	0.000	0.004	-0.007	-0.006
0.080	0.323	0.300	0.008	0.009	0.016	0.000	-0.002
0.125	0.280	0.281	0.011	0.011	0.018	0.000	0.000
0.334	0.385	0.451	-0.021	-0.019	-0.013	-0.004	-0.008
0.218	0.149	0.216	0.008	0.008	0.013	-0.001	-0.001
0.180	0.331	0.340	0.006	0.007	0.014	0.004	0.002
0.176	0.378	0.376	0.002	0.003	0.010	0.007	0.004
0.066	0.314	0.289	0.009	0.009	0.017	-0.001	-0.003
0.075	0.142	0.178	-0.007	-0.009	-0.003	-0.003	-0.006
0.295	0.043	0.172	-0.003	-0.004	0.000	0.002	0.001
0.415	0.051	0.209	0.007	0.007	0.010	0.000	0.001
0.115	0.321	0.311	0.008	0.008	0.015	0.000	-0.001
0.333	0.273	0.347	0.006	0.007	0.012	0.004	0.005
0.388	0.016	0.181	-0.002	-0.002	0.001	0.000	0.000
0.152	0.396	0.393	-0.009	-0.008	-0.001	-0.003	-0.007
Maximum absolute error			0.024	0.024	0.019	0.009	0.009

A simple comparison shows that the errors for the objective functions, Q_4 and Q_5, where the neural functions have been learned, lead to smaller errors, as expected.

■

Table 12.2: Values of the objective functions, threshold values, weights, and coefficients associated with the five options

Objective function	Threshold	Weights		Alphas		
Q	w_0	w_1	w_2	α_1	α_2	α_3
$Q_1 = 0.00292$	0.032	0.353	0.803	–	–	–
$Q_2 = 0.00375$	0.029	0.358	0.812	–	–	–
$Q_3 = 0.01978$	0.034	0.354	0.823	–	–	–
$Q_4 = 0.00086$	0.940	0.369	0.847	4.199	−4.238	2.384
$Q_5 = 0.00988$	0.924	0.415	0.912	4.200	−4.269	2.596

12.2 Applications to CAD

In this section we discuss an application to automatic mesh generation.

Automatic mesh generation is an area that has attracted the interest of many researchers. The most common mesh generation procedures include triangle, in two dimensions, and tetrahedral and hexahedral, in three dimensions, generation methods. Most generation problems involve building partitioning elements on arbitrary three-dimensional surfaces. The main surface mesh generation techniques belong to three main groups:

1. **Parametric space based techniques.** Since surfaces in parametric form have an $u-v$ representation, one can use standard techniques to mesh the $u-v$ parametric region, and then move to the associated $x-y-z$ space (see, for example, Cuilliere [29], and Farouki [36]). The main shortcoming of a direct application of this technique is that the resulting elements can be poorly shaped. Thus, some modifications are normally required.

2. **Direct techniques.** These methods proceed by advancing through the surface itself, by generating elements based on previously generated elements, and using surface normals and tangents to compute the direction of the advancing front.

3. **Optimization-based methods.** These methods are based on optimizing a quality function subject to some constraints. Nodes are moved in a direction such that an improvement of the objective function is obtained (Canann et al. [19], Freitag [42] and Li [67]).

In addition to the mesh generation problem, since most mesh generation techniques output meshes that are not optimal, some postprocessing to improve the quality of the elements is normally required. The two main categories of mesh improvement are

1. **Smoothing.** This consists of adjusting node locations without altering connectivities.

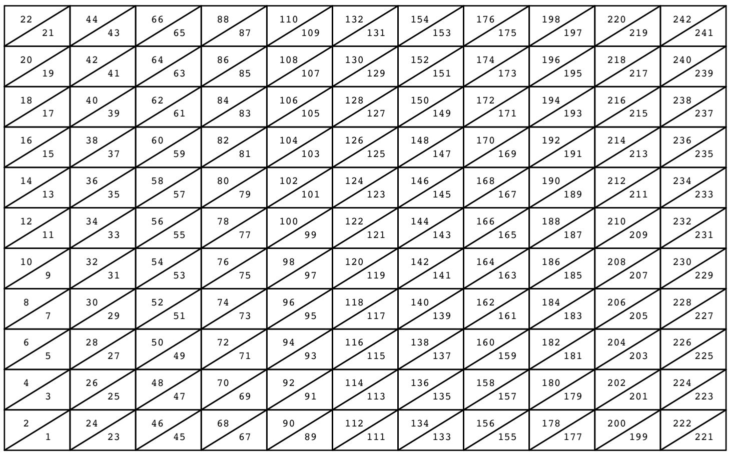

Figure 12.2: Topology of the selected mesh.

2. **Polishing.** This consists of changing the topology or the elements connectivities. Polishing methods use some quality criteria to operate, and can be combined with smoothing techniques (see Freitag [44]).

3. **Refinement.** This consists of reducing the element size. The reduction in size may be required to reproduce better the local reality, or to improve the quality of the elements. In fact, refinement can be seen as a mesh generation. Thus, mesh generation techniques are applicable here.

12.2.1 Automatic Mesh Generation

In this section we describe how to generate or improve a mesh using an optimization based method. This application is based on Castillo and Mínguez [23].

The method starts from a given projected mesh topology (the projection of the surface mesh on the X–Y plane), as that shown in Figure 12.1. In this projection, the corner nodes are assumed to be fixed, the boundary nodes can move in the boundary, and the interior nodes are free to move in any direction. The surface triangularization is selected for minimizing the standard deviation or variance of the triangle areas. Thus, we look for one mesh with the maximum similarity in terms of the triangle areas. In other words, we minimize

$$Q = \sum_{i=1}^{m} \frac{[a(i) - \bar{a}]^2}{m}$$

subject to the boundary constraints, where $a(i)$ is the area of the triangle i, $\bar{a}$ is the corresponding mean, and m is the number of triangles.

In summary, we proceed as follows:

Step 1. The topology of the mesh is defined by a projected mesh. A compact definition of one possible and simple mesh is given in Table 12.3, where (x_{min}, y_{min}) and (x_{max}, y_{max}) are the left-top and right-bottom corner coordinates, and n is the number of nodes per column ($n = 12$ for the mesh in Figure 12.2). The resulting mesh has n^2 nodes and $2(n-1)^2$ elements.

The following three main components of the mesh are defined:

(a) Node coordinates

(b) Nodes belonging to each element

(c) Boundary nodes

Step 2. The surface to be triangulated is defined by its corresponding equations.

Step 3. The degrees of freedom of all the nodes are defined. Figure 12.3 shows the degrees of freedom of the different sets of nodes. Some nodes corresponding to the corner nodes are fixed, having no degrees of freedom. Some nodes, the boundary no corner nodes, have 1 degree of freedom, which is shown by an arrow indicating the free directions of their possible displacements. Finally, the rest, the interior nodes, have 2 degrees of freedom, marked by two arrows showing their possible displacements.

Step 4. The objective function Q is minimized subject to the boundary constraints, and the final projected and surface meshes are obtained (see Figure 12.4).

To avoid overlapping of triangles, we can force the total area of the projected mesh to remain constant and equal to the initial one. This simple constraint avoids stability problems caused by the overlapping of triangles.

Example 12.2 (Surface triangulation). Consider the surface with equation

$$z = \frac{xy(x^2 - y^2)}{x^2 + y^2}$$

defined on the square

$$\{(x, y)| -2 \le x \le 2;\ -2 \le y \le 2\}$$

Applying the proposed method with a rectangular net of size $n = 12$, and

$$(x_{min}, y_{min}) = (-2, -2);\ \ (x_{max}, y_{max}) = (2, 2)$$

the triangulated surface in Figure 12.4 is obtained. Note that the triangles in the vertical projection have been adapted to get 3D triangles with similar area. ■

Table 12.3: Parametric definition of a rectangular net

Coordinates of node k	
x_k	$x_{min} + floor((k-1)/n))(x_{max} - x_{min})/(n-1)$
y_k	$y_{min} + mod(k-1, n)(y_{max} - y_{min})/(n-1)$
Nodes of element i for i even	
First node	(floor((i-1)/(2(n-1))))n+floor(mod((i-1),(2(n-1)))/2+1)
Second node	(floor((i-1)/(2(n-1))))n+floor(mod((i-1),(2(n-1)))/2+1)+1
Third node	(floor((i-1)/(2(n-1))))n+floor(mod((i-1),(2(n-1)))/2+1)+n+1
Nodes of element i for i odd	
First node	(floor((i-1)/(2(n-1))))n+floor(mod((i-1),(2(n-1)))/2+1)
Second node	(floor(i-1)/(2(n-1))))n+floor(mod((i-1),(2(n-1)))/2+1)+n
Third node	(floor((i-1)/(2(n-1))))n+floor(mod((i-1),(2(n-1)))/2+1)+n+1
Boundary elements	
Lower side	$n(j-1)+1;\ j = 1, 2, \ldots, n$
Upper side	$nj;\ j = 1, 2, \ldots, n$
Right-hand side	$n^2 - n + j;\ j = 1, 2, \ldots, n$
Left-hand side	$j;\ j = 1, 2, \ldots, n$

```
$title Mesh

SETS
I number of elements/1*242/
J number of nodes per element/1*3/
K number of nodes/1*144/
N number of nodes per row/1*12/
D space dimension/1*2/
XFIXED(K) nodes with fixed X
YFIXED(K) nodes with fixed Y;

ALIAS(K,K1,K2);


PARAMETER
t number of squares per dimension
m number of elements per column
xmin minimum value of coordinate x
xmax maximum value of ccordinate x
ymin minimum value of coordinate y
ymax maximum value of ccordinate y
COORD(K,D) coordinates of the initial mesh nodes;

t=card(N);
xmin=-2;
xmax=2;
ymin=-2;
ymax=2;

m=2*t-2;
```

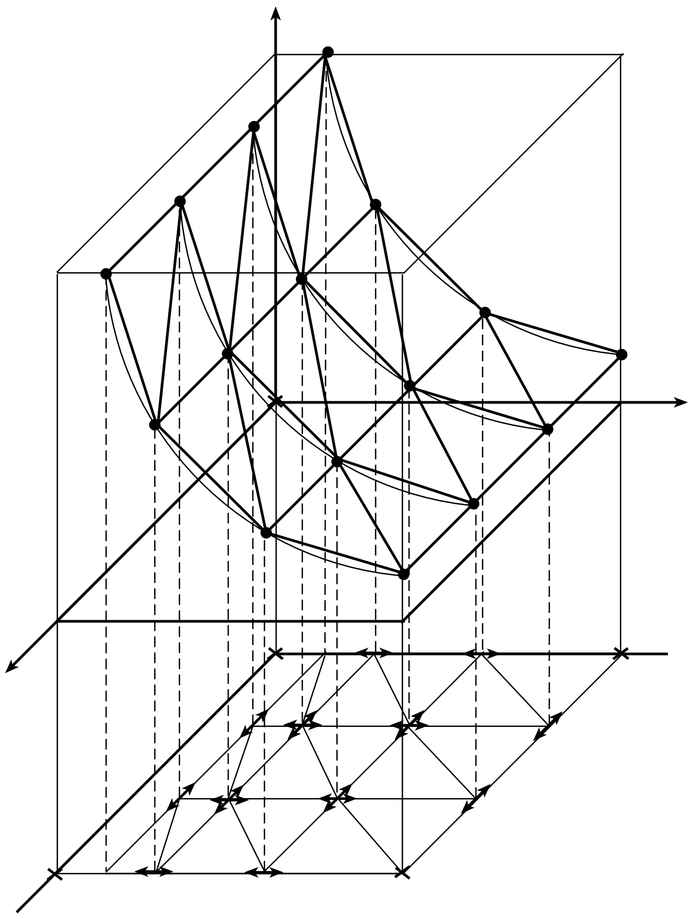

Figure 12.3: Illustration of how the net is generated, showing the degrees of freedom of the nodes.

```
XFIXED(K)=no;
loop(N,
XFIXED(K)$(ord(K) eq ord(N))=yes;
XFIXED(K)$(ord(K) eq ord(N)+t*t-t)=yes;
YFIXED(K)$(ord(K) eq t*ord(N))=yes;
YFIXED(K)$(ord(K) eq (ord(N)-1)*t+1)=yes;
);

COORD(K,'1')=xmin+floor((ord(K)-1)/t)*(xmax-xmin)/(t-1);
COORD(K,'2')=ymin+mod(ord(K)-1,t)*(ymax-ymin)/(t-1);

PARAMETER

EL(I,J);

loop(I,
```

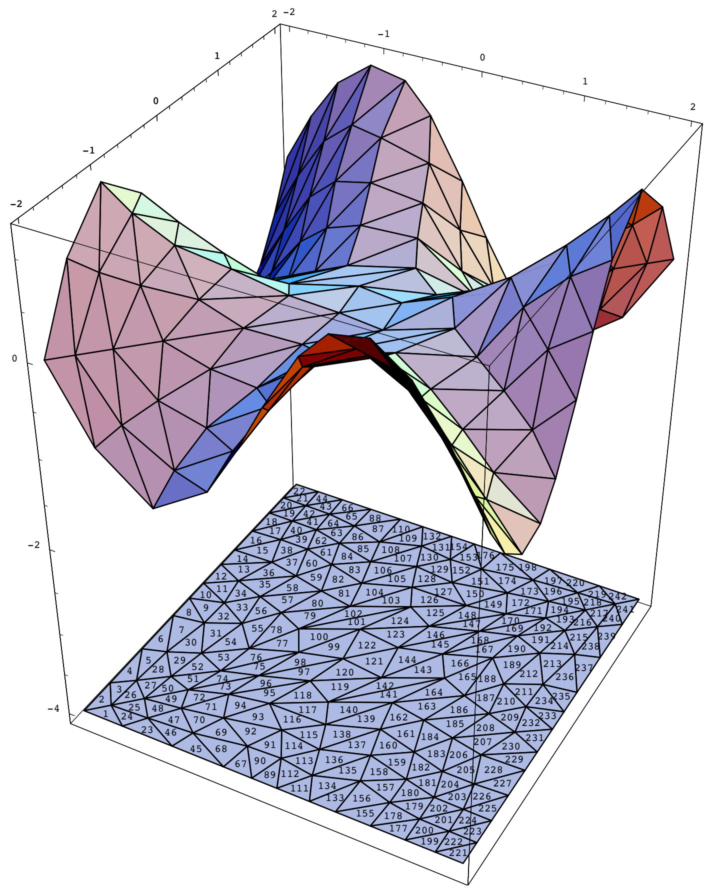

Figure 12.4: Example of a triangulated surface showing the basic mesh and the mesh leading to a minimum variance of triangle areas.

```
if(mod(ord(I),2) eq 0,
EL(I,'1')=(floor((ord(I)-1)/m))*t+floor(mod((ord(I)-1),m)/2+1);
EL(I,'2')=(floor((ord(I)-1)/m))*t+floor(mod((ord(I)-1),m)/2+1)+1;
EL(I,'3')=(floor((ord(I)-1)/m))*t+floor(mod((ord(I)-1),m)/2+1)+t+1;
else
 EL(I,'1')=(floor((ord(I)-1)/m))*t+floor(mod((ord(I)-1),m)/2+1);
EL(I,'2')=(floor((ord(I)-1)/m))*t+floor(mod((ord(I)-1),m)/2+1)+t;
EL(I,'3')=(floor((ord(I)-1)/m))*t+floor(mod((ord(I)-1),m)/2+1)+t+1;
);
);

VARIABLES
z1 function to be optimized
```

```
X(K) coordinate of node K
Y(K) coordinate of node K
Z(K) coordinate of node K
area(I) area of element I
mean mean area of elements;

X.l(K)=COORD(K,'1');
Y.l(K)=COORD(K,'2');

X.fx(XFIXED)=COORD(XFIXED,'1');
Y.fx(YFIXED)=COORD(YFIXED,'2');

EQUATIONS
Q objective function to be optimized
Carea(I)
Cf(K) points are in surface
Cmean;

Q..z1=e=sum(I,sqr(area(I)-mean))/card(I);

Carea(I)..area(I)=e=sum((K,K1,K2)$((EL(I,'1')
   =ord(K)) and (EL(I,'2')=ord(K1)) and (EL(I,'3')=ord(K2))),
sqr((Y(K1)-Y(K))*(Z(K2)-Z(K))-(Y(K2)-Y(K))*(Z(K1)-Z(K)))+
sqr((Z(K1)-Z(K))*(X(K2)-X(K))-(Z(K2)-Z(K))*(X(K1)-X(K)))+
sqr((X(K1)-X(K))*(Y(K2)-Y(K))-(X(K2)-X(K))*(Y(K1)-Y(K))));
Cf(K)..Z(K)=e=(X(K)*Y(K)*(sqr(X(K))-sqr(Y(K))))/(sqr(X(K))+sqr(Y(K)));
Cmean..mean=e=sum(I,area(I))/card(I);

MODEL mallas/ALL/;

SOLVE mallas USING nlp MINIMIZING z1;

file out/mallas3.out/;
put out;


if((card(I) ne (2*(t-1)*(t-1))),
put "Error, card(I) must be equal to ",(2*(t-1)*(t-1)):8:0/;
else
if((card(K) ne t*t),
put "Error, card(K) must be equal to ",(t*t):8:0/;
else
put "z1=",z1.l:12:8," mean=",mean.l:12:8, " cpu=",mallas.resusd:9:2,
     " modelstat=",mallas.modelstat:3:0," solvestat=",mallas.solvestat:3:0/;

loop(K,
put "X(",K.tl:3,")=",COORD(K,'1'):12:8," Y(",K.tl:3,")=",COORD(K,'2'):12:8/;
);

loop(I,
put "I=",i.tl:3," Nodes=",EL(I,'1'):5:0," ",EL(I,'2'):5:0," ",EL(I,'3'):5:0/;
);

loop(K,
put "K=",K.tl:3," X=",X.l(K):12:6," Y=",Y.l(K):12:6," Z=",Z.l(K):12:6/;
);

loop(I,
put "I=",I.tl:3," area=",area.l(I):12:6/;
);

put "COORDINATES"/;
put "{";

loop(K,
if(ord(K)<card(K),
put "{",X.l(K):12:8,",",Y.l(K):12:8,",",Z.l(K):12:8,"},"/;
else
put "{",X.l(K):12:8,",",Y.l(K):12:8,",",Z.l(K):12:8,"}}"/;
);
);

put "ELEMENTS"/;
put "{";

loop(I,
if(ord(I)<card(I),
put "{",EL(I,'1'):5:0,",",EL(I,'2'):5:0,",",EL(I,'3'):5:0,"},"/;
else
put "{",EL(I,'1'):5:0,",",EL(I,'2'):5:0,",",EL(I,'3'):5:0,"}}"/;
););););
```

12.3 Applications to Probability

In this section we present some applications relevant to probability. We deal with the problem of compatibility of conditional distributions, which arises in many fields of applications, as artificial intelligence and expert systems, for example. The assessment of probabilities is crucial in probability based models arising in these areas. This assessment can be done by considering the joint distributions of the variables involved or, more easily, by considering several sets of conditional distributions and/or marginal ones, because they are of lower dimensions. However, conditionals probabilities are closely interrelated and, consequently, they cannot be freely assessed. Even for a qualified expert, the chances of violating the probability axioms during the assessment process are close to one. So, the problem of compatibility must be dealt with, and the help of a computer is unavoidable in practice. In this section, we show how this problem can be solved using linear programming techniques (see Arnold et al. [1, 2, 3, 4, 5] for a more complete treatment of this problem).

12.3.1 Compatibility of Conditional Probability Matrices

Consider a bidimensional discrete random variable (X, Y), with domain $N = \{(x_i, y_j)|i = 1, 2, \dots, I; j = 1, 2, \dots, J\}$, where X and Y take possible values x_1, x_2, … x_I and $y_1, y_2, \dots, y_J$, respectively.

Let $\mathbf{P}$ be its joint probability matrix with $p_{ij} = \text{Prob}[X = x_i, Y = y_j]$. The column vectors

$$\begin{aligned} p_{i.} &= \sum_j^J p_{ij} \\ p_{.j} &= \sum_i^I p_{ij} \end{aligned}$$

where the dot is used to refer to sum in the corresponding index, give the probabilities of X and Y, respectively, after ignoring the other variable, and are called marginal probabilities of X and Y, respectively.

The conditional probability matrices **A** and **B** are defined as

$$a_{ij} = \text{Prob}[X = x_i | Y = y_j] = \frac{p_{ij}}{p_{.j}} \tag{12.15}$$

and

$$b_{ij} = \text{Prob}[Y = y_j | X = x_i] = \frac{p_{ij}}{p_{i.}} \tag{12.16}$$

respectively, and give the probabilities of one variable when the value of the other variable has been fixed.

Note that **A** and **B** exists only if

$$\text{Prob}[Y = y_j] = \sum_i^I p_{ij} = p_{.j} \neq 0$$

and

$$\text{Prob}[X = x_i] = \sum_j^J p_{ij} = p_{i.} \neq 0$$

respectively.

From (12.15) and (12.16) it follows that the columns of **A** and the rows of **B** sum to 1. Thus, valid conditional probability matrices **A** and **B** must have nonnegative elements such that **A** has columns and **B** has rows that sum to 1.

If the joint probabilities p_{ij} are given, the conditional probability matrices **A** and **B** can be obtained using (12.15) and (12.16), respectively. However, obtaining the matrix **P** from **A** and **B** is not a trivial task. In fact, **A** and **B** can be chosen in such form that there is no **P** satisfying (12.15) and (12.16). In this case we say that **A** and **B** are incompatible. If, on the other hand, this matrix **P** exists, we say that **A** and **B** are compatible.

Consequently, **A** and **B** are compatible if and only if a joint distribution **P** with **A** and **B** as its associated conditionals exists, that is, such that

$$p_{ij} = a_{ij} p_{.j}; \ \forall i, j \tag{12.17}$$

$$p_{ij} = b_{ij} p_{i.}; \ \forall i, j \tag{12.18}$$

$$p_{ij} \geq 0, \quad \forall i, j, \tag{12.19}$$

$$\sum_{i=1}^I \sum_{j=1}^J p_{ij} = 1 \tag{12.20}$$

Balow we give three possible methods to determine the compatibility of **A** and **B**:

Method 1 (based on P). Seek a joint probability matrix **P** satisfying

$$\begin{array}{rcl} p_{ij} - a_{ij} \sum\limits_{i=1}^{I} p_{ij} &=& 0, \;\; \forall i,j \\ p_{ij} - b_{ij} \sum\limits_{j=1}^{J} p_{ij} &=& 0, \;\; \forall i,j \\ \sum\limits_{i=1}^{I} \sum\limits_{j=1}^{J} p_{ij} &=& 1 \\ p_{ij} &\geq& 0, \;\; \forall i,j \end{array} \tag{12.21}$$

In this method **P** is directly sought, and it involves $2|N|+1$ equations in $|N|$ unknowns, where $|N|$ is the cardinality of the domain set N of (X,Y).

Method 2 (based on the two marginals). Seek two probability vectors $\boldsymbol{\tau}$ and $\boldsymbol{\eta}$ satisfying

$$\begin{array}{rcl} \eta_j a_{ij} - \tau_i b_{ij} &=& 0, \;\; \forall i,j \\ \sum\limits_{i=1}^{I} \tau_i &=& 1 \\ \sum\limits_{j=1}^{J} \eta_j &=& 1 \\ \tau_i \geq 0, \forall i, \;\; \eta_j &\geq& 0, \;\; \forall j \end{array} \tag{12.22}$$

In this method we seek the two marginal probability vectors, $\boldsymbol{\tau}$ and $\boldsymbol{\eta}$, which, combined with **A** and **B**, give **P** $(p_{ij} = \eta_j a_{ij} = \tau_i b_{ij})$. This involves $|N|+2$ equations in $I+J$ unknowns.

Method 3 (based on one marginal). Seek one probability vector $\boldsymbol{\tau}$ satisfying

$$\begin{array}{rcl} a_{ij} \sum\limits_{k=1}^{I} \tau_k b_{kj} - \tau_i b_{ij} &=& 0, \;\; \forall i,j \\ \sum\limits_{i=1}^{I} \tau_i &=& 1 \\ \tau_i &\geq& 0, \;\; \forall i \end{array} \tag{12.23}$$

In this method, we seek the X-marginal $\boldsymbol{\tau}$ which, combined with **B**, gives **P** $(p_{ij} = \tau_i b_{ij})$. It involves $|N|+1$ equations in I unknowns.

All three methods involve linear equations to be solved subject to non-negativity constraints. However, since system (12.23) is the one that involves a smaller number of variables, it is probably the one we will use to try to solve our problem.

Example 12.3 (Compatibility). Consider the matrices **A** and **B**:

$$\mathbf{A} = \begin{pmatrix} 0.0667 & 0.1905 & 0.3750 & 0.1176 & 0.0769 & 0.1111 \\ 0.1333 & 0.1905 & 0.0000 & 0.1765 & 0.3846 & 0.3333 \\ 0.3333 & 0.1905 & 0.1250 & 0.1765 & 0.0769 & 0.1111 \\ 0.2000 & 0.0952 & 0.2500 & 0.1176 & 0.0769 & 0.1111 \\ 0.2000 & 0.1905 & 0.1250 & 0.1765 & 0.3077 & 0.2222 \\ 0.0667 & 0.1429 & 0.1250 & 0.2353 & 0.0769 & 0.1111 \end{pmatrix} \tag{12.24}$$

$$\mathbf{B} = \begin{pmatrix} 0.0833 & 0.3333 & 0.2500 & 0.1667 & 0.0833 & 0.0833 \\ 0.1176 & 0.2353 & 0.0000 & 0.1765 & 0.2941 & 0.1765 \\ 0.3333 & 0.2667 & 0.0667 & 0.2000 & 0.0667 & 0.0667 \\ 0.2727 & 0.1818 & 0.1818 & 0.1818 & 0.0909 & 0.0909 \\ 0.1765 & 0.2353 & 0.0588 & 0.1765 & 0.2353 & 0.1176 \\ 0.0909 & 0.2727 & 0.0909 & 0.3636 & 0.0909 & 0.0909 \end{pmatrix} \tag{12.25}$$

The following GAMS code generates a random joint probability matrix and the corresponding conditionals. Next, it obtains the joint probability based on A and B using methods 1–3.

```
$title Compatibility Method I

SETS
I number of rows/1*6/
J number of columns/1*6/;

ALIAS(I,I1);
ALIAS(J,J1);

PARAMETER
PP(I,J) Joint probability matrix used to generate conditionals
A(I,J) conditional probability of X given Y
B(I,J) conditional probability of Y given X;

PP(I,J)=round(uniform(0,5));
PP(I,J)=PP(I,J)/sum((I1,J1),PP(I1,J1));
A(I,J)=PP(I,J)/sum(I1,PP(I1,J));
B(I,J)=PP(I,J)/sum(J1,PP(I,J1));

VARIABLE
z value of the objective function;

POSITIVE VARIABLES
P(I,J) Joint probability looked for
TAU(I) Marginal probability of X
ETA(J) Marginal probability of Y;

EQUATIONS
ZDEF1 Function to be optimized in model 1
ZDEF2 Function to be optimized in model 2
ZDEF3 Function to be optimized in model 3
CONST11(I,J)
CONST12(I,J)
CONST2(I,J)
CONST3(I,J)
```

```
NORM1 P add up to one
NORM2 TAU add up to one
NORM3 ETA add up to one;

ZDEF1..z=e=P('1','1');
ZDEF2..z=e=TAU('1');
CONST11(I,J)..P(I,J)-A(I,J)*sum(I1,P(I1,J))=e=0;
CONST12(I,J)..P(I,J)-B(I,J)*sum(J1,P(I,J1))=e=0;
CONST2(I,J)..ETA(J)*A(I,J)-TAU(I)*B(I,J)=e=0;
CONST3(I,J)..A(I,J)*sum(I1,TAU(I1)*B(I1,J))-TAU(I)*B(I,J)=e=0;
NORM1..sum((I,J),P(I,J))=e=1;
NORM2..sum(I,TAU(I))=e=1;
NORM3..sum(J,ETA(J))=e=1;

MODEL model1/ZDEF1,CONST11,CONST12,NORM1/;
MODEL model2/ZDEF2,CONST2,NORM2,NORM3/;
MODEL model3/ZDEF2,CONST3,NORM2/;

file out/comp1.out/;
put out;

put "Initial Matrix PP:"/;
loop(I,loop(J,put PP(I,J):7:4;);put ""/;);
put "Matrix A:"/;
loop(I,loop(J,put A(I,J):7:4;);put ""/;);
put "Matrix B:"/;
loop(I,loop(J,put B(I,J):7:4;);put ""/;);

SOLVE model1 USING lp MINIMIZING z;
put "Final Matrix P Obtained by Model 1:"/;
loop(I,loop(J,put P.l(I,J):7:4;);put ""/;);
put "z=",z.l," modelstat=",model1.modelstat," solvestat=",model1.solvestat/;
SOLVE model2 USING lp MINIMIZING z;
put "Final Matrix P Obtained by Model 2:"/;
loop(I,loop(J,put (TAU.l(I)*B(I,J)):7:4;);put ""/;);
put "z=",z.l," modelstat=",model2.modelstat," solvestat=",model2.solvestat/;
SOLVE model3 USING lp MINIMIZING z;
put "Final Matrix P Obtained by Model 3:"/;
loop(I,loop(J,put (TAU.l(I)*B(I,J)):7:4;);put ""/;);
put "z=",z.l," modelstat=",model3.modelstat," solvestat=",model3.solvestat/;
```

The solution of this problem is the joint probability matrix

$$\mathbf{P} = \begin{pmatrix} 0.0120 & 0.0482 & 0.0361 & 0.0241 & 0.0120 & 0.0120 \\ 0.0241 & 0.0482 & 0.0000 & 0.0361 & 0.0602 & 0.0361 \\ 0.0602 & 0.0482 & 0.0120 & 0.0361 & 0.0120 & 0.0120 \\ 0.0361 & 0.0241 & 0.0241 & 0.0241 & 0.0120 & 0.0120 \\ 0.0361 & 0.0482 & 0.0120 & 0.0361 & 0.0482 & 0.0241 \\ 0.0120 & 0.0361 & 0.0120 & 0.0482 & 0.0120 & 0.0120 \end{pmatrix} \tag{12.26}$$

Since the corresponding programming problem has a solution, the conditional probability matrices **A** and **B** are compatible. ∎

Example 12.4 (Incompatibility). Consider the matrices **A** and **B**:

$$\mathbf{A} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}, \tag{12.27}$$

$$\mathbf{B} = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & 0 & \frac{2}{3} \end{pmatrix}. \tag{12.28}$$

Since the corresponding programming problem has no solution, the conditional probability matrices **A** and **B** are incompatible.

The reader is encouraged to modify the previous GAMS program to check that this problem is infeasible.

■

12.3.2 ϵ Compatibility

In Section 12.3.1 we described three linear equation formulations for the search for a matrix **P** compatible with given conditional matrices **A** and **B**. If, instead of exact compatibility, we are willing to accept approximate compatibility, we can replace Methods 1–3 by the following revised versions:

Revised method 1. Seek a probability matrix **P** to minimize ϵ subject to

$$\begin{array}{rcl} |p_{ij} - a_{ij} \sum\limits_{i=1}^{I} p_{ij}| & \leq & \epsilon\gamma_{ij}, \ \forall i,j \\ |p_{ij} - b_{ij} \sum\limits_{j=1}^{J} p_{ij}| & \leq & \epsilon\gamma_{ij}, \ \forall i,j \\ \sum\limits_{i=1}^{I} \sum\limits_{j=1}^{J} p_{ij} & = & 1 \\ p_{ij} & \geq & 0, \ \forall i,j \end{array} \tag{12.29}$$

Revised method 2. Seek two probability vectors, $\boldsymbol{\tau}$ and $\boldsymbol{\eta}$, to minimize ϵ subject to

$$\begin{array}{rcl} |\eta_j a_{ij} - \tau_i b_{ij}| & \leq & \epsilon\gamma_{ij}, \ \forall i,j \\ \sum\limits_{i=1}^{I} \tau_i & = & 1 \\ \sum\limits_{j=1}^{J} \eta_j & = & 1 \\ \tau_i \geq 0, \eta_j & \geq & 0, \forall i,j \end{array} \tag{12.30}$$

Revised method 3. Seek one probability vector $\boldsymbol{\tau}$ to minimize ϵ subject to

$$\begin{array}{rcl} |a_{ij} \sum\limits_{i=1}^{I} \tau_i b_{ij} - \tau_i b_{ij}| & \leq & \epsilon\gamma_{ij}, \ \forall i,j \\ \sum\limits_{i=1}^{I} \tau_i & = & 1 \\ \tau_i & \geq & 0, \ \forall i \end{array} \tag{12.31}$$

where γ_{ij} are numbers that measure the relative importance of the errors we are willing to accept for the corresponding values p_{ij} of the joint probabilities, and ϵ measures the global degree of incompatibility. Note that if $\epsilon = 0$, we have compatibility; otherwise we do not.

Our aim is to obtain the matrices $\mathbf{P}$, $\boldsymbol{\tau}$ and $\boldsymbol{\eta}$ that minimize ϵ, namely, the degree of incompatibility.

The associated problem is a LPP because each of the constraints involving absolute values in (12.29)–(12.31) can be replaced by two linear inequality constraints (see Chapter 13). So, all constraints can be written as linear constraints.

Example 12.5 (A compatible example). Consider the following matrices $\mathbf{A}$ and $\mathbf{B}$:

$$A = \begin{pmatrix} 0.1429 & 0.3333 & 0.2727 & 0.1667 \\ 0.1429 & 0.0833 & 0.1818 & 0.3333 \\ 0.0000 & 0.2500 & 0.4545 & 0.2500 \\ 0.7143 & 0.3333 & 0.0909 & 0.2500 \end{pmatrix} \tag{12.32}$$

$$B = \begin{pmatrix} 0.1000 & 0.4000 & 0.3000 & 0.2000 \\ 0.1250 & 0.1250 & 0.2500 & 0.5000 \\ 0.0000 & 0.2727 & 0.4545 & 0.2727 \\ 0.3846 & 0.3077 & 0.0769 & 0.2308 \end{pmatrix} \tag{12.33}$$

The GAMS code to build de joint and conditional probability matrices, and solve this problem by the revised methods 1–3 is:

```
$title Compatibility Methods 1, 2 and 3

SETS
I number of rows/1*4/
J number of columns/1*4/;

ALIAS(I,I1);
ALIAS(J,J1);

PARAMETER
PP(I,J) Joint probability matrix used to generate conditionals
A(I,J) conditional probability of X given Y
B(I,J) conditional probability of Y given X;

PP(I,J)=round(uniform(0,5));
PP(I,J)=PP(I,J)/sum((I1,J1),PP(I1,J1));
A(I,J)=PP(I,J)/sum(I1,PP(I1,J));
B(I,J)=PP(I,J)/sum(J1,PP(I,J1));

* The line below modifies the B matrix to get incompatibility

VARIABLE
z value of the objective function
epsilon Maximum error in joint probability;

POSITIVE VARIABLES
P(I,J) Joint probability looked for
TAU(I) Marginal probability of X
```

```
ETA(J) Marginal probability of Y;

EQUATIONS
ZDEF1 Function to be optimized in model 1
ZDEF2 Function to be optimized in model 2
ZDEF3 Function to be optimized in model 3
CONST111(I,J)
CONST112(I,J)
CONST121(I,J)
CONST122(I,J)
CONST21(I,J)
CONST22(I,J)
CONST31(I,J)
CONST32(I,J)
NORM1 P add up to one
NORM2 TAU add up to one
NORM3 ETA add up to one;

ZDEF1..z=e=epsilon;
CONST111(I,J)..P(I,J)-A(I,J)*sum(I1,P(I1,J))=l=epsilon;
CONST112(I,J)..-epsilon=l=P(I,J)-A(I,J)*sum(I1,P(I1,J));
CONST121(I,J)..P(I,J)-B(I,J)*sum(J1,P(I,J1))=l=epsilon;
CONST122(I,J)..-epsilon=l=P(I,J)-B(I,J)*sum(J1,P(I,J1));
CONST21(I,J)..ETA(J)*A(I,J)-TAU(I)*B(I,J)=l=epsilon;
CONST22(I,J)..-epsilon=l=ETA(J)*A(I,J)-TAU(I)*B(I,J);
CONST31(I,J)..A(I,J)*sum(I1,TAU(I1)*B(I1,J))-TAU(I)*B(I,J)=l=epsilon;
CONST32(I,J)..-epsilon=l=A(I,J)*sum(I1,TAU(I1)*B(I1,J))-TAU(I)*B(I,J);
NORM1..sum((I,J),P(I,J))=e=1;
NORM2..sum(I,TAU(I))=e=1;
NORM3..sum(J,ETA(J))=e=1;

MODEL model1/ZDEF1,CONST111,CONST112,CONST121,CONST122,NORM1/;
MODEL model2/ZDEF1,CONST21,CONST22,NORM2,NORM3/;
MODEL model3/ZDEF1,CONST31,CONST32,NORM2/;

file out/comp1.out/;
put out;

put "Initial Matrix PP:"/;
loop(I,loop(J,put PP(I,J):7:4," & ";);put ""/;);
put "Matrix A:"/;
loop(I,loop(J,put A(I,J):7:4," & ";);put ""/;);
put "Matrix B:"/;
loop(I,loop(J,put B(I,J):7:4," & ";);put ""/;);

SOLVE model1 USING lp MINIMIZING z;
put "Final Matrix P Obtained by Model 1:"/;
loop(I,loop(J,put P.l(I,J):7:4," & ";);put ""/;);
put "z=",z.l:12:8," modelstat=",model1.modelstat," solvestat=",model1.solvestat/;
SOLVE model2 USING lp MINIMIZING z;
put "Final Matrix P Obtained by Model 2:"/;
loop(I,loop(J,put (TAU.l(I)*B(I,J)):7:4," & ";);put ""/;);
put "z=",z.l:12:8," modelstat=",model2.modelstat," solvestat=",model2.solvestat/;
SOLVE model3 USING lp MINIMIZING z;
put "Final Matrix P Obtained by Model 3:"/;
loop(I,loop(J,put (TAU.l(I)*B(I,J)):7:4," & ";);put ""/;);
put "z=",z.l:12:8," modelstat=",model3.modelstat," solvestat=",model3.solvestat/;
```

The solution of this problem, which coincides with those for the three methods, is $\epsilon = 0$ and the resulting joint probability matrix

$$\mathbf{P} = \begin{pmatrix} 0.0238 & 0.0952 & 0.0714 & 0.0476 \\ 0.0238 & 0.0238 & 0.0476 & 0.0952 \\ 0.0000 & 0.0714 & 0.1190 & 0.0714 \\ 0.1190 & 0.0952 & 0.0238 & 0.0714 \end{pmatrix} \tag{12.34}$$

■

Example 12.6 (An incompatible example). If we modify the first row of the B matrix in Example 12.5 [in (12.33)], by removing the asterisk in the comment line, we get the output file:

```
Initial Matrix PP:
 0.0238 &  0.0952 &  0.0714 &  0.0476 &
 0.0238 &  0.0238 &  0.0476 &  0.0952 &
 0.0000 &  0.0714 &  0.1190 &  0.0714 &
 0.1190 &  0.0952 &  0.0238 &  0.0714 &
Matrix A:
 0.1429 &  0.3333 &  0.2727 &  0.1667 &
 0.1429 &  0.0833 &  0.1818 &  0.3333 &
 0.0000 &  0.2500 &  0.4545 &  0.2500 &
 0.7143 &  0.3333 &  0.0909 &  0.2500 &
Matrix B:
 0.2500 &  0.2500 &  0.2500 &  0.2500 &
 0.1250 &  0.1250 &  0.2500 &  0.5000 &
 0.0000 &  0.2727 &  0.4545 &  0.2727 &
 0.3846 &  0.3077 &  0.0769 &  0.2308 &
Final Matrix P Obtained by Model 1:
 0.0403 &  0.0630 &  0.0539 &  0.0494 &
 0.0289 &  0.0165 &  0.0549 &  0.1231 &
 0.0000 &  0.0579 &  0.1177 &  0.0784 &
 0.1329 &  0.0858 &  0.0129 &  0.0843 &
z=  0.01139203 modelstat=        1.00 solvestat=        1.00
Final Matrix P Obtained by Model 2:
 0.0478 &  0.0478 &  0.0478 &  0.0478 &
 0.0335 &  0.0335 &  0.0670 &  0.1340 &
 0.0000 &  0.0678 &  0.1131 &  0.0678 &
 0.1124 &  0.0899 &  0.0225 &  0.0674 &
z=  0.02107728 modelstat=        1.00 solvestat=        1.00
Final Matrix P Obtained by Model 3:
 0.0559 &  0.0559 &  0.0559 &  0.0559 &
 0.0309 &  0.0309 &  0.0618 &  0.1237 &
 0.0000 &  0.0524 &  0.0873 &  0.0524 &
 0.1295 &  0.1036 &  0.0259 &  0.0777 &
z=  0.02502453 modelstat=        1.00 solvestat=        1.00
```

which shows the initial P, A, and B matrices and the P matrices resulting from methods 1–3. Note that the A and B matrices are incompatible, where ϵ values for methods 1, 2, and 3 are 0.01139203, 0.02107728, and 0.02502453, respectively. ■

12.4 Regression Models

The second problem we deal with is the regression problem. Least-squares methods have succeeded because of the nice mathematical behavior of the associated quadratic function, which has derivatives and lead to simple systems of equations. However, other interesting methods, such as those based on absolute values or maxima, in spite of its importance, are rarely used because of the nonexistence of derivatives of the associated functions. Fortunately, linear programming techniques allow dealing with these problems in a very satisfactory way, as it is shown in this section.

Consider the standard linear model

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{12.35}$$

or, equivalently

$$y_i = \mathbf{z}_i^T \boldsymbol{\beta} + \varepsilon_i, \; i = 1, \ldots, n \tag{12.36}$$

where $\mathbf{y} = (y_1, \ldots, y_n)^T$ is an $n \times 1$ vector of response variables, $\mathbf{Z}$ is an $n \times p$ matrix of predictor variables, $\mathbf{z}_i^T$ is the ith row in $\mathbf{Z}$, $\boldsymbol{\beta}$ is a $p \times 1$ vector of regression coefficients or parameters, and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^T$ is an $n \times 1$ vector of random errors.

The most popular methods for estimating the regression parameters $\boldsymbol{\beta}$ are

1. The least squares (LS) method. This method minimizes

$$\sum_{i=1}^{n} (y_i - \mathbf{z}_i^t \boldsymbol{\beta})^2 \tag{12.37}$$

This is the standard regression model, and can be easily solved using standard and well known techniques. This method penalizes large errors with respect to small errors because the errors are squared; that is, large errors are enlarged and small errors are reduced. So, this method must be used when the user is concerned about large errors but does not care about small errors.

2. The least-absolute-value (LAV) method. This method minimizes (see, for example, Arthanari and Dodge [6])

$$\sum_{i=1}^{n} |y_i - \mathbf{z}_i^t \boldsymbol{\beta}| \tag{12.38}$$

This method minimizes the sum of the distances between observed and predicted values instead of their squares. However, because of the presence of the absolute-value function, it is difficult to solve using standard regression techniques. This method treats all errors equally. Thus, this method must be used when the user is concerned about any level of error. In fact, what is important is the sum of all absolute errors, not a single error.

3. The minimax (MM) method. It minimizes

$$\max_i |y_i - \mathbf{z}_i^t \boldsymbol{\beta}|, \tag{12.39}$$

This method minimizes the maximum of the distances between observed and predicted values. So the user must be concerned only on the maximum error. However, because of the presence of the maximum function, it has the same difficulties as the previous method.

The estimates of $\boldsymbol{\beta}$ for the last two methods can be obtained by solving some simple linear programming problems. The LAV estimate of $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}$, can be obtained by solving the following linear programming problem (LPP) (see Castillo et al. [22]). Minimize

$$\sum_{i=1}^{n} \varepsilon_i \tag{12.40}$$

subject to

$$\begin{aligned} y_i - \mathbf{z}_i^t \boldsymbol{\beta} &\le \varepsilon_i, \qquad i = 1, \ldots, n \\ \mathbf{z}_i^t \boldsymbol{\beta} - y_i &\le \varepsilon_i, \qquad i = 1, \ldots, n \\ \varepsilon_i &\ge 0, \qquad i = 1, \ldots, n \end{aligned} \tag{12.41}$$

Similarly, the MM estimate of $\boldsymbol{\beta}$, $\tilde{\boldsymbol{\beta}}$, can be obtained by solving the following LPP. Minimize

$$\varepsilon \tag{12.42}$$

subject to

$$\begin{aligned} y_i - \mathbf{z}_i^t \boldsymbol{\beta} &\le \varepsilon, \qquad i = 1, \ldots, n \\ \mathbf{z}_i^t \boldsymbol{\beta} - y_i &\le \varepsilon, \qquad i = 1, \ldots, n \\ \varepsilon &\ge 0 \end{aligned} \tag{12.43}$$

Let $\hat{\boldsymbol{\beta}}_{(i)}$ and $\tilde{\boldsymbol{\beta}}_{(i)}$ be the LAV and MM estimators of $\boldsymbol{\beta}$ when the ith observation is omitted, respectively. The influence of the ith observation on the LAV estimators can be measured by

$$d_i(LAV) = ||\hat{\mathbf{y}} - \hat{\mathbf{y}}_{(i)}|| \tag{12.44}$$

where $\hat{\mathbf{y}} = \mathbf{Z}\hat{\boldsymbol{\beta}}$ is the vector of fitted value, $\hat{\mathbf{y}}_{(i)} = \mathbf{Z}\hat{\boldsymbol{\beta}}_{(i)}$ is the vector of fitted value computed when the i observation is omitted. Thus, $d_i(LAV)$ is the norm of the difference between the vectors of predicted values based on the full and reduced data, respectively.

Similarly, the influence of the ith observation on the MM estimators can be measured by

$$d_i(MM) = ||\tilde{\mathbf{y}} - \tilde{\mathbf{y}}_{(i)}|| \tag{12.45}$$

where $\tilde{\mathbf{y}} = \mathbf{Z}\tilde{\boldsymbol{\beta}}$ and $\tilde{\mathbf{y}}_{(i)} = \mathbf{Z}\tilde{\boldsymbol{\beta}}_{(i)}$ Finally, $d_i(LS)$ can be defined in a similar way.

To compute $d_i(LAV)$ or $d_i(MM)$, one needs to solve $n+1$ linear programming problems, unless one uses some sensitivity tools, that save computational time.

The following example illustrated the different sensitivities (robustness) of the three methods with respect to outliers.

Example 12.7 (Robustness with respect to outliers). Consider the data in Table 12.4, which has been simulated with the model

$$Y_i = 0.2 + 0.1Z_1 - 0.3Z_2 + 0.4Z_3 - 0.2Z_4 + \epsilon_i$$

where $\epsilon_i \sim U(0, 0.05)$ and the data points $7, 15$ and 16 have been modified to be converted in outliers by

$$\begin{array}{rcl} Y_7 & = & 0.2 + 0.1Z_1 - 0.3Z_2 + 0.4Z_3 - 0.2Z_4 - 0.05 \times 2 \\ Y_{15} & = & 0.2 + 0.1Z_1 - 0.3Z_2 + 0.4Z_3 - 0.2Z_4 + 0.05 \times 2 \\ Y_{16} & = & 0.2 + 0.1Z_1 - 0.3Z_2 + 0.4Z_3 - 0.2Z_4 - 0.05 \times 2 \end{array}$$

The three models – LAV, MM, and LS, in (12.37), (12.38), and (12.39), respectively – have been used to fit the data points. To this end, the following GAMS code was used:

```
$ title Regression

SETS
I number of data points /1*30/
P Number of parameters/1*5/
J number of regression models/1*3/
SS(I) subset of data points used in analysis;

PARAMETER
C(P) regression coefficients used in the simulation
/1 0.2
2 0.1
3 -0.3
4 0.4
5 -0.2/;

PARAMETER
Z(I,P) observed and predicted variables;

Z(I,P)=uniform(0,1);
Z(I,'1')=C('1')+sum(P$(ord(P)>1),C(P)*Z(I,P))+uniform(0,1)*0.05;
Z('7','1')=C('1')+sum(P$(ord(P)>1),C(P)*Z('7',P))-0.05*2;
Z('15','1')=C('1')+sum(P$(ord(P)>1),C(P)*Z('15',P))+0.05*2;
Z('16','1')=C('1')+sum(P$(ord(P)>1),C(P)*Z('16',P))-0.05*2;

PARAMETER
Y(I) Observed values
Y1(I) Fitted values with all data points
Y2(I) Fitted values with one data point removed
z0 auxiliary value;

Y(I)=Z(I,'1');
```

```
POSITIVE VARIABLES
EPSILON1(I) error associated with data I
EPSILON error;

VARIABLES
z1 objective function value
z2 objective function value
z3 objective function value
BETA(P);

EQUATIONS
dz1 objective function 1 value definition
dz2 objective function 2 value definition
dz3 objective function 3 value definition
lower1(I) lower bound of error
upper1(I) upper bound of error
lower2(I) lower bound of error
upper2(I) upper bound of error;

dz1..z1=e=sum(SS,EPSILON1(SS));
dz2..z2=e=EPSILON;
dz3..z3=e=sum(SS,sqr(Y(SS)-sum(P,Z(SS,P)*BETA(P))));
lower1(SS)..Y(SS)-sum(P,Z(SS,P)*BETA(P))=l=EPSILON1(SS);
upper1(SS)..-Y(SS)+sum(P,Z(SS,P)*BETA(P))=l=EPSILON1(SS);
lower2(SS)..Y(SS)-sum(P,Z(SS,P)*BETA(P))=l=EPSILON;
upper2(SS)..-Y(SS)+sum(P,Z(SS,P)*BETA(P))=l=EPSILON;

MODEL regreslav/dz1,lower1,upper1/;
MODEL regresMM/dz2,lower2,upper2/;
MODEL regresLS/dz3/;

file out/regress1.out/;
put out;

put "Data"/;

loop(I,
put I.tl:3;
loop(P,put " & ",Z(I,P):8:4;);
put "\\"/;
);
Z(I,'1')=1.0;

loop(J,
  SS(I)=yes;
  if(ord(J) eq 1,SOLVE regreslav USING lp MINIMIZING z1;Z0=z1.l);
  if(ord(J) eq 2,SOLVE regresMM USING lp MINIMIZING z2;Z0=z2.l);
  if(ord(J) eq 3,SOLVE regresLS USING nlp MINIMIZING z3;Z0=z3.l);
  put "All data points z0=",z0:12:8 ;
    loop(P,put " & ",BETA.l(P):10:5;);
    put "\\"/;
    Y1(I)=sum(P,Z(I,P)*BETA.l(P));
  loop(I,
    SS(I)=no;
    if(ord(J) eq 1,SOLVE regreslav USING lp MINIMIZING z1;Z0=z1.l);
    if(ord(J) eq 2,SOLVE regresMM USING lp MINIMIZING z2;Z0=z2.l);
```

Table 12.4: Simulated data points

i	Y_i	Z_{1i}	Z_{2i}	Z_{3i}	Z_{4i}
1	0.2180	0.8433	0.5504	0.3011	0.2922
2	-0.0908	0.3498	0.8563	0.0671	0.5002
3	0.2468	0.5787	0.9911	0.7623	0.1307
4	0.3431	0.1595	0.2501	0.6689	0.4354
5	0.1473	0.3514	0.1315	0.1501	0.5891
6	0.3076	0.2308	0.6657	0.7759	0.3037
7	**0.3981**	**0.5024**	**0.1602**	**0.8725**	**0.2651**
8	0.2089	0.5940	0.7227	0.6282	0.4638
9	0.0879	0.1177	0.3142	0.0466	0.3386
10	0.3795	0.6457	0.5607	0.7700	0.2978
11	0.2259	0.7558	0.6274	0.2839	0.0864
12	-0.0147	0.6413	0.5453	0.0315	0.7924
13	0.3731	0.1757	0.5256	0.7502	0.1781
14	0.1575	0.5851	0.6212	0.3894	0.3587
15	**0.5829**	**0.2464**	**0.1305**	**0.9334**	**0.3799**
16	**0.3781**	**0.3000**	**0.1255**	**0.7489**	**0.0692**
17	0.3163	0.0051	0.2696	0.4999	0.1513
18	0.0804	0.3306	0.3169	0.3221	0.9640
19	0.3913	0.3699	0.3729	0.7720	0.3967
20	-0.0961	0.1196	0.7355	0.0554	0.5763
21	0.1868	0.0060	0.4012	0.5199	0.6289
22	0.0703	0.3961	0.2760	0.1524	0.9363
23	0.2184	0.1347	0.3861	0.3746	0.2685
24	0.1059	0.1889	0.2975	0.0746	0.4013
25	0.1961	0.3839	0.3241	0.1921	0.1124
26	0.3889	0.5114	0.0451	0.7831	0.9457
27	0.2762	0.6073	0.3625	0.5941	0.6799
28	0.2523	0.1593	0.6569	0.5239	0.1244
29	0.1535	0.2281	0.6757	0.7768	0.9325
30	0.1480	0.2971	0.1972	0.2463	0.6465

```
      if(ord(J) eq 3,SOLVE regresLS USING nlp MINIMIZING z3;Z0=z3.l);
      put "Removing point ",I.tl:3," z0=",z0:12:8;
      loop(P,put " & ",BETA.l(P):10:5;);
      put "\\"/;
      Y2(I)=sum(P,Z(I,P)*BETA.l(P))-Y1(I);
      SS(I)=yes;
   );
   loop(I,put I.tl:3,Y2(I):10:6/;);
);
```

The resulting values of $d_i(LAV), d_i(MM)$, and $d_i(LS)$ are shown in Table 12.5, where the outliers and the largest three values of each column are boldfaced. Note that the LS method clearly identifies the three outliers, while the MM method has difficulties in identifying the data point 16, and the LAV method is unable to identify any of them. This shows that the LAV method is more robust, against outliers, than the MM method, and thus more robust than the LS method. It is interesting to point out the zero change in the $d_i(MM)$ when removing single data that are not extreme points. This is due to the fact that the regression line is defined by only two data extreme points (outliers or not).

Table 12.6 shows the parameter estimates for the three models, LAV, MM, and LS, estimated using all data points, and removing one of the first 5 points or the outliers. The sensitivities of each parameter with respect to each outlier can be observed. Again we can see that the LAV method is more robust to outliers (smaller changes in the parameter estimates when removing one outlier) than is the MM method.

■

12.5 Applications to Discretization of Continuous Optimization Problems

Optimization problems in the infinite-dimensional case, where functions replace vectors in cost functions, become finite-dimensional when one discretizes such situations to compute approximations to optimal functions. Although this is a subject beyond the scope of this book, we believe that it is interesting to point out how the common link between finite and infinite-dimensional optimization problems is carried out in a few selected and typical situations. Needless to say, these examples are elementary and academic so that it will be impossible to convey through them the richness of all situations and the difficulties attached to more complex, realistic instances. From this perspective, our aim is to examine such problems to indicate how discretizations can be carried out and, in this way, broaden and enlarge the set of optimization problems that can be, at least computationally, analyzed using the techniques and tools described in this text. In particular, we shall stress the use of a computational tool like GAMS to approximate and find optimal solutions for discretized versions of infinite-dimensional optimization problems.

There are two main categories of infinite-dimensional optimization: variational problems and optimal control problems. We shall describe and make computations for two selected, well-known examples in each category: the braquistochrone and the hanging cable, for the first case; and the optimal controls of a hitting target and of an harmonic oscillator, for the second. In all these examples, the point of view we have taken is related to a basic discretization scheme. More efficient and accurate approximations would require a more specialized treatment. For some more examples, see Polak [86] and Troutman [99].

Table 12.5: Values of $d_i(LAV), d_i(MM)$, and $d_i(LS)$, which allow analysis of the influence of outliers on the predictions

i	$d_i(LAV)$	$d_i(MM)$	$d_i(LS)$
1	**-0.017913**	0.000000	-0.008427
2	0.007130	0.000000	0.007199
3	0.003571	0.000000	0.006221
4	0.008645	0.000000	-0.000730
5	0.001230	0.000000	0.000278
6	-0.005584	0.000000	-0.001787
7	0.000400	**0.032893**	**0.021929**
8	0.000000	0.000000	0.001975
9	-0.001667	0.000000	-0.000668
10	-0.009012	0.000000	-0.003484
11	**-0.014035**	-0.017450	-0.010407
12	-0.011423	0.000000	-0.001843
13	-0.005477	0.000000	-0.004351
14	0.000000	0.000000	0.001429
15	-0.007005	**-0.056369**	**-0.019154**
16	0.002400	0.011714	**0.023611**
17	0.009702	0.000000	-0.003079
18	0.001470	0.017649	0.003613
19	-0.004419	0.000000	-0.001829
20	0.004550	0.000000	0.008311
21	-0.000332	0.000000	0.000371
22	-0.005133	0.000000	-0.003675
23	0.001806	0.000000	-0.000840
24	-0.002236	0.000000	-0.001722
25	0.000000	0.000000	0.001087
26	-0.005122	0.000000	-0.006582
27	0.001186	0.000000	-0.001038
28	-0.006072	0.000000	-0.004491
29	**0.014205**	**0.054685**	0.013893
30	0.000673	0.000000	0.000769

12.5.1 Variational Problems

Typically, a scalar, one-dimensional variational problem comes in the following form. Minimize

$$T(u) = \int_a^b F(t, u(t), u'(t))dt$$

subject to

$$u(a) = A, u(b) = B$$

where the unknown function u is assumed to be continuous.

Table 12.6: Parameter estimates for the three methods using all data and removing one of the first 6 data points or the outliers

i	Y_i	β_1	β_2	β_3	β_4
Parameters estimates for the LAV method					
All	0.23134	0.11741	-0.31868	0.41337	-0.22561
1	0.22128	0.09177	-0.30947	0.43343	-0.21652
2	0.22807	0.11690	-0.30427	0.40866	-0.22850
3	0.22944	0.11313	-0.31254	0.41648	-0.22957
4	0.23360	0.09454	-0.31811	0.42908	-0.22703
5	0.23074	0.12023	-0.31984	0.41182	-0.22353
6	0.23098	0.10934	-0.32361	0.40977	-0.21668
7	0.23108	0.11721	-0.31960	0.41392	-0.22401
15	0.22807	0.11690	-0.30427	0.40866	-0.22850
16	0.23291	0.11002	-0.31565	0.41743	-0.23107
Parameters estimates for the MM method					
All	0.21723	0.00519	-0.28273	0.37877	-0.09172
1	0.21723	0.00519	-0.28273	0.37877	-0.09172
2	- 0.21723	0.00519	-0.28273	0.37877	-0.09172
3	0.21723	0.00519	-0.28273	0.37877	-0.09172
4	0.21723	0.00519	-0.28273	0.37877	-0.09172
5	0.21723	0.00519	-0.28273	0.37877	-0.09172
6	0.21723	0.00519	-0.28273	0.37877	-0.09172
7	0.09010	0.13240	-0.27174	0.48689	-0.09162
15	0.14378	0.06891	-0.18892	0.38863	-0.14452
16	0.22157	-0.03975	-0.23504	0.40342	-0.14350
Parameters estimates for the LS method					
All	0.20098	0.08080	-0.26760	0.39807	-0.17825
1	0.20066	0.06789	-0.26633	0.40153	-0.17470
2	0.20061	0.07838	-0.25826	0.39316	-0.17677
3	0.19746	0.08240	-0.26064	0.40084	-0.17975
4	0.20037	0.08183	-0.26696	0.39742	-0.17827
5	0.20131	0.08090	-0.26801	0.39779	-0.17824
6	0.20173	0.08264	-0.26996	0.39625	-0.17818
7	0.20658	0.09872	-0.29242	0.41429	-0.18902
15	0.19585	0.08594	-0.24951	0.37834	-0.17624
16	0.22322	0.08249	-0.29511	0.40603	-0.20198

Integral constraints of the type

$$k = \int_a^b G(t, u(t), u'(t))dt$$

are also common. How is a discretized version built for the original problem? To this end, we divide the interval $[a, b]$ in $n+1$ equal subintervals, and assume that feasible functions for the new, discretized optimization problem are piecewise affine, that is, affine on each subinterval

$$[a + j\Delta, a + (j + 1)\Delta]$$

where

$$\Delta = \frac{(b - a)}{(n + 1)}$$

Notice that such class of functions are uniquely determined by its values at the nodes

$$a + j\Delta, \quad j = 1, \dots, n$$

and therefore, feasible vectors for the new optimization problem correspond to these values. We see that this process changes the original, infinite-dimensional problem to a finite-dimensional one. The point is that by letting $n + 1$, the number of subintervals becomes larger and larger, optimal solutions for these discretized optimization problems will resemble and approximate well enough, under conditions to be overlooked here, the true optimal solutions for the initial optimization problem. Let

$$x = (x_j), \ \ 1 \le j \le n. \ \ x_0 = A, \ \ x_{n+1} = B$$

be the nodal values of feasible functions. In this way the function u that we consider to optimize $T(u)$ is

$$u(t) = x_j + \frac{(x_{j+1} - x_j)}{\Delta}(t - a - j\Delta), \quad \text{if } t \in [a + j\Delta, a + (j + 1)\Delta] \quad (12.46)$$

This is the continuous, piecewise affine function that takes on values x_j at nodal points $a + j\Delta$. Thus, we get

$$T(u) = \sum_{j=0}^{n} \int_{a+j\Delta}^{a+(j+1)\Delta} F\left(t, u(t), \frac{(x_{j+1} - x_j)}{\Delta}\right) dt$$

where $u(t)$ in the interval of integration is given by (12.46). If we realize that u is determined by the vector $x = (x_1, \dots, x_n)$ and interpret $T(u)$ as a function of x, we are faced with a (nonlinear, unconstrained) programming problem. By solving it, we find an approximate solution to the initial, continuous optimization problem. The form of the functional $T(u)$ in terms of x depends on each particular situation. We will see and solve from this perspective the two examples mentioned above.

The Braquistochrone

This was a famous optimization problem in the twentieth century and one of the first examples where optimal solutions were explicitly found. Assume that we are given two points, A and B, in a plane at different heights. We are asked to find the path joining those two points so that a unit mass, under the action of gravity, and without slipping, takes the least time possible in getting from the highest to the lowest point. This is one the most celebrated examples of the so-called minimum transit problems.

The first thing to do is to derive the functional measuring the time spent by the mass, in going from A to B, under the action of gravity. For convenience, we put the X axis along the vertical direction so that gravity acts along this axis, and the Y axis horizontally. Let, without loss of generality, $A = (1, 0)$, $B = (0, 1)$. Assume that $y = u(x)$ is a continuous curve joining A and B, so that $u(0) = 1$ and $u(1) = 0$. If the unit mass is supposed to travel from A to B through the path determined by the graph of u, what would the time spent in going from A to B be? We know that space is velocity times time. In our continuous situation space is measured by the elementary element's length

$$d\ell = \sqrt{1 + u'(x)^2}\, dx$$

while velocity, in terms of g, is given by

$$v = \sqrt{2gx}$$

Therefore, if $T(u)$ stands for the time spent associated with the path $y = u(x)$, we have

$$T(u) = \int_0^1 \frac{\sqrt{1 + u'(x)^2}}{\sqrt{2gx}}\, dx$$

or equivalently, keeping in mind that positive multiplicative constants do not affect the solution of optimization problems, we obtain

$$T(u) = \int_0^1 \frac{\sqrt{1 + u'(x)^2}}{\sqrt{x}}\, dx$$

This is the functional to be minimized with respect to all those continuous $y = u(x)$ functions satisfying the constraints $u(0) = 1$ and $u(1) = 0$.

To fully understand this situation, one has to study the convexity properties of the integrand for T and the associated Euler–Lagrange equation. Since this would take us too far, we shall restrict our initial optimization problem to a discretized version of it, and stress the use of GAMS to compute optimal solutions for the discretized version.

The resulting discretized objective function $\bar{T}(x)$, in terms of the nodal points, becomes

$$\bar{T}(x) = \sum_{j=0}^{n} \sqrt{1 + \left(\frac{x_{j+1} - x_j}{\Delta}\right)^2} \int_{j/(n+1)}^{(j+1)/(n+1)} \frac{dx}{\sqrt{x}}$$

or even more explicitly, neglecting positive constants

$$\bar{T}(x) = \sum_{j=0}^{n} \left(\sqrt{\frac{j+1}{n+1}} - \sqrt{\frac{j}{n+1}}\right) \sqrt{1 + \left(\frac{x_{j+1} - x_j}{\Delta}\right)^2}$$

keeping in mind that $x_0 = 1$ and $x_{n+1} = 0$. This is the function we want to minimize with the help of GAMS, for several values of the number of subintervals n. By doing so, we find a very good agreement with the arc of a cycloide, which is the optimal solution of the continuous optimization problem.

The GAMS code for this problem, together with a summary of the optimal solution, follows.

```
$title FUNCTIONAL1 n=20
SET J /0*20/;
VARIABLES z,x(J);
  x.fx(J)$(ord(J) eq 1) = 0;
  x.fx(J)$(ord(J) eq card(J)) = 1;
SCALAR n;
  n = card(J)-2;
EQUATION
  cost   objective function;
cost.. z =e= SUM(J$(ord(J) lt card(J)),
(sqrt(ord(J))-sqrt(ord(J-1))*sqrt(1+sqr(n+1)*sqr(x(J+1)-x(J))));
MODEL funct1 /all/;
SOLVE funct1 USING nlp MINIMIZING z;
```

```
                      LOWER     LEVEL     UPPER    MARGINAL
---- VAR Z             -INF      5.776     +INF       .
---- VAR X
      LOWER     LEVEL     UPPER    MARGINAL
0       .         .         .       -2.091
1      -INF      0.005     +INF  -1.163E-6
2      -INF      0.018     +INF  1.6563E-6
3      -INF      0.036     +INF  8.1950E-7
4      -INF      0.057     +INF  -1.405E-6
5      -INF      0.082     +INF  -4.992E-7
6      -INF      0.110     +INF  -1.353E-6
7      -INF      0.141     +INF  -3.233E-6
8      -INF      0.176     +INF  3.1623E-7
9      -INF      0.215     +INF  3.0808E-6
10     -INF      0.257     +INF  1.8503E-6
11     -INF      0.303     +INF  -6.250E-7
12     -INF      0.353     +INF  -1.116E-6
13     -INF      0.408     +INF  3.2812E-6
14     -INF      0.468     +INF  -1.583E-6
15     -INF      0.534     +INF       EPS
16     -INF      0.606     +INF  7.5934E-7
17     -INF      0.687     +INF  -8.374E-7
18     -INF      0.777     +INF       EPS
19     -INF      0.880     +INF       EPS
20     1.000     1.000     1.000     2.091
```

If we plot in the same graph these computed values with the ones corresponding to the exact solution, we are unable to distinguish them apart (see Figure 12.5).

Alternatively, we can set up the discretization scheme considering the slopes on each subinterval as independent variables. This leads to a simpler form of the objective function, but we would have to enforce a (linear) constraint because

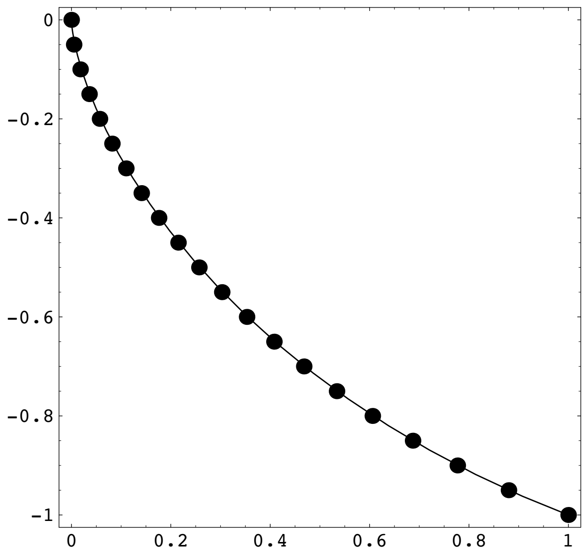

Figure 12.5: The exact (continuous) and the approximated (dotted) solutions to the braquistochrone problem.

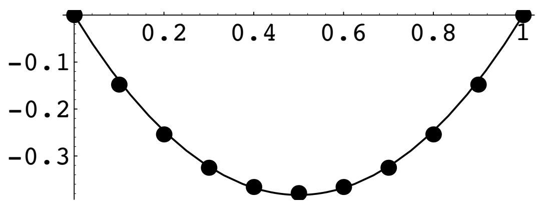

Figure 12.6: The exact (continuous) and the approximated (dotted) solutions to the hanging cable.

the slopes in the different subintervals must be such that the value of u at 1 is given, and this imposes a constraint on the sets of possible slopes. Rather than solving the same example in this format with an integral constraint, we prefer to get into the next example.

The Hanging Cable

The hanging cable is also one of the classical problems Euler solved in the twentieth century by variational techniques. It consists of determining the shape adopted by a cable, hanging freely on its two endpoints at the same height, under the action of its own weight (see Figure 12.6).

We postulate that the optimal shape will be that corresponding to the least potential energy, and assume that the cable has constant cross sections along its length. Let us say that the two endpoints of the cable are a distance H apart, and in the same horizontal plane, and that the length of the cable is L. Obviously, we must require $L > H$ so that the problem is well posed. Assume that we place the X axis along the two endpoints of the cable, and the Y axis vertically starting at the left endpoint of the cable. In this way $(0, 0)$ and $(H, 0)$ are the coordinates of the two, fixed endpoints. Let $y = u(x)$, $x \in (0, H)$ be any continuous function joining the two endpoints. If the cable would adopt the profile described by the graph of u, its potential energy is measured by the integral

$$k \int_0^H u(x)\sqrt{1 + u'(x)^2}\,dx$$

where $k > 0$ is a constant related to the material the cable is made of, $u(x)$ is the height of each material cross-section, and

$$\sqrt{1 + u'(x)^2}\,dx$$

is the element's length. Therefore, and discarding again positive constants, we seek the optimal profile $u(x)$ so that it minimizes the potential energy functional

$$P(u) = \int_0^H u(x)\sqrt{1 + u'(x)^2}\,dx$$

subject to $u(0) = 0$, $u(H) = 0$. This time, however, we need an extra constraint since we must enforce the cable to have total length L. Otherwise the optimization problem would not make much sense because we could lower the potential energy all the way to $-\infty$ by letting the length become larger and larger. Hence, we must enforce the constraint

$$\int_0^H \sqrt{1 + u'(x)^2}\,dx = L$$

Then, our optimization problem becomes minimization of

$$P(u) = \int_0^H u(x)\sqrt{1 + u'(x)^2}\,dx$$

subject to

$$\int_0^H \sqrt{1 + u'(x)^2}\,dx = L \quad u(0) = 0, \;\; u(H) = 0$$

Again the complete analysis of this problem would compel us to use the Lagrange multipliers associated with this integral constraint, and study the Euler–Lagrange equation for the augmented Lagrangian. Alternatively, we set up a discretized version of this problem, and solve it using GAMS, as before.

The process is exactly the same as above. We restrict our optimization problem to the finite-dimensional subspace of piecewise affine functions determined by the nodal values x_j, $j = 1, 2, \ldots, n$ over a uniform partition of the interval

$(0, H)$. Without loss of generality, we may take $H = 1$. As we have done before, we have

$$P(x) = \sum_{j=0}^{n} \int_{j/(n+1)}^{(j+1)/(n+1)} u(x)\sqrt{1 + \left(\frac{x_{j+1} - x_j}{\Delta}\right)^2}\, dx$$

where again $\Delta = 1/(n+1)$. Since the square roots appearing under the integral sign are constants, we can also write

$$P(x) = \sum_{j=0}^{n} \sqrt{1 + \left(\frac{x_{j+1} - x_j}{\Delta}\right)^2} \int_{j/(n+1)}^{(j+1)/(n+1)} u(x)\, dx$$

In addition we can use the trapezoidal rule to obtain a good approximation of the integrals

$$\int_{j/(n+1)}^{(j+1)/(n+1)} u(x)\, dx \simeq \frac{x_{j+1} + x_j}{2(n+1)}$$

Altogether we obtain the objective functional

$$P(x) = \sum_{j=0}^{n} \sqrt{1 + \left(\frac{x_{j+1} - x_j}{\Delta}\right)^2}\frac{x_{j+1} + x_j}{2(n+1)}$$

Finally, expressing the initial integral constraint in terms of the vector x, we get

$$L = \frac{1}{n+1} \sum_{j=0}^{n} \sqrt{1 + \left(\frac{x_{j+1} - x_j}{\Delta}\right)^2}$$

We must remember that $x_0 = 0$ and $x_{n+1} = H$. This discretized, finite-dimensional formulation is appropriate for computation using GAMS. With this package we can find good approximations to the classical catenary, which is the optimal solution of the continuous problem.

The GAMS code to approximate the optimal solution, together with part of the solution file, can be found below. L is taken to be 1.3116.

```
$title FUNCTIONAL2 n=10
SET J /0*10/;
VARIABLES z,x(J);
  x.fx(J)$(ord(J) eq 1) = 0;
  x.fx(J)$(ord(J) eq card(J)) = 0;
  x.l(J)=-0.5;
SCALAR n;
  n = card(J)-2;
EQUATION
  cost   objective function
  rest   equality constraint;
cost.. z =e= SUM(J$(ord(J) lt card(J)),
                  sqrt(1+sqr(n+1)*sqr(x(J+1)-x(J)))*(x(J+1)+x(J)));
rest.. (n+1)*1.3116 =e= SUM(J$(ord(J) lt card(J)),sqrt(1+sqr(n+1)*sqr(x(J+1)-x(J))));
MODEL funct2 /all/;
SOLVE funct2 USING nlp MINIMIZING z;
```

```
                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR Z              -INF     -6.114     +INF       .
---- VAR X
     LOWER     LEVEL     UPPER    MARGINAL
0      .         .         .        13.116
1     -INF    -0.148     +INF        EPS
2     -INF    -0.254     +INF  6.1259E-7
3     -INF    -0.325     +INF  8.3847E-7
4     -INF    -0.366     +INF  6.0564E-7
5     -INF    -0.379     +INF       .
6     -INF    -0.366     +INF  1.2127E-6
7     -INF    -0.325     +INF  1.0593E-6
8     -INF    -0.254     +INF  -5.058E-7
9     -INF    -0.148     +INF  8.8932E-7
10     .         .         .        13.116
```

Figure 12.6 shows a graphic of the exact and approximated solutions.

12.5.2 Optimal Control Problems

Optimal control problems are more complex continuous optimization problems than its variational counterpart. Indeed, variational problems are a very special class of optimal control problems. The main ingredients of an optimal control problem are

1. A vector $x(t)$ determining the dynamics of the state of a certain system under control; the number of components in x indicates the number of parameters to be uniquely determined to identify the state of the system under consideration.

2. A vector $u(t)$ designating the control that we can exercise on the system so as to modify its dynamics with some specific objective in mind; frequently, the control is restricted by requiring $u(t) \in K$ for a given set K.

3. The state equation

$$x'(t) = f(t, x(t), u(t)), \quad t \in (0, T)$$

which governs the dynamics of the system, and expresses the interaction between states and controls.

4. Additional constraints on the initial and/or final state of the system.

5. The objective functional

$$I(u) = \int_0^T g(t, x(t), u(t))\, dt$$

yielding a measure of optimality when the control u is exercised on the system, and the resulting dynamics, $x(t)$, is obtained by solving the state equation together with initial and/or final states, as indicated by $x(t)$.

The aim of the optimal control problem is to find the best way of acting on the system, the optimal control $u(t)$, measured in terms of the proposed cost functional $I(u)$. The relationship between the state x and the control u, through the state equation and additional constraints, is what makes optimal control problems much more complex than variational problems. Notice that variational problems correspond to the simplest, nontrivial state equation

$$x'(t) = u(t),$$

where both the initial and final states are prescribed a priori.

The computational procedure is, likewise, much more involved because it requires one to solve (often numerically by an appropriate solver) the state equation. We do not pretend to enter a full discussion of this issue. Our goal here is to provide two easy examples of optimal control problems to stress the usefulness of a tool such as GAMS, possibly in conjunction with some additional package to take care of the state equation, in approximating optimal controls.

Optimal Control of Hitting a Target

Assume that we would like to hit a target that is a distance $3 + \frac{5}{6}$ length units apart from us in 3 units of time. Then, the control we can exercise on the projectile is the magnitude of acceleration $u(t)$, so that the state of the projectile $(x(t), x'(t))$ must obey the state equation and auxiliary conditions

$$x''(t) = u(t), \quad x(0) = x'(0) = 0, \quad x(3) = 3 + \frac{5}{6}$$

The initial conditions reflect the fact that the projectile departs from rest. We must also respect a constraint in the size of $u(t)$ by imposing $0 \le u \le 1$. Finally, the objective is to hit the target in the cheapest possible way measured by the cost functional

$$I(u) = k \int_0^3 u(t)^2 \, dt$$

where $k > 0$ is a constant.

This is a typical optimal control problem. The peculiar form of the data allows us to obtain the optimal solution analytically, by exploiting optimality conditions through Pontryagin's maximum principle (see, for example, Polak [86] or Troutman [99]). Indeed the (unique) optimal control for this problem is

$$u(t) = \begin{cases} 1, & t \le 1 \\ \dfrac{3-t}{2}, & t \ge 1 \end{cases}$$

A justification of this is far beyond the scope of this book. However, we can compare the exact solution with an approximated one, computed using GAMS.

To set up a discretized version of the optimal control is, as announced above, more involved because it requires to solve the state equation. In our particular example this can be done explicitly so that we will end up with a precise optimization problem suitable for GAMS. The underlying idea behind the discretization is, however, the same: we divide the time interval $(0, 3)$ in n equal subintervals, and we set the control u to be u_j, constant, in the associated subinterval $(3(j-1)/n, 3j/n)$. By doing so, and after solving the state equation for this class of piecewise constant controls, we hope to express the discretized, optimal control problem as a (nonlinearly constrained) mathematical programming problem in the u_j variables.

In the subinterval $(3(j-1)/n, 3j/n)$ the control u takes on the constant value u_j; therefore assuming, recursively, that we have solved the state equation in the previous subinterval $(3(j-2)/n, 3(j-1)/n)$ and letting a_{j-1} and b_{j-1} be the values of the velocity and position, respectively, for $t = 3(j-1)/n$, we need to find the solution of

$$x''(t) = u_j, \quad x'\left(\frac{3(j-1)}{n}\right) = a_{j-1}, \; x\left(\frac{3(j-1)}{n}\right) = b_{j-1}$$

to get

$$x(t) = \frac{u_j}{2}\left(t - \frac{3(j-1)}{n}\right)^2 + a_{j-1}\left(t - \frac{3(j-1)}{n}\right) + b_{j-1}$$

The values of the velocity and position at $t = 3j/n$ are

$$a_j = a_{j-1} + \frac{3u_j}{2n}, \quad b_j = b_{j-1} + a_{j-1}\frac{3}{n} + \frac{9u_j}{2n^2}$$

It should be noted that the initial conditions imply $a_0 = b_0 = 0$. By using this recursion formulas repeatedly, it is not hard to find

$$a_j = \frac{3}{n}\sum_{k=1}^{j} u_k, \quad b_j = \frac{9}{2n^2}\sum_{k=1}^{j}(2j-2k+1)u_k$$

The final target condition $x(3) = 3 + \frac{5}{6}$ will translate into the linear constraint

$$\frac{9}{2n^2}\sum_{k=1}^{n}(2n-2k+1)u_k = 3 + \frac{5}{6}$$

On the other hand, the cost quadratic functional, neglecting positive constants, becomes

$$I(u) = \sum_{k=1}^{n} u_k^2, \quad u = (u_k); \;\; k = 1, 2, \ldots, n$$

Thus, we must solve the following (quadratic) mathematical programming problem. Minimize

$$Z = \sum_{k=1}^{n} u_k^2$$

subject to

$$\frac{9}{2n^2}\sum_{k=1}^{n}(2n-2k+1)u_k = 3 + \frac{5}{6}, \quad 0 \le u_k \le 1$$

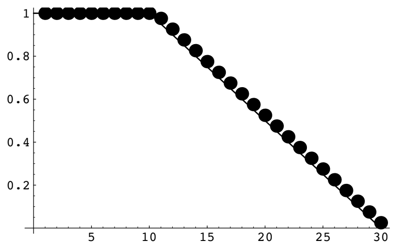

Figure 12.7: The exact (continuous) and the approximated (dotted) solution to the optimal control of a hitting target problem.

In this format, GAMS can find an approximation to the original optimal control problem, which is in close agreement with the exact solution given previously. Figure 12.7 shows the exact and the numerical solutions of this problem. The GAMS code and computed solutions follows.

```
$title FUNCTIONAL3 n=30
SET K /1*30/;
VARIABLES z,u(K);
  u.up(K) = 1;
  u.lo(K) = 0;
SCALAR n;
  n = card(K);
EQUATION
  cost   objective function
  rest   equality constraint;
cost.. z =e= SUM(K,sqr(u(K)));
rest.. 3+(5/6) =e= (9/(2*sqr(n)))*SUM(K,(2*n-2*ord(K)+1)*u(K));
MODEL funct3 /all/;
SOLVE funct3 USING nlp MINIMIZING z;
```

```
                        LOWER     LEVEL     UPPER    MARGINAL
---- VAR Z               -INF     16.671     +INF        .
---- VAR U
      LOWER     LEVEL     UPPER    MARGINAL
1       .       1.000     1.000    -0.952
2       .       1.000     1.000    -0.852
3       .       1.000     1.000    -0.752
4       .       1.000     1.000    -0.652
5       .       1.000     1.000    -0.552
6       .       1.000     1.000    -0.452
7       .       1.000     1.000    -0.351
8       .       1.000     1.000    -0.251
9       .       1.000     1.000    -0.151
10      .       1.000     1.000    -0.051
11      .       0.976     1.000      EPS
12      .       0.926     1.000 3.4120E-6
13      .       0.876     1.000 2.7012E-6
14      .       0.826     1.000 1.9903E-6
15      .       0.775     1.000 1.2794E-6
16      .       0.725     1.000 -2.347E-7
17      .       0.675     1.000 -2.185E-7
18      .       0.625     1.000      .
```

```
19       .        0.575     1.000 -1.564E-6
20       .        0.525     1.000      EPS
21       .        0.475     1.000      EPS
22       .        0.425     1.000      EPS
23       .        0.375     1.000      EPS
24       .        0.325     1.000      EPS
25       .        0.275     1.000 -1.026E-5
26       .        0.225     1.000 -8.395E-6
27       .        0.175     1.000 -6.530E-6
28       .        0.125     1.000 -4.664E-6
29       .        0.075     1.000 -2.798E-6
30       .        0.025     1.000      EPS
```

Optimal Control of an Harmonic Oscillator

There is a special class of important optimal control problems where the objective is to perform a known task in minimum time. In these situations the governing state equation

$$x'(t) = f(t, x(t), u(t)), \quad t \in (0, T), u(t) \in K$$

is completed with both initial and final states x_I and x_F, respectively. The task is to find the control u that accomplishes the final, given task

$$x(T) = x_F$$

for the smallest possible value of T.

To fix our ideas, we treat below the example of a linear harmonic oscillator with state equation

$$x''(t) + x(t) = u(t), \quad t \in (0, T), |u| \leq 1$$

The goal is to lead the oscillator from given, initial conditions

$$x(0) = a_0, \;\; x'(0) = b_0$$

to rest

$$x(T) = x'(T) = 0$$

in minimum time. Again this example can be treated analytically by exploiting the optimality conditions coming from the Pontryaguin's maximum principle. In particular, the linear (in fact, constant) dependence of the cost functional on the control implies that the optimal control will take exclusively the extremal values $+1$ and -1. But let us pretend not to have that information at our disposal, and proceed to formulate a discretized version of the optimization problem suitable for GAMS.

Since this time T is not known, we must incorporate it as one of our independent variables for the resulting mathematical programming problem, so that $u = (u_j)_{j=0,1,\ldots,n}$ is the unknown, where u_0 stands for T and the associated, piecewise constant control takes the constant value u_j on the interval

$(u_0(j-1)/n, u_0j/n)$. As before, if we let

$$a_j = x\left(\frac{u_0 j}{n}\right), \quad b_j = x'\left(\frac{u_0 j}{n}\right)$$

then we can find x by recursively solving

$$x''(t) + x(t) = u_j, \quad x\left(\frac{u_0(j-1)}{n}\right) = a_{j-1}, x'\left(\frac{u_0(j-1)}{n}\right) = b_{j-1}$$

For this example, it is much more complicated to find exact formulas for a_j and b_j. Even though we could use some symbolic computation package to find such expressions, there is no point in doing so if, after all, what we are about to compute is an approximation to the exact optimal control. Therefore it suffices to use, for instance, an Euler integrator of step size precisely u_0/n to find reasonable approximations for a_j and b_j. If we do not distinguish between the exact and approximated values for a_j and b_j, it is easy to find, in matrix notation

$$\begin{pmatrix} a_j \\ b_j \end{pmatrix} = \begin{pmatrix} a_{j-1} \\ b_{j-1} \end{pmatrix} + \frac{u_0}{n}\left[\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} a_{j-1} \\ b_{j-1} \end{pmatrix} + \begin{pmatrix} 0 \\ u_j \end{pmatrix}\right]$$

for $j = 1, 2, \ldots, n$. By using this identity recursively, we obtain

$$\begin{pmatrix} a_j \\ b_j \end{pmatrix} = \begin{pmatrix} 1 & \frac{u_0}{n} \\ -\frac{u_0}{n} & 1 \end{pmatrix}^j \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} + \frac{u_0}{n}\sum_{k=0}^{j-1}\begin{pmatrix} 1 & \frac{u_0}{n} \\ -\frac{u_0}{n} & 1 \end{pmatrix}^k \begin{pmatrix} 0 \\ u_{j-k} \end{pmatrix}$$

The constraints come from demanding the desired final rest conditions $a_n = b_n = 0$. Thus, the discrete (nonlinearly constrained) mathematical programming problem whose optimal solution provides a good approximation for our problem involves minimization of

$$Z = u_0$$

subject to

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & \frac{u_0}{n} \\ -\frac{u_0}{n} & 1 \end{pmatrix}^n \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} + \frac{u_0}{n}\sum_{j=1}^{n}\begin{pmatrix} 1 & \frac{u_0}{n} \\ -\frac{u_0}{n} & 1 \end{pmatrix}^{n-j} \begin{pmatrix} 0 \\ u_j \end{pmatrix}$$

and

$$u_0 \geq 0, \quad -1 \leq u_j \leq 1, j = 1, 2, \ldots, n$$

For specific values of the initial conditions a_0 and b_0, and a value for n, GAMS can find good approximations of the optimal control of a linear harmonic oscillator.

With the objective of simplifying the GAMS formulation of this situation, we have used the change of variables

$$\frac{u_0}{n} = \tan\alpha, \quad 0 \leq \alpha \leq \frac{\pi}{2}$$

so that

$$\begin{pmatrix} 1 & \frac{u_0}{n} \\ -\frac{u_0}{n} & 1 \end{pmatrix}^k = (1+\tan^2\alpha)^k \begin{pmatrix} \cos(k\alpha) & \sin(k\alpha) \\ -\sin(k\alpha) & \cos(k\alpha) \end{pmatrix}$$

We have chosen several possibilities for initial conditions (a_0, b_0), namely, $(-1, 0)$, $(3, 0)$, $(-7, 0)$. What we see in the computations below, except for some inaccuracies, is the typical bang–bang control jumping from -1 to 1, as predicted by Pontryaguin maximum principle.

A GAMS input file as well as a part of the corresponding output file for the case $(a_0, b_0) = (3, 0)$ is written below:

```
$title FUNCTIONAL n=20, azero
SET J /1*20/;
SCALAR pi the pi number /3.1416/;
VARIABLES z,theta,u(J);
 u.lo(J)=-1;
 u.up(J)=1;
 theta.lo=0;
 theta.up=pi/2;
 theta.l=pi/4;
 u.l(J)=0.;
SCALARS n, azero;
  n = card(J);
  alpha=-7;
EQUATION
  cost   objective function
  const1
  const2;
cost.. z =e= SIN(theta)/COS(theta);;
const1..
alpha*COS(n*theta)*POWER((1+POWER((SIN(theta)/COS(theta)),2)),n)+(SIN(theta)
/COS(theta))*
         SUM(J,u(J)*SIN((n-ORD(J))*theta)*
           POWER((1+POWER((SIN(theta)/COS(theta)),2)),n-ORD(J))
         ) =E= 0;
const2..
-azero*SIN(n*theta)*POWER((1+POWER((SIN(theta)/COS(theta)),2)),n)+(SIN(theta
)/COS(theta))*
         SUM(J,u(J)*COS((n-ORD(J))*theta)*
         POWER((1+POWER((SIN(theta)/COS(theta)),2)),n-ORD(J))
         ) =E= 0;
MODEL funct5 /all/;
SOLVE funct5 USING nlp MINIMIZING z;

  azero=3;
                      LOWER     LEVEL     UPPER    MARGINAL
---- VAR Z             -INF      0.982     +INF       .
---- VAR THETA           .       0.776     1.571      .
---- VAR U
      LOWER     LEVEL     UPPER    MARGINAL
1     -1.000    -1.000     1.000 1.2962E+5
2     -1.000     1.000     1.000 -2.455E+5
3     -1.000     1.000     1.000 -2.118E+5
4     -1.000     1.000     1.000 -9.025E+4
5     -1.000     1.000     1.000 -1.067E+4
6     -1.000    -1.000     1.000 15633.593
```

```
7     -1.000    -1.000     1.000 14116.404
8     -1.000    -1.000     1.000  6201.667
9     1.000    -1.000     1.000   846.919
10    -1.000     1.000     1.000  -991.476
11    -1.000     1.000     1.000  -939.412
12    -1.000     1.000     1.000  -425.373
13    -1.000     1.000     1.000   -65.560
14    -1.000    -1.000     1.000    62.581
15    -1.000    -1.000     1.000    62.430
16    -1.000    -1.000     1.000    29.126
17    -1.000    -1.000     1.000     4.979
18    -1.000     1.000     1.000    -3.929
19    -1.000     1.000     1.000    -4.143
20    -1.000     1.000     1.000    -1.991
```

12.6 Transportation Systems

In this section we deal with some network equilibrium models that are applied to transportation problems. This section has two main objectives: (1) to show some models that are routinely used in transportation planning, and (2) to illustrate how GAMS is an appropriate tool to implement these models.

We avoid the discussion of specific algorithms to solve these kinds of problems efficiently. The interested reader may consult the comprehensive reviews on this topic given in Patriksson [84].

Readers interested in mathematical models used in transportation planning are directed to the books of Ortúzar and Willumsen [82], and Sheffi [98]. A good introduction is provided in the book of Potts and Oliver [88].

12.6.1 Introduction

Traffic planning and transportation problems have motivated a large amount of mathematical models. The use of traffic planning models aids planners in predicting what effects on network performance are produced by changes in the network topology, or in their parameters. The classical transportation model the following stages:

Base Inventory. In this stage the study area is defined, and an inventory of the main transportation networks and travel patterns is obtained.

Model Analysis. The second phase of the process is the selection and calibration of the model. It has four steps:

1. *Trip generation step.* This step starts by considering a zoning and network system, and a database of each zone. These data, which include information about economic activity, social distribution, educational and recreational facilities, and shopping space are used to estimate the total number of trips generated and attracted by each zone of the study area.

2. *Distribution step.* This stage is the allocation of these trips to particular destinations, such that their distribution over space, and building an origin–destination (O–D) trip matrix.

3. *Modal split step.* The modal split step produces the allocation of trips to different transportation modes. In this phase the origin–destination matrices are obtained for every transportation mode (public, private, etc.). Their elements are the total number of trips associated with a transportation mode for each origin–destination pair Ω.

4. *Assignment step.* Finally, the last stage requires the *assignment* of these trips to the transportation network. This section deals with some assignment equilibrium models to the road network. These models predict the utilization level of the different arcs in the network. Thus, it can be used to answer questions such as what would happen in the network service level if a new road were built or the capacity of an existing road were modified.

Travel Forecasts. In this stage, the service level and demand of the transportation network is forecast using the corresponding mathematical models for different scenarios.

Network Evaluation. In the final phase of the process, alternative future transportation systems are evaluated and the optimal one is selected.

In this section we present four assignment models for private vehicles. These models take into account the congestion effect, and use Wardrop's principle [102] as a general framework to formulate them. Wardrop's first principle states that under congested conditions drivers choose routes until no one can reduce its costs by switching to other path.

12.6.2 Elements of a Road Transportation Network

In this subsection we give a brief outline of the main elements of the road transportation network theory. The mathematical model used to represent a road transportation network is called a *directed graph.* It is defined as a pair $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is a finite set of nodes, and $\mathcal{A}$ is a set of ordered pairs (*arcs* or *links*) of elements of $\mathcal{N}$. The nodes are denoted by i or j, and the arcs or links, or more precisely the *directed links*, are denoted as (i, j). The link directions are important because they allow distinction between one-way routes and two-way routes.

Example 12.8 (Nguyen–Dupuis Network). Consider the graph $\mathcal{G}$, in Table 12.7, where $\mathcal{N} = \{1, \ldots, 13\}$ and $\mathcal{A}$ has 19 links. This example is taken from Nguyen and Dupuis [80] (ND network) and is illustrated in Figure 12.8.

■

Table 12.7: Links of the graph $\mathcal{G}$

Links			
(1, 5)	(1, 12)	(4, 5)	(4, 9)
(5, 6)	(5, 9)	(6, 7)	(6, 10)
(7, 8)	(7, 11)	(8, 2)	(9, 10)
(9, 13)	(10, 11)	(11, 2)	(11, 3)
(12, 6)	(12, 8)	(13, 3)	

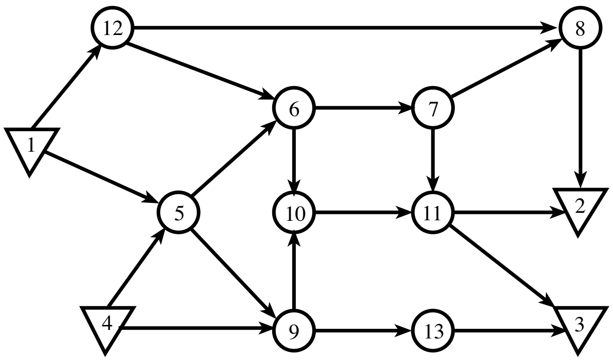

Figure 12.8: Transportation network represented by a graph. Circles refer to intermediate nodes, and triangles are the centroids (origins and destinations).

Single-Commodity Problem

In many problems, flows of vehicles, goods, or passengers can be associated with links of a graph. In this case we denominate the graph a *network* or, more specifically, a *transportation network*, if the application to transportation is to be emphasized. The term *flow* denotes quantity per unit time, such as vehicles per hour or pedestrians per minute.

Fundamental to a network, such as electrical networks, water pipe networks, or transportation networks, is the *flow conservation law*, which states that the flows are neither created nor destroyed.

The nodes of $\mathcal{N}$ are classified as *centroids* or *intermediate nodes.* The first group represents either zones where vehicle trips are *produced* by residents going on a trip elsewhere, and are called *origins*, or zones where trips are *attracted* to places of work, shopping, and so on, and are called *destinations.*

The conservation flow law states that the sum of all flows leaving a node minus the sum of the flows entering that node equals the flow generated or attracted to that node.

The main elements of this problem are

1. **Data.**

$\mathcal{N}$**:** the set of nodes in the network

$\mathcal{A}$**:** the set of arcs in the network

r_i**:** the flow produced or attracted by node i. Note that if i is an intermediate node then $r_i = 0$. If $r_i > 0$ then i is an origin and if $r_i < 0$, i is a destination

$A(i)$**:** the sets of nodes $\{j : \ j \in \mathcal{N}, (i,j) \in \mathcal{A}\}$ "after" node i

$B(i)$**:** the sets of nodes $\{j : \ j \in \mathcal{N}, (j,i) \in \mathcal{A}\}$ "before" node i

$c_{ij}(x)$**:** cost of the transportation associated with link $i - j$ and flow x

2. **Variables.**

   f_{ij}**:** the *link flow* on the directed link (i,j)

3. **Constraints.** The conservation equations are written in the following form:

$$\sum_{j\in A(i)} f_{ij} - \sum_{j\in B(i)} f_{ji} = r_i, \ \forall i \in \mathcal{N}$$

4. **Function to be optimized.** In this problem, we minimize

$$Z = \sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} c_{ij}(x)dx \tag{12.47}$$

Example 12.9 (Single-commodity problem). To illustrate the flow conservation equations, consider the network given in Figure 12.8, where circles refer to intermediate nodes, and triangles are the centroids (origins and destinations). For the intermediate node 5, we have

$$i = 5, \quad A(5) = \{6, 9\}, \quad B(5) = \{1, 4\}$$

$$\sum_{j\in A(5)} f_{5j} - \sum_{j\in B(5)} f_{j5} = f_{5,6} + f_{5,9} - f_{1,5} - f_{4,5} = r_5 = 0$$

■

Multicommodity Flow Problems

The previous discussion was based on the assumption that only one type of flow exists on the network, thus, we are in front of a *single-commodity* network.

Multicommodity flow problems arise when several commodities use the same underlying network. The commodities might be differentiated either by their physical characteristics and/or because they have different origins and/or destinations. For example, in a communication network where video and audio flows have to be considered, or in a road network, where trips are classified according to origins and destinations. In these cases, it is essential to distinguish certain O–D flows from others, to make sure travelers get their correct destinations. This imposes that each *commodity* defined by an O–D demand pair must satisfy its flow conservation equation. We denote by $\Omega = (i, j)$ an specific origin–destination (O–D) pair, and by W the set of all the O–D pairs. In traffic studies the traffic on the network is studied as a superposition of traffic between specific O–D pairs.

To formulate the multicommodity network, we consider that the flow of the commodity Ω from O$^\Omega$ -D$^\Omega$ is $g^\Omega > 0$. Table 12.8 defines a set of O–D pairs for the ND network of Figure 12.8.

Table 12.8: O–D pairs for the example network

Pair	Demand g^Ω	Pair	Demand g^Ω
$\Omega_1 = (1, 2)$	400	$\Omega_2 = (1, 3)$	800
$\Omega_3 = (4, 2)$	600	$\Omega_4 = (4, 3)$	200

(12.48)

Consequently, for the multicommodity problem the constraints described above become

1. The flow conservation equations for all commodities are

$$\sum_{j \in A(i)} f_{ij}^\Omega - \sum_{j \in B(i)} f_{ji}^\Omega = r_i^\Omega, \ \forall i \in \mathcal{N}, \quad \forall \Omega \in W$$

   where

$$r_i^\Omega = \begin{cases} g^\Omega & \text{if } O^\Omega = i \\ -g^\Omega & \text{if } D^\Omega = i \\ 0 & \text{if } i \text{ is an intermediate node} \\ & \text{for the commodity } \Omega. \end{cases}$$

2. The total link flow is given by superposing the link flow of all commodities:

$$\sum_{\Omega \in W} f_{ij}^\Omega = f_{ij}, \ \forall (i, j) \in \mathcal{A}$$

In matrix form the previous relations can be stated as

$$\begin{array}{rcl} \mathbf{E}\mathbf{f}^\Omega & = & \mathbf{r}^\Omega \ \forall \Omega \in W \\ \sum\limits_{\Omega \in W} \mathbf{f}^\Omega & = & \mathbf{f} \end{array}$$

where $\mathbf{E}$ is the $n \times l$ node–link incidence matrix whose element in the row corresponding to node i, and the column corresponding to the link (j, k) is defined as

$$e_{i(jk)} = \begin{cases} +1 & \text{if } i = j \\ -1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}$$

Note that the conservation flow equations are equivalent to consider the total link flow, specifically, the sum of the link flows associated with all commodities. It is possible to reduce the amount of copies of the network by using a network for all the O–D pairs with the same origin, or alternatively for all O–D pairs with the same destination.

Congested Traffic

We have considered the structure of graphs and road transportation networks. It is characteristic of the road networks that the passage of flow through the network creates delays (effect of congestion) and, as a result of growing traffic volumes, the speed on a link tends to decrease. To take the congestion effect into account, we introduce the notion of the *link cost*, $c_{ij}(f_{ij})$, as the average travel time in traversing the street segment defined by the link (i, j) with flow f_{ij}. These functions are usually modeled as positive, nonlinear, and strictly increasing functions in the analysis of traffic systems. The basic parameters of a link performance function, relating travel time, c_{ij}, on link (i, j), to the flow f_{ij}, on the link, are the *free-flow travel time*, c^0_{ij}, which is a measure of the travel time at zero flow, and the practical capacity of the link, k_{ij}, which is a measure of the flow from which the travel time will increase very rapidly if the flow is further increased. The most common expression for $c_{ij}(f_{ij})$ is called the BPR function

$$c_{ij}(f_{ij}) = c^0_{ij} + b_{ij}(f_{ij}/k_{ij})^{n_{ij}}, \tag{12.49}$$

where b_{ij} and n_{ij} are parameters to be calibrated.

12.6.3 The Traffic Assignment Problem

We must introduce a principle that models user behavior in their route choice in the transportation network. Wardrop [102] was the first to formally enunciate this principle:

> "Under equilibrium conditions, traffic in congested networks arranges itself in such a way that no individual tripmaker can reduce his paths cost by switching routes."

This principle assumes perfect information of all users, and means that they would change the route if this produces a time reduction. A corollary of this principle is that if all tripmakers perceive costs in the same way, under equilibrium conditions, then traffic in congested networks arranges itself such that all used routes between an O–D pair have equal and minimum costs while all unused routes have greater or equal costs than used routes.

This principle has been used as a framework to build equilibrium assignment models. Beckman and McGuire [10] formulated the optimization problem below to express the equilibrium condition derived from Wardrop's first principle [traffic assignment problem] (TAP). Minimize

$$Z = \sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} c_{ij}(x)dx \tag{12.50}$$

subject to

$$\begin{array}{rcl} \sum_{j\in A(i)} f_{ij}^{\Omega} - \sum_{j\in B(i)} f_{ji}^{\Omega} & = & r_i^{\Omega},\ \forall i \in \mathcal{N},\ \ \forall \Omega \in W \\ \sum_{\Omega\in W} f_{ij}^{\Omega} & = & f_{ij},\ \forall (i,j)\in\mathcal{A} \\ f_{ij}^{\Omega} & \geq & 0,\ \ \forall (i,j)\in\mathcal{A}, \forall\Omega\in W \end{array}$$

This formulation of the problem is known as the *link-node* formulation. Since the equilibrium conditions are given in terms of route flows and costs, it follows that the optimization problem is based on route flows. Next, an alternative formulation of the equilibrium conditions based on path flows, called the *linkflow formulation*, is given.

The Linkflow Formulation

The main elements of this formulation are

1. **Data.**

   $\mathcal{R}_\Omega$**:** the set of routes for the commodity Ω

   $c_a(x)$**:** the cost associated with flow x through arc a

2. **Variables.**

   h_r**:** the flow in route r

3. **Constraints.** The amount of users of a demand pair Ω is the sum of the total amount of users in different paths satisfying the demand

   $$\sum_{r\in\mathcal{R}_\Omega} h_r = g_\Omega,\ \forall\Omega\in W \tag{12.51}$$

   Moreover, the flow must be nonnegative

   $$h_r \geq 0,\ \forall r\in\mathcal{R}_\Omega,\ \forall\Omega\in W \tag{12.52}$$

   The relationship between linkflow and routeflow is that the flow on each link $a\in\mathcal{A}$ is the sum of the flow in all paths that use it:

   $$\sum_{w\in W}\sum_{r\in\mathcal{R}_\Omega} \delta_{a,r} h_r = f_a \quad \forall a\in\mathcal{A} \tag{12.53}$$

   where

   $$\delta_{a,r} = \begin{cases} 1 & \text{if } r\in\mathcal{R}_\Omega \text{ contains arc } a \\ 0 & \text{otherwise} \end{cases}$$

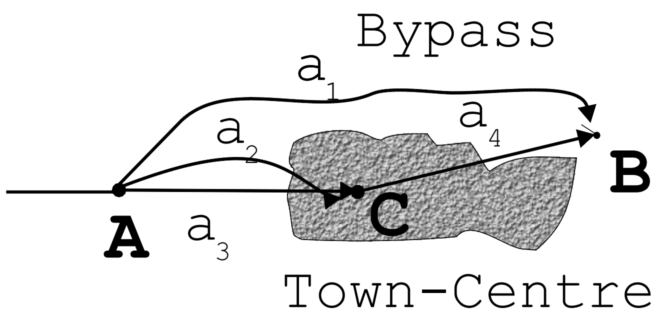

Figure 12.9: Diagram showing the routes.

4. **Function to be optimized.** In this problem, we minimize

$$Z = \sum_{a \in \mathcal{A}} \int_0^{f_a} c_a(x)dx$$

Thus, the linkflow formulation of TAP can be stated as follows. Minimize

$$Z = \sum_{a \in \mathcal{A}} \int_0^{f_a} c_a(x)dx$$

subject to

$$\sum_{r \in \mathcal{R}_\Omega} h_r = g_\Omega, \ \forall \Omega \in W \tag{12.54}$$

$$\sum_{\Omega \in W} \sum_{r \in \mathcal{R}_\Omega} \delta_{ar} h_r = f_a \ , \forall a \in \mathcal{A} \tag{12.55}$$

$$h_r \geq 0, \ \forall r \in \mathcal{R}_\Omega, \ \forall \Omega \in W \tag{12.56}$$

Example 12.10 (Linkflow formulation). To illustrate the previous constraints, consider the problem of a town served with a bypass and several town-center routes as illustrated in Figure 12.9. Assume that there are 4000 trips from A to B, and 2500 trips from A to C. The routes available to satisfy the demand pair $\Omega_1 = (A, B)$ are $r_1 = \{a_1\}$, $r_2 = \{a_2, a_4\}$, and $r_3 = \{a_3, a_4\}$, and the routes for the pair $\Omega_2 = (A, C)$ are $r_4 = \{a_2\}$ and $r_5 = \{a_3\}$. In this example $W = \{1, 2\}$, and $\mathcal{R}_{\Omega_1} = \{r_1, r_2, r_3\}$ and $\mathcal{R}_{\Omega_2} = \{r_4, r_5\}$. The path flow variables are $h_1, \ldots, h_5$, and the line flow variables are $f_1, \ldots, f_4$.

For this example the constraints of the problem are:

- Constraints (12.51)

$$\begin{aligned} h_1 + h_2 + h_3 &= 4000 \\ h_4 + h_5 &= 2500 \end{aligned}$$

- Constraints (12.53)

$$\begin{aligned} h_1 &= f_1 \\ h_2 + h_4 &= f_2 \\ h_3 + h_5 &= f_3 \\ h_2 + h_3 &= f_4 \end{aligned}$$

- Constraints (12.52):

$$h_1, \ldots, h_5 \geq 0$$

■

Appropriateness of the Model

We show below that this mathematical model is appropriate for describing the Wardrop's equilibrium conditions. The first observation is

$$\begin{aligned} C_a(f_a) &= \int_0^{f_a} c_a(x)dx \\ C'_a(f_a) &= c_a(f_a) \\ C''_a(f_a) &= c'_a(f_a) \end{aligned}$$

Since $c_a(f_a)$, the cost of the link as a function of the service level in this link, is a nondecreasing function, this implies that $c'_a(f_a) \geq 0$ and that $C_a(f_a)$ is a convex function. Since the objective function is the sum of convex functions, it is also a convex function. Since the constraints are linear, the optimization problem becomes a convex mathematical programming problem. This implies that the KKTCs are necessary and sufficient conditions. We shall show by means of the KKTCs that the optimal solution of the TAP meets the equilibrium conditions.

By expressing the link flows f_a in terms of the path flows in the objective function (12.54) [using Equation (12.55)], the problem can be formulated only in terms of the path flow variables $\{h_r : r \in \mathcal{R}_\Omega, \ \Omega \in W\}$ as follows. Minimize

$$Z = \sum_{a \in \mathcal{A}} \int_0^{\sum\limits_{\Omega \in W} \sum\limits_{r \in \mathcal{R}_\Omega} \delta_{ar} h_r} c_a(x)dx$$

subject to

$$\sum_{r \in \mathcal{R}_\Omega} h_r = g_\Omega, \ \forall \Omega \in W \tag{12.57}$$

$$h_r \geq 0, \ \forall r \in \mathcal{R}_\Omega, \ \forall \Omega \in W \tag{12.58}$$

The Lagrangian function of this problem is

$$\mathcal{L}(\mathbf{h}, \boldsymbol{\lambda}, \boldsymbol{\psi}) = Z + \sum_{\Omega \in W} \lambda_\Omega \left(g_\Omega - \sum_{r \in \mathcal{R}_\Omega} h_r \right) + \sum_{\Omega \in W} \sum_{r \in \mathcal{R}_\Omega} \psi_r(-h_r)$$

and the KKT condition (8.3) are

$$\frac{\partial \mathcal{L}}{\partial h_{r'}} = \sum_{a\in\mathcal{A}} c_a \left(\sum_{\Omega\in W} \sum_{r\in\mathcal{R}_\Omega} \delta_{ar} h_r \right) \delta_{ar'} - \lambda_{\Omega'} - \psi_{r'} = 0 \tag{12.59}$$

where $r' \in \mathcal{R}_{\Omega'}$. Note that $\sum_{\Omega\in W} \sum_{r'\in\mathcal{R}_\Omega} \delta_{ar} h_{r'}$ gives the flow on the link a, and $\sum_{a\in\mathcal{A}} c_a(f_a)\delta_{ar'}$ is the sum of the cost associated with all arcs contained in the route r', specifically, it is the length of path r'. We denote this quantity as $C^*_{r'}$. The condition (12.59) becomes

$$C^*_{r'} = \lambda_{\Omega'} + \psi_{r'}$$

The slackness condition leads to $\psi_r h_r = 0$, so $h_r = 0$ or $\psi_r = 0$. If $h_{r'} > 0$ then $\psi_{r'} = 0$ and $C^*_{r'} = \lambda_{\Omega'}$. Otherwise, using the nonnegativity of the multiplier $\psi_{r'}$ we obtain $C^*_{r'} = \lambda_{\Omega'} + \psi_{r'} \geq \lambda_{\Omega'}$. In other words, the preceding condition says that the a set of paths flows that is optimal must be positive on paths with a minimum cost length. The condition also implies that at an optimum, the paths along which the demand g_Ω of O–D pair Ω is split must have equal lengths (and length less than or equal to that of all other paths of Ω).

A GAMS implementation

Below, we use the GAMS package to implement the TAP formulation. The test example is that stated in Section 12.6.2 and illustrated in Figure 12.8. This example uses the BPR functions as link costs, and the objective function becomes

$$\begin{aligned} Z &= \sum_{a\in\mathcal{A}} C_a(f_a) = \sum_{a\in\mathcal{A}} \int_0^{f_a} c_a(x)dx = \sum_{a\in\mathcal{A}} \int_0^{f_a} \left[c_a^0 + b_a \left(\frac{x}{k_a}\right)^{n_a} \right] dx \\ &= \sum_{a\in\mathcal{A}} \left[c_a^0 f_a + \frac{b_a}{n_a+1} \left(\frac{f_a}{k_a}\right)^{n_a+1} \right] = \sum_{a\in\mathcal{A}} \left(c_a^0 f_a + d_a {f_a}^{m_a} \right) \end{aligned}$$

where

$$d_a = \frac{b_a}{(n_a+1)k_a^{n_a+1}} \text{ and } m_a = n_a + 1$$

The parameters of the ND network are shown in Table 12.9.

A GAMS input file is provided below:

```
$title TRAFFIC ASSIGNMENT PROBLEM.

** Sets are declared in first place.
** Set N is the set of nodes of the road network.
** Set A(N,N) is the set of links.
** Set W is the set of O--D pairs

SET
```

Table 12.9: Parameters of the link cost functions of the Nguyen–Dupuis network

Link	c_a^0	d_a	m_a	Link	c_a^0	d_a	m_a
(1,5)	7.0	0.00625	2.0	(1,12)	9.0	0.00500	2.0
(4,5)	9.0	0.00500	2.0	(4,9)	12.0	0.00250	2.0
(5,6)	3.0	0.00375	2.0	(5,9)	9.0	0.00375	2.0
(6,7)	5.0	0.00625	2.0	(6,10)	13.0	0.00250	2.0
(7,8)	5.0	0.00625	2.0	(7,11)	9.0	0.00625	2.0
(8,2)	9.0	0.00625	2.0	(9, 10)	10.0	0.00250	2.0
(9,13)	9.0	0.00250	2.0	(10,11)	6.0	0.00125	2.0
(11,2)	9.0	0.00250	2.0	(11,3)	8.0	0.00500	2.0
(12,6)	7.0	0.00125	2.0	(12,8)	14.0	0.00500	2.0
(13,3)	11.0	0.00500	2.0				

```
N set of nodes /I1*I13/
W pairs        /W1*W4/
A(N,N) set of links
/I1.(I5,I12)
I4.(I5,I9)
I5.(I6,I9)
I6.(I7,I10)
I7.(I8,I11)
I8.(I2)
I9.(I10,I13)
I10.(I11)
I11.(I2,I3)
I12.(I6,I8)
I13.(I3)/

ODE(N,W) set of origins by demand
/I1.(W1,W2)
I4.(W3,W4)/

DDE(N,W) set of destinations by demand
/I2.(W1,W3)
I3.(W2,W4)/;

** The set of nodes I must be  duplicated to refer to its different element
*  in the same constraint.

ALIAS(N,I)
ALIAS(N,J)

PARAMETER
G(W) demand in the O--D pair  W
/W1 400
W2 800
W3 600
W4 200/;
```

```
TABLE  CDATA(N,N,*)  link cost parameters
         CO      D       M
 I1.I5    7.0  0.00625   2.0
 I1.I12   9.0  0.00500   2.0
 I4.I5    9.0  0.00500   2.0
 I4.I9   12.0  0.00250   2.0
 I5.I6    3.0  0.00375   2.0
 I5.I9    9.0  0.00375   2.0
 I6.I7    5.0  0.00625   2.0
 I6.I10  13.0  0.00250   2.0
 I7.I8    5.0  0.00625   2.0
 I7.I11   9.0  0.00625   2.0
 I8.I2    9.0  0.00625   2.0
 I9.I10  10.0  0.00250   2.0
 I9.I13   9.0  0.00250   2.0
 I10.I11  6.0  0.00125   2.0
 I11.I2   9.0  0.00250   2.0
 I11.I3   8.0  0.00500   2.0
 I12.I6   7.0  0.00125   2.0
 I12.I8  14.0  0.00500   2.0
 I13.I3  11.0  0.00500   2.0;

** Optimization variables are declared.

VARIABLES
z         objective function variable
f(I,J)    the flow at the link I-J
fc(W,I,J) the flow of the commodity W at the link I-J
cos(I,J)  cost in the link I-J at equilibrium;

POSITIVE VARIABLE fc(W,N,N);

** Constraints are declared.

EQUATIONS
COSTZ Objective function
BALANCE(W,I) conservation of flow condition for the commodity W.
FLOW(I,J) total link flow
COST(I,J) cost of the link I-J;

** The objective function is the sum of the integrated cost
** in all links.
** The equation FLOW illustrate the use of dollar operation
** to restrict the number of constraints generated to less
** than implied by the domain of the defining sets.

COSTZ.. z=E= SUM((I,J)$A(I,J),CDATA(I,J,'CO')*f(I,J)
              +CDATA(I,J,'D')*f(I,J)**CDATA(I,J,'M'));
BALANCE(W,I) .. SUM(J$A(I,J),fc(W,I,J))-SUM(J$A(J,I),fc(W,J,I))
              =E=G(W)$ODE(I,W)-G(W)$DDE(I,W);
FLOW(I,J)$A(I,J)..SUM(W,fc(W,I,J))=E=f(I,J);
COST(I,J)$A(I,J)..cos(I,J)=E=CDATA(I,J,'CO')
             +2*CDATA(I,J,'D')*f(I,J);

** The next two sentences define the ND (Nguyen-Dupis) problem,
** considering all the above constraints, and direct GAMS to
** solve the problem using the nlp solver.
```

```
MODEL ND /ALL/;
SOLVE ND USING nlp MINIMIZING z;
```

Using the previous code we compute link flows and costs at equilibrium, which are shown in Table 12.10. The resulting information – the service level of the network transportation links – is crucial in transportation planning.

Table 12.10: Linkflows and costs at equilibrium

Link	f_a	$c_a(f_a)$	Link	f_a	$c_a(f_a)$
(1,5)	675.144	15.439	(1,12)	524.856	14.249
(4,5)	102.571	10.026	(4,9)	697.429	15.487
(5,6)	416.187	6.121	(5,9)	361.528	11.711
(6,7)	356.416	9.455	(6,10)	184.626	13.923
(7,8)	102.571	6.282	(7,11)	253.845	12.173
(8,2)	502.571	15.282	(9,10)	497.429	12.487
(9,13)	561.528	11.808	(10.I11)	682.056	7.705
(11.I2)	497.429	11.487	(11.I3)	438.472	12.385
(12.I6)	124.856	7.312	(12.I8)	400.000	18.000
(13.I3)	561.528	16.615			

We also illustrate that the equilibrium conditions are satisfied. To this end, we can compute the paths used through the link flow for every commodity. Moreover, the link costs at equilibrium are used to compute the costs of the path used. This information is shown in Table 12.11. For each origin and destination the table provides all optimal paths with their corresponding flows and costs.

It can be observed that the paths used for the same commodity have approximately the same cost (see commodities Ω_2, Ω_3). A second observation is that the linkflows at equilibrium are unique, but this is not true for pathflows. In this example paths flows satisfying

$$\begin{aligned} h_5 + h_6 &= 124.856 \\ h_3 + h_4 &= 313.616 \\ h_5 + h_3 &= 253.845 \\ h_4 + h_6 &= 184.626 \end{aligned}$$

produce linkflows at equilibrium.

12.6.4 Side-Constrained Assignment Models

For applications to transportation networks, it is sometimes important to consider a *capacitated network*, in which the following constraints are introduced:

$$s_k(\mathbf{f}) \leq 0, \quad \forall k \in \mathcal{K}$$

Table 12.11: Paths used at the equilibrium for TAP

O–D Pair	Used paths	h_r	C_r
$\Omega_1 = (1-2)$	$r_1 = 1-12-8-2$	400	47.53
$\Omega_2 = (1-3)$	$r_2 = 1-5-9-13-3$	361.528	55.57
	$r_3 = 1-5-6-7-11-3$	h_3	55.62
	$r_4 = 1-5-6-10-11-3$	h_4	55.57
	$r_5 = 1-12-6-7-11-3$	h_5	55.62
	$r_6 = 1-12-6-10-11-3$	h_6	55.56
$\Omega_3 = (4-2)$	$r_7 = 4-5-6-7-8-2$	102.571	47.17
	$r_8 = 4-9-10-11-2$	184.626	47.16
$\Omega_4 = (4-3)$	$r_9 = 4-9-13-3$	200	43.91

where the index set $\mathcal{K}$ may, for instance, consist of the index set of the links, nodes, routes or any combinations of subsets of them. A special case of these constraints are

$$0 \le f_a \le u_a, \quad \forall a \in \mathcal{B} \subset \mathcal{A} \tag{12.60}$$

that lead to the *capacitated traffic assignment problem*, CTAP.

In designing a future network, and depending on the service level to be provided, the traffic engineer computes the capacities of streets and intersections as a function of the road widths, number of lanes, shoulder widths, gradients, traffic signalization, and other factors.

If we use the TAP to predict the service level, then the predicted flow on some links may be lower or greater than the traffic assumed a priori by the engineer. This is due to the fact that TAP allows every road to carry an arbitrarily large traffic volume. It is clear, then, that inequalities such as (12.60) are significant for the planning process.

For the capacitated problem a simple optimality condition, similar to that of the Wardropp's first principle, can be formulated. Assume that the used routes for a demand Ω are numbered in the order of increasing costs, and that they are denoted as c_i^Ω; $i = 1, 2, \dots, l$. Assume also that the number of routes is l, and among these the first m are *saturated*, containing at least one link which carries flow at its capacity limit. The equilibrium exists if and only if

$$c_1^\Omega \le c_2^\Omega, \dots \le c_m^\Omega \le c_{m+1}^\Omega = \cdots c_l^\Omega, \quad \forall \Omega \in W$$

Now we discuss two ways of taking into account side constraints:

1. We consider explicitly the upper bounds on the link flows (12.60). Taking into account the link capacities is equivalent to modifying the link costs in the following way

$$\hat{c}_a(f_a) = c_a(f_a) + \phi_a(f_a), \quad \forall a \in \mathcal{B}$$

where

$$\phi_a(f_a) = \begin{cases} +\infty & \text{if } f_a > k_a \\ 0 & \text{if } f_a \le k_a \end{cases}$$

the functions ϕ_a are not continuous, and this introduces a new difficulty.

2. We use the so-called travel-time function, which tend to infinity when the link flows approach their respective capacities. This is similar to barrier functions. In this approach, we approximate the function ϕ_a by means of a continuously differentiable function. For example, the implicit case considers approximations such as

$$\hat{c}_a(f_a) = c_a(f_a) + s_a \ln\left(1 + \frac{q_a}{k_a - f_a}\right), \forall a \in \mathcal{B}$$

where $s_a \to 0$, and k_a is the maximum capacity of link a. These functions are not defined on flow links such that their flow is over the capacity, and a special recovery procedure is needed to handle infeasible points. This fact may produce computational errors when the nonlinear solver progresses. We approximate the functions ϕ_a by a BPR formula $s_a\,(f_a/k_a)^{n_a}$, and the link cost functions become

$$\bar{c}_a(f_a) = c_a(f_a) + s_a \left(\frac{f_a}{k_a}\right)^{n_a}, \quad \forall a \in \mathcal{B} \tag{12.61}$$

where $n_a \to +\infty$.

Example 12.11 (capacitated network). To illustrate both approaches, we have used a test example adding to the ND network the side constraints

$$\begin{aligned} f_{10,11} &\le 400 \\ f_{12,8} &\le 300 \end{aligned}$$

This example is called ND-C, and the set $\mathcal{B}$ is $\{(10,11),(12,8)\}$. In the implicit approach we have used the values $n_a = 9$, and $s_a = 20$, $\forall a \in \mathcal{B}$.

Now we list the new commands to deal with the capacitated traffic assignment problem. To compress the code, we have shown only the new command to be added to the previously described TAP mode. The code to deal with the side constraints can be obtained by changing the links costs $c_a(f_a)$ by $\bar{c}_a(f_a)$. So, it is not listed.

```
$title CAPACITATED TRAFFIC ASSIGNMENT PROBLEM.

** Set B(N,N) is the set of bounded links. This is a new set.

SET
...

B(N,N) set of  bounded links
               /I10.I11
               I12.I8/;
PARAMETER
U(I,J) upper bound in link I-J
               /I10.I11 400
                I12.I8  300/
...

** Upper bounds for the link capacities.

VARIABLES
...
f.up(I,J)$B(I,J)=U(I,J);
```

We have obtained the link flows, the costs, and the equilibrium using the two approaches, which are shown in Table 12.12. Using the link flows for each commodity we have computed the paths used and their costs. This information is also shown in Table 12.13. It can be observed that the equilibrium conditions are satisfied. The pairs Ω_1 and Ω_3 illustrate the fact that saturated routes have less cost than nonsaturated ones. Moreover, the fact that the nonsaturated routes have the same cost is demonstrated in the pairs Ω_1 and Ω_2. In Table 12.13 we have computed the route cost using the cost $c_a(f_a)$ instead of $\bar{c}_a(f_a)$. It is relevant to note that the link cost in capacitated links is unrealistically high for the implicit approach (see Table 12.12). This indicates that the equilibrium conditions for the implicit approach, Wardrop's first principle, do not hold in the obtained approximation. This is due to the high nonlinearity of the capacitated link costs. This is a computational difficulty for the implicit approach.

From a modeling point of view, explicit upper bounds have the advantage of allowing links flows to attain the capacity values, whereas the use of travel-time functions such as (12.61) will force all link flows to be strictly greater than the capacities (see Table 12.12). Note that both approaches generate the same routes, but there exists a small difference in the load of these routes, and the cost. It can be concluded that the predictions of both approaches are not significantly different.

An important advantage of explicit modeling is the interpretation of the Lagrangian multipliers for the capacitated constraints. The Lagrangian multipliers for this example are shown in Table 12.12. They measure the time gained by the users of saturated routes compared to the fastest route still available. For example, the route r_1 is saturated because it contains the link $(12, 8)$. The Lagrangian multiplier associated with this link is 11.501, which is approximately equal to the difference between the costs of the saturated route and the nonsaturated routes, i.e., $58.70 - 47.20$.

This interpretation shows that the capacitated equilibrium link flow pattern can be found by solving the corresponding uncapacitated problem TAP with the travel time functions adjusted to $\hat{c}_a(f_a) = c_a(f_a) + \beta_a$ for all $a \in \mathcal{B}$, where β_a are the Lagrangian multipliers. The reader can prove in this example that using the link costs $\hat{c}_a$ leads used paths to satisfy Wardrop's first principle.

Finally, we can conclude that the implicit approach has the computational advantage of using the uncapacitated assignment model but has the disadvan-

Table 12.12: Link flows and costs at the equilibrium for CTAP, and Lagrangian multipliers

	Explicit approach		Implicit approach	
Link	f_a	$c_a(f_a)$	f_a	$c_a(f_a)$
(1,5)	689.920	15.624	690.242	15.628
(1,12)	510.080	14.101	509.758	14.098
(4,5)	200.000	11.000	178.735	10.787
(4,9)	600.000	15.000	621.265	15.106
(5,6)	400.266	6.002	393.659	5.952
(5,9)	489.655	12.672	475.318	12.565
(6,7)	610.345	12.629	586.810	12.335
(6,10)	.	13.000	.	13.000
(7,8)	267.749	8.347	253.988	8.175
(7,11)	342.597	13.282	332.822	13.160
(8,2)	567.749	16.097	570.595	16.132
(9,10)	400.000	12.000	421.265	12.106
(9,13)	689.655	12.448	675.318	12.377
(10,11)	400.000	7.000	421.265	7.053
	($\beta_a = 8.914$)			$\bar{c}_a = 342.775$
(11,2)	432.251	11.161	429.405	11.147
(11,3)	310.345	11.103	324.682	11.247
(12,6)	210.080	7.525	193.151	7.483
(12,8)	300.000	17.000	316.607	17.166
	($\beta_a = 11.501$)			$\bar{c}_a = 359.950$
(13,3)	689.655	17.897	675.318	17.753

Table 12.13: Flow and cost paths at equilibrium for CTAP

		Approach			
		Explicit		Implicit	
O–D Pair	Used paths	h_r	C_r	h_r	C_r
Ω_1	$r_1 = 1-12-8-2$	300	47.20	316.6	47.40
(1-2)	$r_2 = 1-12-6-7-8-2$	67.75	58.70	75.25	58.23
	$r_3 = 1-12-6-7-11-2$	32.25	58.70	8.14	58.23
Ω_2	$r_4 = 1-5-6-7-11-3$	200.3	58.64	214.9	58.33
(1-3)	$r_5 = 1-5-9-13-3$	489.7	58.64	475.3	58.33
	$r_6 = 1-12-6-7-11-3$	110.1	58.64	109.8	58.32
Ω_3	$r_9 = 4-9-10-11-2$	400	45.10	421.3	45.42
(4-2)	$r_{10} = 4-5-6-7-8-2$	200	54.07	178.74	53.39
Ω_4	$r_{11} = 4-9-13-3$	200	45.34	200	45.24
(4-3)					

tage from a modeling point of view of overloading the capacitated routes, and difficulties in the interpretation of the Lagrangian multipliers. ∎

12.6.5 The Variable-Demand Case

The TAP has been formulated as a problem with fixed demand, but it is more realistic to consider the *elastic* nature of te demand. Travelers have a number of choices available and are motivated by economical considerations in their ecisions. For example, as congestion increases, motorists may decide to use a different mode of transport (e.g. underground).

In order to take the elastic nature of the demand into account, the number of trips, g_Ω, between the pair Ω in the network can be assumed to be a function of the travel cost for that pair Ω:

$$g_\Omega = G_\Omega(c_\Omega)$$

where c_Ω is the minimum travel cost for the pair Ω, and G_Ω is the demand function. We assume that G_Ω is nonnegative, continuous, and strictly decreasing for each $\Omega \in W$. Its inverse function gives the number of trips as a function of the travel cost, i.e. $c_\Omega = G_\Omega^{-1}(g_\Omega)$.

The equilibrium condition states that the O–D trip rate satisfy the demand function, and that the travel times on all used paths between any O–D pair are equal, and are also equal to or less than the travel times on any unused paths.

The following model combines variable-demand and assignment. It is possible to demonstrate that the equilibrium conditions are those obtained by solving the following problem (TAPE). Minimize

$$Z = \sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} c_{ij}(x)dx - \sum_{\Omega\in W} \int_0^{g_\Omega} G_\Omega^{-1}(x)dx$$

subject to

$$\begin{aligned} \sum_{j\in A(i)} f_{ij}^\Omega - \sum_{j\in B(i)} f_{ji}^\Omega &= r_i^\Omega, \ \forall i \in \mathcal{N}, \quad \forall\Omega \in W & (12.62)\\ \sum_{\Omega\in W} f_{ij}^\Omega &= f_{ij}, \forall(i,j) \in \mathcal{A} & (12.63)\\ f_{ij}^\Omega &\geq 0 \quad \forall\Omega \in W, \forall(i,j) \in \mathcal{A} & (12.64) \end{aligned}$$

where

$$r_i^\Omega = \begin{cases} g_\Omega & \text{if } O^\Omega = i \\ -g_\Omega & \text{if } D^\Omega = i \\ 0 & \text{if i is an intermediate node for the commodity } \Omega \end{cases}$$

Note that for TAPE the terms r_i^Ω are variables, but for TAP they are constants.

To illustrate the TAPE, we deal with a particular case of TAPE. Assume that the travelers choose between several modes of transport, and a logit function (demand model) gives the number of trips taken on each alternative, g_Ω^k, by the equation

$$g_\Omega^k = G_\Omega^k(\mathbf{c}_\Omega) = \frac{\exp[-\left(\alpha^k + \beta_1 c_\Omega^k\right)]}{\sum\limits_{k'} \exp[-\left(\alpha^{k'} + \beta_1 c_\Omega^{k'}\right)]}\bar{g}_\Omega \tag{12.65}$$

where c_Ω^k is the user's perception of the generalized cost of traveling of the pair $\Omega = (i,j)$ by mode k, that corresponds to a user optimal route choice of the network; $\{\mathbf{c}_\Omega\}$ is the vector of generalized costs for all the modes present, $\bar{g}_\Omega$ is the demand for the O–D pair Ω by all considered modes; and α^k, β_1 are parameters of the logit model. For two alternatives, such as car (a) and public transport (b), (12.65) simplifies to

$$G_\Omega^a(\mathbf{c}_\Omega) = \frac{1}{1 + \exp[-\left(\alpha^{ab} + \beta_1(c_\Omega^b - c_\Omega^a)\right)]}\bar{g}_\Omega$$

where $\alpha^{ab} = \alpha^b - \alpha^a$. We assume that the travel cost c_Ω^b by public transportation is independent of traffic volumes, and by this reason it is constant. The inverse function of the demand model becomes

$$c_\Omega^a = G^{-1}(g_\Omega^a) = c_\Omega^b + \frac{1}{\beta_1}\left[\alpha^{ab} + \log(\bar{g}_\Omega - g_\Omega^a) - \log(g_\Omega^a)\right]$$

Using the relationship $g_\Omega^a + g_\Omega^b = \bar{g}_\Omega$, we obtain the equality

$$-\int_0^{g_\Omega^a} G^{-1}(x)dx = c_\Omega^b g_\Omega^b + \left(\frac{1}{\beta_1}\right) \sum_{k\in\{a,b\}} g_\Omega^k(\log g_\Omega^k - 1 + \alpha^k) + C$$

where C is a constant. The objective function then becomes

$$\sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} c_{ij}(x)dx + \sum_{\Omega\in W} c_\Omega^b g_\Omega^b + \left(\frac{1}{\beta_1}\right) \sum_{\Omega\in W} \sum_{k\in\{a,b\}} g_\Omega^k(\log g_\Omega^k - 1 + \alpha^k)$$

Example 12.12 (Variable-demand case). To illustrate the TAPE model consider that there exists an underground network. Assume that there exist connections to satisfy the four demands pairs, and the trip times are flow independent (Table 12.14).

The following code implements this model in GAMS.

```
$title TRAFFIC ASSIGNMENT PROBLEM WITH ELASTIC DEMAND.

** Sets are declared in first place.
** Set N is the set of nodes of the road network.
** Set A is the set of links.
** Set W is the set of O--D pairs

SET
N set of nodes /I1*I13/
W pairs         /W1*W4/
A(N,N) set of links
/I1.(I5,I12)
```

Table 12.14: Inputs for TAPE

O–D pair	$\bar{g}_\Omega$	c^b_Ω	Logit parameters
Ω_1	600	41	$\alpha^a = -2.0$
Ω_2	1000	46	$\alpha^b = 0.0$
Ω_3	800	43	$\beta_1 = 0.1$
Ω_4	400	40	

```
I4.(I5,I9)
I5.(I6,I9)
I6.(I7,I10)
I7.(I8,I11)
I8.(I2)
I9.(I10,I13)
I10.(I11)
I11.(I2,I3)
I12.(I6,I8)
I13.(I3)/

ODE(N,W) set of origins by demand
/I1.(W1,W2)
I4.(W3,W4/

DDE(N,W) set of destinations by demand
/I2.(W1,W3)
I3.(W2,W4)/;

ALIAS(N,I)
ALIAS(N,J)

PARAMETER
G(W) demand in O--D pair  W
/W1 600
W2 1000
W3 800
W4 400/

Cb(W) travel cost by public transport in O--D pair  W
/W1 41
W2 35
W3 43
W4 40/

BETA
ALPHAa
ALPHAb;
BETA=0.1;
ALPHAa=-2.;
ALPHAb=0.;

TABLE  CDATA(N,N,*)  link cost parameters
          C0      D
 I1.I5    7.0  0.00625
 I1.I12   9.0  0.005
 I4.I5    9.0  0.005
 I4.I9   12.0  0.0025
 I5.I6    3.0  0.00375
 I5.I9    9.0  0.00375
 I6.I7    5.0  0.00625
 I6.I10  13.0  0.0025
 I7.I8    5.0  0.00625
 I7.I11   9.0  0.00625
 I8.I2    9.0  0.00625
 I9.I10  10.0  0.0025
 I9.I13   9.0  0.0025
 I10.I11  6.0  0.00125
 I11.I2   9.0  0.0025
 I11.I3   8.0  0.005
 I12.I6   7.0  0.00125
 I12.I8  14.0  0.005
 I13.I3  11.0  0.005;


** Optimization variables are declared.

VARIABLES
 z          objective function variable
 f(N,N)     flow link
 fc(W,N,N)  is the flow link of the commodity W
 cos(N,N)   cost in the link at equilibrium
 ga(W)      demand by car
 gb(W)      demand by public transport;

POSITIVE VARIABLE fc(W,N,N);

ga.LO(W)=0.01;
gb.LO(W)=0.01;

** Constraints are declared.

EQUATIONS
 COST Objective function
 BALANCE(W,I) conservation of flow condition for the commodity W.
 FLOW(I,J) total link flow
 MODAL(W) Modal split of the demand
 COSTE(I,J) cost at equilibrium;

COST .. z=E= SUM((I,J)$A(I,J),CDATA(I,J,'CO')*f(I,J)
             +CDATA(I,J,'D')*f(I,J)**CDATA(I,J,'M'))
+SUM(W,Cb(W)*gb(W))+1/BETA* SUM(W,ga(W)*(-1+ALPHAa+LOG(ga(W)))
             + gb(W)*(-1+ALPHAb+LOG(gb(W))) );
BALANCE(W,I) .. SUM(J$A(I,J),fc(W,I,J))-SUM(J$A(J,I),fc(W,J,I))
                =E=ga(W)$ODE(I,W)-ga(W)$DDE(I,W);
FLOW(I,J)$A(I,J)..SUM(W,fc(W,I,J))=E=f(I,J);
COSTE(I,J)$A(I,J)..cos(I,J)=E=CDATA(I,J,'CO')
                              +2*CDATA(I,J,'D')*f(I,J);
MODAL(W).. G(W)=E=ga(W)+gb(W);
MODEL nd /ALL/;
SOLVE nd USING nlp MINIMIZING z;
```

The results obtained are shown in Table 12.15.

Table 12.15: Cost at the equilibrium for TAPE and modal split

O–D Pair	c^a_Ω	c^b_Ω	g^a_Ω	g^b_Ω
Ω_1	49.29	41	459.821	140.179
Ω_2	56.73	46	736.288	263.712
Ω_3	48.52	43	647.730	152.270
Ω_4	45.56	40	323.124	76.876

Some comments are in order. For example, consider that the system operator increases train frequencies to satisfy the demand pair Ω_2, and the travel time is reduced to 35. The operator wishes to compute the new demand and the congestion level on the road network for this demand. The outputs obtained with TAPE are shown in Table 12.16. Note that the demand in public transportation for the pair Ω_2 increases. This produces a reduction of the congestion level on the road network, and the alternative car for the others pairs is slightly more attractive.

Table 12.16: Cost at equilibrium for TAPE, and modal split with intervention on the transportation system

O–D Pair	c^a_Ω	c^b_Ω	g^a_Ω	g^b_Ω
Ω_1	48.53	41	466.009	133.991
Ω_2	52.79	35	554.843	445.157
Ω_3	48.34	43	649.947	150.053
Ω_4	44.56	40	329.594	70.406

■

12.6.6 Combined Distribution and Assignment

One way of dealing with the four steps of the planning process (explained in Section **??**) is to merge as many steps as possible into one, in particular, we can include assignment and distribution in the same process. We assume that the number of trips emanating from the origins O_i, and attracted to destinations D_j are known, but not the O–D trip matrix. The O–D trip matrix, g_Ω $\Omega = (i,j) \in W$, must satisfy

$$\begin{aligned} \sum_j g_{ij} &= O_i, \ \forall i \\ \sum_i g_{ij} &= D_j, \ \forall j \end{aligned} \tag{12.66}$$

Distribution models of different kinds have been developed to assist in forecasting future trip patterns when important changes in the network take place. They consider a trip making behavior and the way that behavior is influenced by external factors such as total trip ends and traveled distance. These models consider that the number of trips from zones depends on the travel cost between the zones, and the potential of each zone. They use the expression

$$g_{ij} = p(c_{ij}) = \alpha O_i D_j f(c_{ij})$$

where α is a proportionality factor, and $f(c_{ij})$ has one or more parameters to calibrate. This function often receives the name of *deterrence function.* The most common used expressions are

$$\begin{array}{ll} p(c_{ij}) = \exp(-\beta c_{ij}) & \text{exponential function} \\ p(c_{ij}) = c_{ij}^{-n} & \text{power function} \\ p(c_{ij}) = c_{ij}^{n} \exp(-\beta c_{ij}) & \text{combined function} \end{array} \tag{12.67}$$

Assuming that the function $p(c_{ij})$ is a decreasing function of the travel cost, the previous model is formulated as the following mathematical program (TAPD). Minimize

$$Z = \sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} c_{ij}(x)dx - \sum_{\Omega\in W} \int_0^{g_\Omega} p^{-1}(x)dx$$

subject to

$$\sum_j g_{ij} = O_i, \ \forall i \tag{12.68}$$

$$\sum_i g_{ij} = D_j, \ \forall j \tag{12.69}$$

$$\sum_{j\in A(i)} f_{ij}^{\Omega} - \sum_{j\in B(i)} f_{ji}^{\Omega} = r_i^{\Omega}, \ \forall i \in \mathcal{N}, \ \ \forall \Omega \in W \tag{12.70}$$

$$\sum_{\Omega\in W} f_{ij}^{\Omega} = f_{ij}, \ \forall (i,j) \in \mathcal{A} \tag{12.71}$$

$$f_{ij}^{\Omega} \geq 0 \quad \forall \Omega \in W, \ \forall (i,j) \in \mathcal{A} \tag{12.72}$$

where

$$r_i^{\Omega} = \begin{cases} g_\Omega & \text{if } O^\Omega = i \\ -g_\Omega & \text{if } D^\Omega = i \\ 0 & \text{if} i \text{ is an intermediate nodefor the commodity } \Omega \end{cases}$$

The example below illustrates the TAPD model

Example 12.13 (gravity distribution model). We illustrate the model TAPD using the *gravity distribution model*, which is derived using an exponential function for the deterrence function. In this case we obtain

$$\begin{aligned} -\int_0^{g_\Omega} p^{-1}(x)dx &= -\int_0^{g_\Omega} \left[-\frac{1}{\beta} \log(x) + (d_i + d_j + \alpha') \right] dx \\ &= \frac{1}{\beta} g_\Omega \left(\log g_\Omega - 1 \right) + (d_i + d_j + \alpha') g_\Omega \end{aligned}$$

where $d_i = \log O_i/\beta$, $d_j = \log D_j/\beta$, and $\alpha' = \alpha/\beta$. The objective function becomes

$$\sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} c_{ij}(\mathbf{x})d\mathbf{x} + \frac{1}{\beta}\sum_{\Omega\in W} g_\Omega(\log g_\Omega - 1) + \sum_{\Omega\in W}(d_i + d_j + \alpha')g_\Omega$$

The term $\sum_{\Omega\in W}(d_i + d_j + \alpha')g_\Omega$ is constant in the set of feasible solutions defined by (12.66), and it can be dropped. It can be shown that the previous mathematical programming problem is convex, and using the KKT conditions the optimal solution satisfies

$$g^*_{ij} = A_i O_i B_j D_j \exp(-\beta c^*_{ij})$$

where c^*_{ij} is the equilibrium cost. The set of parameters A_i and B_j replace the proportionality factor α to enforce constraints (12.66).

The following GAMS file implements this model for the inputs given in Table 12.17. The results obtained are shown in Table 12.18

```
$title TRAFFIC DISTRIBUTION-ASSIGNMENT PROBLEM.

** Sets are declared in first place.
** Set N is the set of nodes of the road network.
** Set A is the set of links.
** Set W is the set of O--D pairs

SET
 N set of nodes /I1*I13/
 W pairs        /W1*W4/
 O origins      /I1
                 I4/
 D destinations /I2
                 I3/
OW(O,W)
/I1.(W1,W2)
I4.(W3,W4)/
DW(D,W)
/I2.(W1,W3)
I3.(W2,W4)/

A(N,N) set of links
/I1.(I5,I12)
I4.(I5,I9)
I5.(I6,I9)
I6.(I7,I10)
I7.(I8,I11)
I8.(I2)
I9.(I10,I13)
I10.(I11)
I11.(I2,I3)
I12.(I6,I8)
I13.(I3)/

ODE(N,W) set of origins by demand
/I1.W1
```

```
I1.W2
I4.W3
I4.W4/
DDE(N,W) set of destinations by demand
/I2.W1
I3.W2
I2.W3
I3.W4/;

ALIAS(N,I)
ALIAS(N,J)

PARAMETER
Oi(O) number of trips emanating from O
/I1 1200
I4 800/

Dj(D) number of trips attracted to D
/I2 1000
I3 1000/

BETA;
BETA=0.2;

TABLE  CDATA(N,N,*)  link cost parameters
          CO      D       M
 I1.I5    7.0  0.00625   2.0
 I1.I12   9.0  0.005     2.0
 I4.I5    9.0  0.005     2.0
 I4.I9   12.0  0.0025    2.0
 I5.I6    3.0  0.00375   2.0
 I5.I9    9.0  0.00375   2.0
 I6.I7    5.0  0.00625   2.0
 I6.I10  13.0  0.0025    2.0
 I7.I8    5.0  0.00625   2.0
 I7.I11   9.0  0.00625   2.0
 I8.I2    9.0  0.00625   2.0
 I9.I10  10.0  0.0025    2.0
 I9.I13   9.0  0.0025    2.0
 I10.I11  6.0  0.00125   2.0
 I11.I2   9.0  0.0025    2.0
 I11.I3   8.0  0.005     2.0
 I12.I6   7.0  0.00125   2.0
 I12.I8  14.0  0.005     2.0
 I13.I3  11.0  0.005     2.0;

** Optimization variables are declared.

VARIABLES
 z         objective function variable
 f(N,N)    flow link
 fc(W,N,N) is the flow link of the commodity W
 cos(N,N)  cost in the link at equilibrium
 g(W)      demand in the pair W;

POSITIVE VARIABLE fc(W,N,N);
```

```
g.LO(W)=0.01;

** Constraints are declared.

EQUATIONS
 COST Objective function
 BALANCE(W,I) conservation of flow condition for the commodity W.
 FLOW(I,J) total link flow
 ORIGIN(O)
 DESTIN(D)
 COSTE(I,J) cost at equilibrium;

COST .. z=E= SUM((I,J)$A(I,J),CDATA(I,J,'CO')*f(I,J)
             +CDATA(I,J,'D')*f(I,J)**CDATA(I,J,'M'))
             +(1/BETA)*SUM(W,g(W)*(LOG(g(W))-1 ));
BALANCE(W,I) .. SUM(J$A(I,J),fc(W,I,J))-SUM(J$A(J,I),fc(W,J,I))
              =E=g(W)$ODE(I,W)-g(W)$DDE(I,W);
FLOW(I,J)$A(I,J)..SUM(W,fc(W,I,J))=E=f(I,J);
COSTE(I,J)$A(I,J)..cos(I,J)=E=CDATA(I,J,'CO')
                           +2*CDATA(I,J,'D')*f(I,J);
ORIGIN(O)..Oi(O)=E= SUM(W,g(W)$OW(O,W));
DESTIN(D)..Dj(D)=E= SUM(W,g(W)$DW(D,W));
MODEL nd /ALL/;
SOLVE nd USING nlp MINIMIZING z;
```

Table 12.17: Inputs for TAPD

Origin	O_i	Destination	D_j	β
1	1200	2	1000	0.2
4	800	3	1000	–

Table 12.18: Outputs for TAPD

O–D pair	g_Ω	Equilibrium cost
Ω_1	650.312	52.21
Ω_2	549.688	53.65
Ω_3	349.688	46.77
Ω_4	450.312	45.05

■

12.7 Short-Term Hydrothermal Coordination

Short-term hydrothermal coordination (STHTC) determines the startup and shutdown of thermal plants, as well as the power output of hydro and thermal plants to meet customer demand with an appropriate level of security and so that total operating costs are minimized. Because of the high cost associated with the startup of thermal plants, selecting the plants to meet customer demand in the most economical manner can save large amounts of money. If the electric energy system under consideration does not include hydroelectric plants, the above problem is called unit commitment.

Mathematically, the STHTC problem can be formulated as a mixed-integer nonlinear optimization problem. For realistic size electric energy systems it is also a large-scale problem. Solving this large-scale nonlinear and combinatorial optimization problem is not an easy task. Lagrangian relaxation (LR) techniques are the most suitable techniques to solve this kind of problems (see Muckstadt and Koening [77], Merlin and Sandrin [74], Bertsekas et al. [12], Zhuang and Galiana [107], Yan et al. [106], Mendes et al. [73], Rakic and Marcovic [91], Wang et al. [101], Pellegrino et al. [85], Luh et al.[69], Jiménez and Conejo [60]). Dynamic programming techniques require discretization of continuous variables and drastic simplifying assumptions to make the problem computationally tractable (Hobbs et al. [52]). Mixed-integer linear programming techniques not only linearize the problem but also make important simplifications to be able to solve such a large-scale problem (see Dillon et al. [32], Brannlund et al. [16], Medina et al. [70]).

When using LR techniques to solve the STHTC problem, the resulting relaxed primal problem can be naturally decomposed into one subproblem per thermal plant and one subproblem per hydro system. Therefore, by using LR techniques, the solution of the STHTC problem (large-scale and complex optimization problem) is accomplished by the solution of many small sized and structurally homogeneous subproblems.

This decomposition property allows a very precise modeling of each generating plant as well as the possibility of applying to each subproblem the most suitable optimization technique to its structure. It also allows the natural application of parallel computing with the corresponding advantages regarding CPU time.

In addition to all these advantages, derived from the decomposition property of the relaxed primal problem, the application of LR techniques to solve the STHTC problem presents another important advantage: the dual problem variables (the Lagrange multipliers) have an economical meaning that can be very helpful in the framework of deregulated electric energy markets, and also in the traditional framework of centralized systems.

12.7.1 Problem Formulation and the LR Solution Procedure

The STHTC problem can be formulated as a nonlinear and combinatorial optimization problem in which total operating costs are minimized subject to meeting constraints modeling the technical limitations of thermal and hydro plants, and to meet load constraints. Load constraints include electric energy customer demand constraints plus spinning reserve constraints. Spinning reserve constraints ensure an appropriate level of security.

The main elements of this problem are

1. **Data.**

I:	the number of thermal plants
J:	the number of hydro systems
$\mathbf{H}$:	the vector of demands
$\mathbf{h}_i(\mathbf{x}_i)$:	the contribution of thermal unit i to meet the demand
$\mathbf{h}_j(\mathbf{x}_j)$:	the contribution of hydro system j to meet the demand
$\mathbf{G}$:	the vector of power reserves
$\mathbf{g}_i(\mathbf{x}_i)$:	the contribution of thermal unit i to meet the power reserve
$\mathbf{g}_j(\mathbf{x}_j)$:	the contribution of hydro system j to meet the power reserve

   $\mathbf{H}$, $\mathbf{h}_i(\mathbf{x}_i)$, $\mathbf{h}_j(\mathbf{x}_j)$, $\mathbf{G}$, $\mathbf{g}_i(\mathbf{x}_i)$ and $\mathbf{g}_j(\mathbf{x}_j)$ are vectors of dimension equal to the number of time periods in the planning horizon.

2. **Variables.**

$\mathbf{x}_i$:	the vector of variables associated with thermal plant i
$\mathbf{x}_j$:	the vector of variables associated with hydro system j

3. **Constraints.**

$$\begin{array}{rcl} \mathbf{s}_i(\mathbf{x}_i) & \leq & \mathbf{0}, i = 1, \dots, I \\ \mathbf{s}_j(\mathbf{x}_j) & \leq & \mathbf{0}, j = 1, \dots, J \\ \sum\limits_{i=1}^{I} \mathbf{h}_i(\mathbf{x}_i) + \sum\limits_{j=1}^{J} \mathbf{h}_j(\mathbf{x}_j) & = & \mathbf{H} \\ \sum\limits_{i=1}^{I} \mathbf{g}_i(\mathbf{x}_i) + \sum\limits_{j=1}^{J} \mathbf{g}_j(\mathbf{x}_j) & \leq & \mathbf{G} \end{array} \tag{12.73}$$

   The first set of constraints expresses thermal plant constraints, the second one represents hydro system constraints, the third one expresses customer demand constraints, and the fourth one represents spinning reserve constraints. It should be noted that time is embedded in the formulation shown above.

4. **Function to be optimized.** The objective function represents the total operating cost (the cost of hydro power production is negligible compared to the cost of thermal power production)

$$f(\mathbf{x}) = \sum_{i=1}^{I} f_i(\mathbf{x}_i) \tag{12.74}$$

This problem, referred to as the *primal problem* (PP), is formulated as follows

$$\min_{\mathbf{x}=(\mathbf{x}_i,\mathbf{x}_j)} f(\mathbf{x}) = \sum_{i=1}^{I} f_i(\mathbf{x}_i) \tag{12.75}$$

subject to

$$\begin{array}{rcl} \mathbf{s}_i(\mathbf{x}_i) & \leq & \mathbf{0}, i = 1, \dots, I \\ \mathbf{s}_j(\mathbf{x}_j) & \leq & \mathbf{0}, j = 1, \dots, J \\ \sum\limits_{i=1}^{I} \mathbf{h}_i(\mathbf{x}_i) + \sum\limits_{j=1}^{J} \mathbf{h}_j(\mathbf{x}_j) & = & \mathbf{H} \\ \sum\limits_{i=1}^{I} \mathbf{g}_i(\mathbf{x}_i) + \sum\limits_{j=1}^{J} \mathbf{g}_j(\mathbf{x}_j) & \leq & \mathbf{G} \end{array} \tag{12.76}$$

Load constraints are the complicating or global constraints of this primal problem. Load constraints include equality constraints (demand constraints) and inequality constraints (spinning reserve constraints). They couple together decisions related to thermal and hydro plants. Because of the existence of these constraints, the preceding problem cannot be decomposed and cannot be easily solved.

By applying LR techniques, load constraints are incorporated into the objective function to form the relaxed primal problem. The vector of multipliers associated with the vector of demand constraints is called $\boldsymbol{\lambda}$ and the vector of multipliers associated to the spinning reserve constraints is called $\boldsymbol{\mu}$.

The Lagrangian function is defined as

$$\begin{array}{rcl} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) & = & \sum\limits_{i=1}^{I} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T \left(\mathbf{H} - \sum\limits_{i=1}^{I} \mathbf{h}_i(\mathbf{x}_i) - \sum\limits_{j=1}^{J} \mathbf{h}_j(\mathbf{x}_j) \right) \\ & & + \boldsymbol{\mu}^T \left(\mathbf{G} - \sum\limits_{i=1}^{I} \mathbf{g}_i(\mathbf{x}_i) - \sum\limits_{j=1}^{J} \mathbf{g}_j(\mathbf{x}_j) \right) \end{array} \tag{12.77}$$

and the dual function is the solution of the problem

$$\theta(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \min_{(\mathbf{x}_i,\mathbf{x}_j)} \mathcal{L}(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{12.78}$$

subject to

$$\begin{array}{rcl} \mathbf{s}_i(\mathbf{x}_i) & \leq & \mathbf{0}; \;\; i = 1, 2, \dots, I \\ \mathbf{s}_j(\mathbf{x}_j) & \leq & \mathbf{0}; \;\; j = 1, 2, \dots, J \end{array} \tag{12.79}$$

which can be expressed as

$$\theta(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \boldsymbol{\lambda}^T \mathbf{H} + \boldsymbol{\mu}^T \mathbf{G} + d(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{12.80}$$

where $d(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is the solution of the following optimization problem

$$\min_{\mathbf{x}_i,\mathbf{x}_j} \left(\sum_{i=1}^{I} \left(f_i(\mathbf{x}_i) - \boldsymbol{\lambda}^T \mathbf{h}_i(\mathbf{x}_i) - \boldsymbol{\mu}^T \mathbf{g}_i(\mathbf{x}_i) \right) - \sum_{j=1}^{J} \left(\boldsymbol{\lambda}^T \mathbf{h}_j(\mathbf{x}_j) + \boldsymbol{\mu}^T \mathbf{g}_j(\mathbf{x}_j) \right) \right) \tag{12.81}$$

subject to

$$\begin{aligned}\mathbf{s}_i(\mathbf{x}_i) &\leq \mathbf{0}, i = 1, \dots, I \\ \mathbf{s}_j(\mathbf{x}_j) &\leq \mathbf{0}, j = 1, \dots, J.\end{aligned} \tag{12.82}$$

This problem can be naturally decomposed into one subproblem per thermal plant i and one subproblem per hydro system j. For fixed values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, problem (12.78) is called the *relaxed primal problem*, and problem (12.81) is called the *decomposed primal problem*.

The subproblem associated with thermal plant i is

$$\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) - \boldsymbol{\lambda}^T \mathbf{h}_i(\mathbf{x}_i) - \boldsymbol{\mu}^T \mathbf{g}_i(\mathbf{x}_i) \tag{12.83}$$

subject to

$$\mathbf{s}_i(\mathbf{x}_i) \leq \mathbf{0} \tag{12.84}$$

and the subproblem associated with hydro system j is

$$\max_{\mathbf{x}_j} \boldsymbol{\lambda}^T \mathbf{h}_j(\mathbf{x}_j) + \boldsymbol{\mu}^T \mathbf{g}_j(\mathbf{x}_j) \tag{12.85}$$

subject to

$$\mathbf{s}_j(\mathbf{x}_j) \leq \mathbf{0} \tag{12.86}$$

LR techniques are based on the solution of the following dual problem [as problem (9.64)]

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \theta(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{12.87}$$

subject to

$$\boldsymbol{\mu} \geq \mathbf{0} \tag{12.88}$$

Because of the existence of integer variables (e.g., thermal plant unit commitment variables) in the formulation of the primal problem, the STHTC is a non-convex problem. Therefore, the optimal solution of the dual problem is not the optimal solution of the primal problem but it is a lower bound. Nevertheless, as the size of the problem increases the per unit duality gap (see Section 1.1) decreases (see Ferreira [37], Bertsekas et al. [12], Everett [34]), therefore the optimal solution of the dual problem becomes closer to the optimal solution of the primal problem. Once the optimal solution of the dual problem is found, heuristic procedures can be easily applied to derive a near-optimal primal problem solution (Zhuang and Galiana [107]).

In the STHTC problem, inequality complicating constraints are closely related to the integer variables (the thermal plant unit commitment variables) and the equality constraints are closely related to the continuous variables (thermal and hydro plant power output variables). This motivates the decomposition of phase 2 into two consecutive phases. In the first one, called *phase 2A*, the solution of the dual problem (or phase 1) is slightly modified to find values of the integer variables that meet inequality global constraints (spinning reserve constraints). In the second one, called *phase 2B*, the solution of phase 2A is modified by adjusting the values of the continuous variables to meet equality global constraints (demand constraints).

Therefore, the LR procedure to solve the STHTC problem consists of

Phase 1. Solution of the dual problem.

Phase 2A. Search for a spinning reserve primal feasible solution.

Phase 2B. Search for a load balanced primal feasible solution: multiperiod economic dispatch.

The solution of the dual problem (phase 1) is the key element to solve the STHTC problem by LR. The efficiency of a STHTC algorithm relies on the efficiency of the solution of this phase.

Phase 2A is an iterative procedure in which the $\boldsymbol{\mu}$ multipliers are updated in those periods where spinning reserve constraints are not satisfied. In these periods, the corresponding $\boldsymbol{\mu}$ multipliers are increased proportionally to the mismatches in spinning reserve constraints (subgradient type updating) until these constraints are met in all the time periods of the planning horizon. At the end of phase 2A a primal set of feasible commitment decisions is found. This phase requires typically little CPU time to achieve a solution very close to the solution of phase 1. phase 2B is a multiperiod economic dispatch procedure (Wood and Wollenberg [104]) in which, once unit commitment variables are set to the solution of phase 2A, power output is adjusted in order to meet the demand constraints in all time periods. This is a traditional problem routinely solved by electric energy system operators.

12.7.2 Dual-Problem Solution: Multiplier Updating Techniques

The procedure to solve the dual problem of the STHTC problem was described earlier. Specifically, at each iteration the relaxed primal problem is solved and, with the information obtained, the multiplier vector is updated. The information derived from the resolution of the relaxed primal problem is

- The value of the dual function

$$\theta(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}) = \boldsymbol{\lambda}^{(t)T}\mathbf{H} + \boldsymbol{\mu}^{(t)T}\mathbf{G} + d(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}) \tag{12.89}$$

  which is

$$\begin{aligned} d(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}) = & \sum_{i=1}^{I}\left(f_i(\mathbf{x}_i^{*(t)}) - \boldsymbol{\lambda}^{(t)T}\mathbf{h}_i(\mathbf{x}_i^{*(t)}) - \boldsymbol{\mu}^{(t)T}\mathbf{g}_i(\mathbf{x}_i^{*(t)})\right) \\ & - \sum_{j=1}^{J}\left(\boldsymbol{\lambda}^{(t)T}\mathbf{h}_j(\mathbf{x}_j^{*(t)}) + \boldsymbol{\mu}^{(t)T}\mathbf{g}_j(\mathbf{x}_j^{*(t)})\right) \end{aligned} \tag{12.90}$$

where $\mathbf{x}_i^{*(t)}$ is the vector of optimal values for the variables associated with thermal plant i and $\mathbf{x}_j^{*(t)}$ is the vector of optimal values for the variables associated with hydro system j obtained from the solution of the relaxed primal problem

- A subgradient $\mathbf{s}^{(t)}$ of the dual function at the optimal solution of the relaxed primal problem: $\mathbf{x}_i^*, \forall i$, $\mathbf{x}_j^*, \forall j$

A subgradient can be easily computed as the vector of mismatches in demand constraints and the vector of mismatches in spinning reserve constraints:

$$\mathbf{s}^{(t)} = \text{column}[\mathbf{h}^{(t)}, \mathbf{g}^{(t)}] \tag{12.91}$$

where

$$\begin{aligned} \mathbf{h}^{(t)} &= \mathbf{H} - \sum_{i=1}^{I} \mathbf{h}_i(\mathbf{x}_i^{*(t)}) - \sum_{j=1}^{J} \mathbf{h}_j(\mathbf{x}_j^{*(t)}) \\ \mathbf{g}^{(t)} &= \mathbf{G} - \sum_{i=1}^{I} \mathbf{g}_i(\mathbf{x}_i^{*(t)}) - \sum_{j=1}^{J} \mathbf{g}_j(\mathbf{x}_j^{*(t)}) \end{aligned} \tag{12.92}$$

Although the four methods previously described can be applied to update the multipliers, the most suitable methods for the STHTC problem are the bundle method Pellegrino et al. [85]) and the dynamically constrained cutting plane method (Jiménez and Conejo [60]).

12.7.3 Economical Meaning of the Multipliers

As it has been indicated above, one of the advantages of the use of LR techniques is the availability of the useful economical information provided by the variables of the dual problem, the Lagrange multipliers.

Multiplier $\boldsymbol{\lambda}$ at a given time represents, from the point of view of the system, the cost of producing one extra unit of electric energy [megawatt-hour (MWh)], that is, the electric energy marginal cost. Equivalently, from the point of view of a generating company, multiplier $\boldsymbol{\lambda}$ at a given time represents an indicator of the price a plant should be paid for each MWh of energy. It also represents an indicator of the price that a generating company should bid to get its plant online.

Analogously, multiplier $\boldsymbol{\mu}$ at a given time represents the cost of keeping an incremental unit (MW) of power reserve or equivalently, an indicator of the price a generator should be paid for each MW of reserve.

This economical interpretation is useful in the traditional framework of centralized electric energy systems to elaborate electric tariffs, but also in the framework of modern deregulated electric energy markets.

In the framework of deregulated electric energy markets, the LR procedure to solve the STHTC problem can be interpreted as the actual functioning of a free market. In other words, it can be thought of as a mechanism to meet customer demand with an appropriate level of security (measured in terms of the spinning reserve) by choosing the cheapest generator offers.

Hourly energy prices proposals (Lagrange multipliers) are specified by the market operator for the planning horizon. Each generator (or generating company) schedules its production independently along the planning horizon to maximize its benefit (i.e., each generator solves an optimization problem). Analogously, each hydro system is scheduled so that its benefit is maximum (i.e., each hydro system solves a problem). After the submissions of production proposals by all generators the demand equation is evaluated in each hour of the planning horizon. Hourly prices are updated by the market operator with any of the techniques stated above, and the previous procedure is repeated until the demand is satisfied. This mechanism constitutes a competitive energy market. Similarly, a spinning reserve market can be established.

It should be noted that by applying the LR technique to solve the STHTC problem, each generator schedules its production attending only to prices of energy and prices of reserve. The interchange of information between the market operator and the generators is clear and concise. The resulting market is therefore economically efficient and transparent.

The Augmented Lagrangian decomposition technique has also been applied to the STHTC problem. Relevant information can be found in Batut et al. [7], Batut and Renaud [8], Renaud [94], and Wang et al. [101].