CHAPTER 3

Partial Equilibrium in Mathematica

It is customary to begin the study of microeconomics with market behavior in a partial equilibrium setting. This is done by analyzing the determination of price and quantity in a single competitive market under the assumption that all other influences from the rest of the economy remain constant. Such a study usually begins with the theory of the consumer and the derivation of demand curves and then proceeds to the theory of the firm and the derivation of supply curves. Then demand and supply are brought together to study market equilibrium. This is the standard approach we follow here. We are interested primarily in the derivation of analytical results and graphical representations, for which Mathematica, owing to its power to deal with symbolic mathematics problems and to its plotting capabilities, is a very useful tool.

3.1. UTILITY AND PRODUCTION FUNCTIONS

The starting point of consumer theory is the specification of preferences and their representations by means of a utility function, whereas the starting point of the theory of the firm is the specification of technology and its representation by means of a production function. While many theoretical results are derived for very general forms of those functions, in most examples and in applications we commonly work with a few functional specifications. Leontief and Cobb-Douglas functions are probably the most popular, and they can be used to represent preferences or technology. We present both of them and focus on the two-good case since this is the one that can be easily handled in graphical representations, although the results can be generalized to more goods and displayed analytically.

3.1.1. Leontief Function

A Leontief function for a two-good case is

$$f(x_1, x_2) = \min(a_1 x_1, a_2 x_2) \tag{1}$$

where f is the function, a_1 and a_2 are parameters, and x_1 and x_2 are interpreted as goods consumed, if we use the function to represent preferences as in consumer's theory. Alternatively, they may be interpreted as inputs if we use the function to represent technology as in the theory of the firm. This function specifies that no substitution is possible between goods or between inputs. The consumer always spends all of his or her income in fixed proportions between the two goods, and similar behavior is displayed by the firm in connection with its inputs. As we will see later, this implies a peculiar form for the consumer's indifference curves and for the firm's production isoquants.

The graphical representation of such a function in Mathematica is straightforward and it is available in the `Leontief.nb` file on the book web site. If you are already familiar with Mathematica you can begin with that notebook file, but if you are a first-time user, we recommend that you type in the commands. The instructions for running Mathematica are in Appendix B at the end of this book.

We begin by assigning values to the parameters a_1 and a_2. In this case we give them both the value 1. Start Mathematica and on the Untitled-1 window that opens type

```
a1 = 1
```

followed by `Return` and then

```
a2 = 1
```

followed by `Shift-Enter`. Mathematica acts as an interpreter and commands are processed one at a time. Using `Return` at the end of the line in effect asks Mathematica to postpone the processing while you enter another command on the next line, whereas `Shift-Enter` at the end of the line asks the program to process all of the input since the last `Shift-Enter`. Mathematica then responds by converting your input to

```
IN[1]:=   a1 = 1
          a2 = 1
```

The symbols `IN[]:=` in Mathematica denote input and the other expressions are the input to be evaluated. The output statements corresponding to the input are

```
Out[1]:=  1
Out[2]:=  1
```

Thus Mathematica displays the result of the assignments as output. Note that separate output is generated for each statement, regardless of whether we write the inputs in a single input prompt or in separate ones. Note also the sequential numbering of inputs and outputs.

The outputs of the previous evaluations are quite simple and redundant. To avoid the display of output, we could have added a semicolon, `;`, at the end of the statement whose output we wanted to suppress.

Next we assign to the variable Leontief the corresponding Mathematica function `Min[]` that yields the numerically smallest of its arguments:

```
IN[3]:=    Leontief = Min[a1 x1, a2 x2]
```

In Mathematica two symbols can be multiplied either by using the asterisk operator as `a1*x1` or simply by juxtaposing the two symbols with a space between them as `a1 x1`. When you finish typing the line above be sure to strike `Shift-Enter`. This yields the output

```
OUT[3]:=  Min[x1,x2]
```

Note that Mathematica replaced the parameters `a1` and `a2` with their numerical values of 1 while keeping everything else the same, since, for the time being, the evaluation of the statement cannot be carried out beyond this point.

Next we ask Mathematica to generate a three-dimensional plot of the function within given numerical intervals for x_1 and x_2 using the Mathematica function `Plot3D[f,{x,xmin,xmax},{y,ymin,ymax}]`, where `f` is the function to be plotted over the variables `x` and `y` between their specified minimum and maximum values. Type

```
IN[6]:=    Plot3D[Leontief,{x1,0,1},{x2,0,1}]
```

Be careful not to misspell Leontief or the program will give you more error messages than you care to see. Also, be sure to end the line with `Shift-Enter`. The resulting graph is shown in Figure 3.1.

Finally, with the statement

```
IN[8]:=    ContourPlot[Leontief,{x1,0,1},{x2,0,1}]
```

we obtain the contour plot of the Leontief function shown in Figure 3.2, which illustrates the consumer's indifference curves or, equivalently, the firm's isoquants. Contour plots produced by Mathematica are by default shaded, and regions with higher functional values are lighter. Contour curves for the Leontief function form 90° angles. Note that the graph shows the kinks with some error as we get farther away from the origin.

Whenever you run a program in Mathematica it is important that you wipe out any previous values associated with the parameters and variables of the problem.

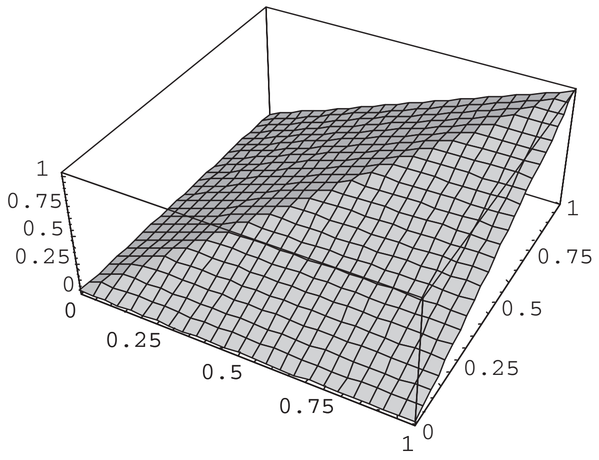

Figure 3.1. Leontief function.

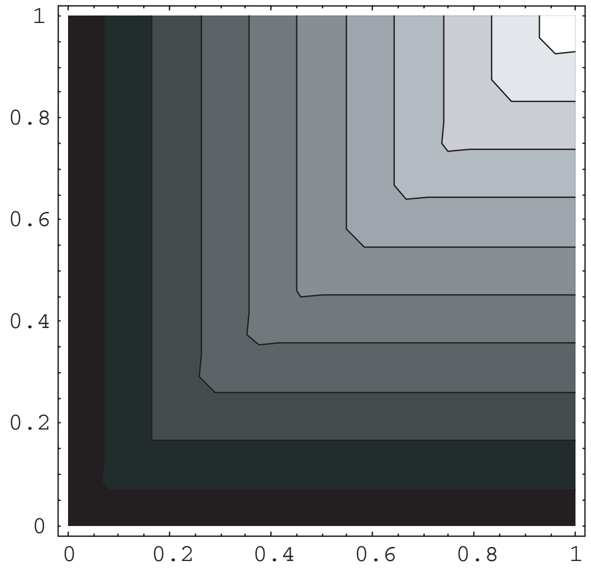

Figure 3.2. Leontief function contour lines.

This can be achieved by adding the following statement at the beginning of the program:

```
IN[]:=     Clear[a1,a2,x1,x2,Leontief];
```

3.1.2. Cobb-Douglas Function

A Cobb-Douglas function with constant returns to scale (we use a special case) is

$$f(x_1, x_2) = x_1{}^{\rho} x_2^{1-\rho} \tag{2}$$

where f is the function, x_1 and x_2 are goods or inputs, and ρ is a parameter. In consumer theory ρ and 1– ρ represent the consumer's expenditure shares on each good. In the theory of the firm, the fact that the two exponents of the inputs add up to one implies that the technology the function represents displays constant returns-to-scale. Unlike the Leontief function, the Cobb-Douglas function allows for smooth substitution between goods or between inputs.

The Mathematica statements corresponding to the graphical representation of the Cobb-Douglas function are shown below and are available in the `CobbDouglas.nb` file on the book web site. This time we recommend that you open the input file and use it to follow the discussion. When you open the notebook file you see a bunch of brackets on the right-hand side of the window. You can execute the program by selecting these brackets and striking `Shift-Enter`. For example, selecting the bracket opposite the lines

```
Clear[x1,x2,ρ];
ρ = 0.7;
CD = x1^ρ x2^(1-ρ);
Plot3D[CD,{x1,0,1},{x2,0,1}]
ContourPlot[CD,{x1,0,1},{x2,0,1}]
```

and striking `Shift-Enter` causes the lines to be processed and results in their being reprinted as

```
In[1]:=
      Clear[x1,x2,ρ];
      ρ = 0.7;
      CD = x1^ρ x2^(1-ρ);
      Plot3D[CD,{x1,0,1},{x2,0,1}]
      ContourPlot[CD,{x1,0,1},{x2,0,1}]
```

with input prompt `In[1]:=` now showing. In this way you can use the notebook file to modify the input and rerun the program. For example, you might change ρ from 0.7 to 0.8, select the bracket to its right, and type `Shift-Enter`. Be aware, however, that only that part of the program covered by the bracket you select is rerun. Therefore, if you want to redo the plots you must select one of the more inclusive brackets on the right before striking `Shift-Enter`.

The foregoing statements follow the pattern presented in the previous section. We named the function `CD` and assigned a value of 0.7 to the ρ parameter. Unlike the

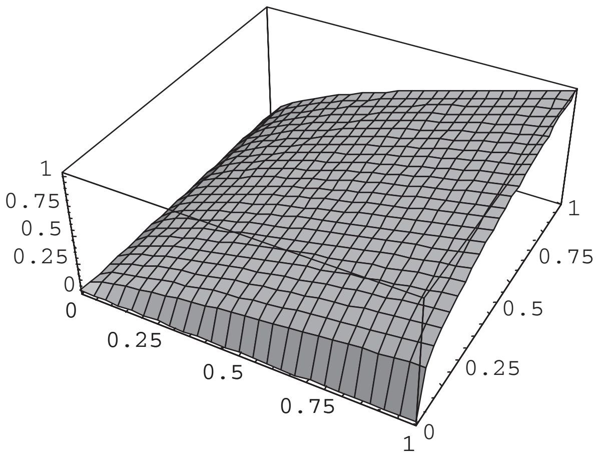

Figure 3.3. Cobb-Douglas function.

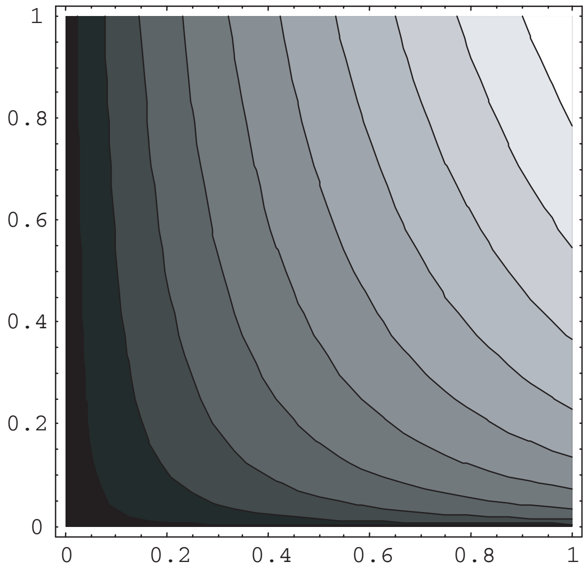

Figure 3.4. Cobb-Douglas function contour lines.

program for the Leontief function, here we put all the statements together in one input prompt and suppressed output using semicolons at the end of the first three statements. Note that Mathematica allows you to enter Greek letter symbols such as ρ. To do so, and also to enter formulas in a mathematical form instead of the text form we used here, you have to use a palette you can access from the `File/Palettes/BasicInput` main menu option.

Figures 3.3 and 3.4 show the corresponding three-dimensional and contour graphs. If you are following along with Mathematica, you might close all the files you have opened so far to reduce the clutter on your computer desktop and give yourself a fresh start in the next section.

3.2. CONSUMER THEORY

The standard theory of consumer's behavior poses the problem faced by the consumer as one of maximizing utility subject to a budget constraint. That is, given a bundle of goods, their prices, and a certain amount of income, the consumer buys those goods according to her preferences while trying to maximize her utility, a quantity that is supposed to measure the level of consumer satisfaction.

In formal terms, and for a two-good example that can be easily generalized, the problem can be stated as

$$\begin{aligned}&\text{max}\quad u(x_1, x_2)\\&\text{subject to}\quad p_1x_1 + p_2x_2 = m\end{aligned} \tag{3}$$

where u is the utility function, x_1 and x_2 are goods, p_1 and p_2 are prices, and m is income.

From here on we work with a Cobb-Douglas function. Thus, using Eq. (2) we can restate the foregoing problem as

$$\begin{aligned}&\text{max}\quad u = x_1^{\rho}\, x_2^{1-\rho}\\&\text{subject to}\quad p_1x_1 + p_2x_2 = m\end{aligned} \tag{4}$$

An equivalent but simpler expression for the utility function is obtained by taking logs:

$$\log \mathrm{u} = \rho \log(x_1) + (1 - \rho) \log(x_2) \tag{5}$$

We start the Mathematica program of the consumer's problem—available in the `Consumer.nb` file—by inputting the utility function

```
In[]:=     logu = ρ Log[x1] + (1-ρ) Log[x2];
```

and the budget constraint

```
In[]:=     bc = m - (p1 x1 + p2 x2);
```

Note that we give a name to the budget constraint, i.e., `bc`, and then assign all its elements. We will soon see the usefulness of that.

The next step is to form the Lagrangian corresponding to the maximization problem. Thus we write

```
In[]:=     eqL = L == logu + λ bc
```

assigning the expression

```
    L == logu + λ bc
```

to the variable `eqL`. The presence of the double equal symbol == indicates that the expression is an equation, not an assignment to the variable `L`. The corresponding output is the content of the variable `eqL` with the expressions for `logu` and `bc` replaced by their definitions[1]:

```
    Out[]= L == (m - p1 x1 - p2 x2) λ + ρ Log[x1]+(1-ρ)Log[x2]
```

If instead of writing the budget constraint in the way we did previously, we write it in a more standard way, that is,

```
    In[]:=     m = p1 x1 + p2 x2;
```

to later write the Lagrangian as

```
    In[]:=     eqL = L == logu + λ (m - p1 x1 - p2 x2)
```

the output generated by Mathematica would be

```
    Out[]=     L == ρ Log[x1] + (1 - ρ) Log[x2]
```

Indeed, when evaluating the part of the input expression corresponding to (`m - p1 x1 - p2 x2`), Mathematica replaces the variable `m` with its definition. Then this part of the expression becomes (`p1 x1 + p2 x2 - p1 x1 - p2 x2`). Thus, it would be equal to zero. It was to avoid this kind of problem that we defined the variable bc in the way we did.

Once we form the Lagrangian, we compute the first-order conditions of the problem as follows:

```
    In[]:=     foc1 = D[eqL, x1]
               foc2 = D[eqL, x2]
               foc3 = D[eqL, λ]
```

The Mathematica function `D` computes the partial derivatives of a function. In this case, we ask Mathematica to compute the partial derivatives of the expression `eqL` w.r.t. (with respect to) the variable of choice. The corresponding outputs are

1. It is common in Lagrangian functions to put the objective term first followed by the λ and the constraint, but given the sequence of commands we used, Mathematica does things in reverse order. This causes no problem except for making the output below slightly harder to comprehend at first.

```
Out[]=      0 == -p1λ+ρ/x1

Out[]=      0 == -p2λ+(1 - ρ)/x2

Out[]=      0 == m - p1 x1 - p2 x2
```

Using the Mathematica function `Solve,` we can obtain the goods' demand functions from the system of equations formed by the first-order conditions. Within this function, we first have to specify the equations and then the variables over which they are solved:

```
In[]:=      Solve[{foc1,foc2,foc3},{x1,x2,λ}]
```

The previous statement generates the output

```
Out[]=      {{λ → 1/m, x1 → mρ/p1, x2 → (m - mρ)/p2}}
```

Finally, we want to plot the goods' demand functions. Since the standard procedure is to plot quantities on the horizontal axis and prices on the vertical axis, we have to solve the demand functions for the corresponding prices. Starting with good 1, the Mathematica statements are:

```
In[]:=      p1 = ρ m / x1;
            Plot[p1 /. {ρ → 0.7, m → 0.1},
            {x1,0.01,0.1},
            AxesLabel → {"x1", "p1"},
            PlotLabel → "Demand Curve for x1"]
```

We use the replacement operator `/.` in the first line of the `Plot[]` function. This operator, whose general form is "`expression /. Rules`" applies a rule or list of rules in an attempt to transform each subpart of an expression. In our case the transformation rules are `ρ → 0.7` and `m → 0.1,` which are used to give particular values to the parameters ρ and `m`. To write the arrows, you must type -> as a pair of characters, with no space in between.

The second line of the `Plot` function contains the specification of the range for the horizontal axis, writing first the name of the corresponding variable and then the minimum and the maximum values for the plot. Finally, the last two lines label the axes and assign a label to the plot by means of the options `AxesLabel` and `PlotLabel`. The plot generated is shown in Figure 3.5.

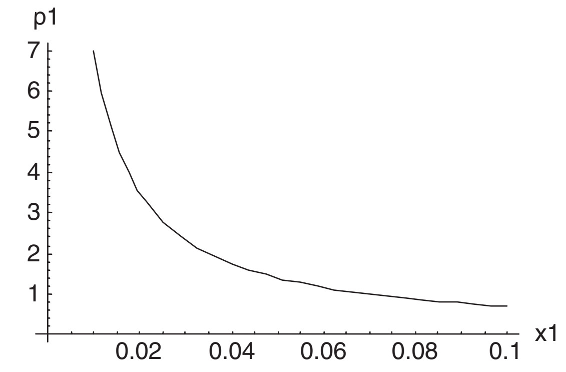

Figure 3.5. Demand curve for x1.

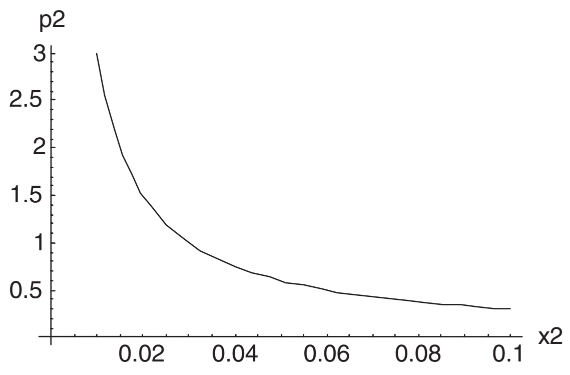

Figure 3.6. Demand curve for x2.

In an analogous way, we generate a plot for the demand function of good x2, shown in Figure 3.6:

```
In[]:=      p2 = (m - ρ m) / x2;
            Plot[p2 /. {ρ → 0.7, m → 0.1},
            {x1,0.01,0.1},
            AxesLabel → {"x2", "p2"},
            PlotLabel → "Demand Curve for x2"]
```

3.3. THE THEORY OF THE FIRM

The standard theory of firm behavior assumes that the main goal of a firm is to maximize profits given technology and the prices of output and inputs. To develop a simple example, let us assume that the firm produces a single output x_1 with price p_1, using labor L as a single input whose price is the wage w. Let us also assume that the production function is of the form TL^b, where T and b are parameters and let us denote profits by π.

In formal terms the problem of the firm can be stated as

$$\begin{aligned} &\max \quad \pi = p_1 x_1 - wL \\ &\text{subject to} \quad x_1 = TL^b \end{aligned} \tag{6}$$

Substituting the production function into the profit function, we obtain the first input for the Mathematica representation of the problem—available in the `Firm.nb` file—as

```
In[]:=     pi = p1 T L^b - w L;
```

Note that we write `pi` instead of π since the Greek letter π is a reserved symbol in Mathematica.

Next we solve the first-order condition of the problem for `L`. By means of the `D[ ]` function we compute the partial derivative of the profit function w.r.t. the variable labor and set the result equal to zero. Finally, we nest this operation within a `Solve[ ]` function:

```
In[]:=     Solve[D[pi,L]==0,L]
```

The resulting output is the labor demand function

Out[]= $\left\{L \to \left(\frac{w}{b\ p1\ T}\right)^{\frac{1}{-1+b}}\right\}$

Next we assign the expression for the labor demand function to the temporary variable `tempL` using the replacement operator `/.`. The `%` symbol in the following statement refers to the last result generated, and `[[1]]` refers to the first solution from the output list, which in this case contains only one solution. Thus, `tempL` is equal to `L`, where `L` is replaced by the solution generated in the previous output line.

```
In[]:=     tempL = L /.%[[1]]
```

Substituting `tempL`—that is, the labor demand function—into the production function in Eq. (6), we obtain the supply function for `x1` that we assign to the temporary variable `tempx1`:

```
In[]:=     tempx1 = T tempL^b
```

The resulting output is

Out[]= $T\left(\left(\frac{w}{b\ p1\ T}\right)^{\frac{1}{-1+b}}\right)^{b}$

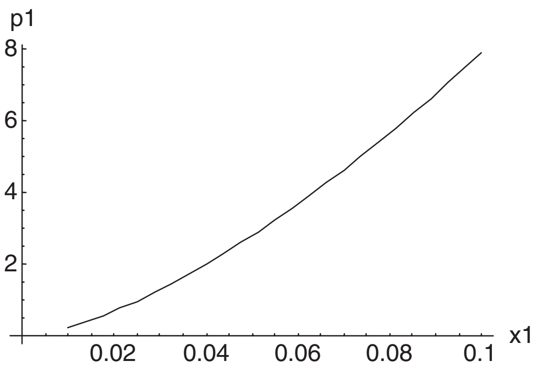

Figure 3.7. Supply curve for `x1`.

Having obtained the good supply and the labor demand functions, we want to plot them in the standard way, that is, with price and wage on the vertical axis in the respective plots. We begin with the good supply function. In the next two statements we (1) create an equation setting `x1` equal to the expression contained in the temporary variable `tempx1`, and (2) assign the result of solving the equation for `p1` to the variable `plotx1`:

```
In[]:=     eqx1 = x1 == tempx1;
           plotx1 = Solve[eqx1,p1]
```

The result is the inverted good supply function in which `p1` appears as a function of `x1`:

Out[]= $\left\{\left\{p1\to\frac{w\left(\left(\frac{x1}{T}\right)^{\frac{1}{b}}\right)^{1-b}}{bT}\right\}\right\}$

Finally, we assign the above result to the temporary variable `tempp1`, give numerical values to the parameters, and generate the corresponding plot, obtaining the graph shown in Figure 3.7.

```
In[]:=     tempp1 = p1 /. plotx1[[1]];
           Plot[tempp1 /. {b → 0.4, T → 1, w → 100} ,
           {x1,0.01,0.1},
           AxesLabel → {"x1", "p1"},
           PlotLabel → "Supply Curve for x1"]
```

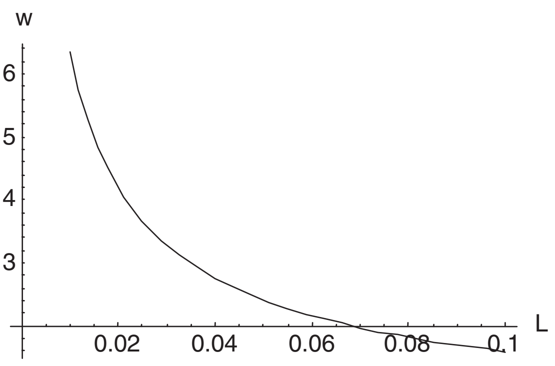

Figure 3.8. Labor demand curve.

In a similar way, with the following statements we generate the plot for the labor demand curve shown in Figure 3.8:

```
In[]:=      eqL = L == tempL;
            plotL = Solve[eqL, w];
            tempw = w /. plotL[[1]]
            Plot[tempw /. {b → 0.4, T → 1, p1 → 1},
            {L,0.01,0.1},
            AxesLabel → {"L", "w"},
            PlotLabel → "Labor Demand Curve"]
```

Now we are in a position to turn our attention to the market equilibrium.

3.4. MARKET EQUILIBRIUM

Having derived demand and supply curves, we can put them together to analyze the resulting market equilibrium. We do so for the case of good `x1`. We begin from the corresponding demand and supply curves obtained in the previous sections with a slight modification: the variable `p1` from the demand curve is renamed `p1d`, while the variable `p1` from the supply curve is renamed `p1s`.

We begin the Mathematica representation of the model of partial market equilibrium—available in the `MarketEquil.nb` file—with the statements

```
In[]:=      p1d = ρ m / x1;
            p1s = w (((x1 / T)^(1 / b))^(1-b)) / (b T);
```

Then we solve for the equilibrium quantity when demand equals supply,

```
In[]:=     equilx1 = Solve[p1d == p1s,x1]
```

obtaining as output

$$\texttt{Out[]=} \quad \{\{x1 \to \left(\frac{T^{-1/b}\ w}{b\ m\ p}\right)^{-b}\}\}$$

The equilibrium price can then be obtained by substituting the solution for `x1` into `p1d`:

```
In[]:=     equilp1 = p1d /. equilx1[[1]]
```

$$\texttt{Out[]=} \quad m\left(\frac{T^{-1/b}\ w}{b\ m\ p}\right)^{b} \rho$$

Next we assign values to the parameters and to the wage variable and compute the corresponding numerical values for the equilibrium quantity and the price. To do so, we write the variables `equilx1` and `equilp1` without semicolons, since Mathematica automatically replaces each parameter with its value, and performs the corresponding calculations:

```
In[]:=     ρ = 0.7;
           m = 0.1;
           T = 1;
           b = 0.4;
           w = 100;
           equilx1
           equilp1
Out[]=     {{x1 → 0.0379196}}
Out[]=     1.84601
```

Finally we plot jointly the demand and supply curves, obtaining the graph shown in Figure 3.9:

```
In[]:=     Plot[{p1d, p1s},
           {x1,0.01,0.1},
           AxesLabel → {"x1", "p1"},
           PlotLabel → "Market for x1"]
```

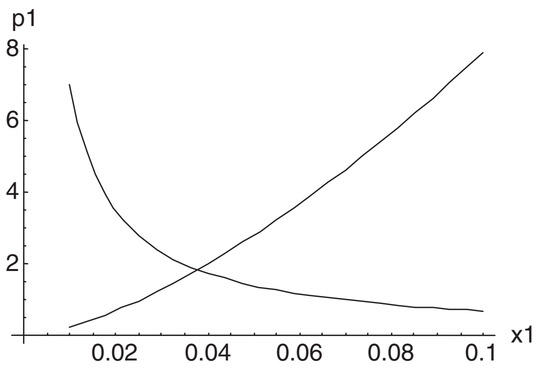

Figure 3.9. Market for `x1`.

Once we obtain the graphical representation of market equilibrium, it is interesting to engage in some comparative static exercises. To do so, we use a statement of the form

```
Plot[Evaluate[Table[ ] ] ]
```

This statement nests three Mathematica functions. The function

```
Table[expr, {i, imin, imax, di}]
```

makes a list of the values of an expression `expr` with `i` running from `imin` to `imax` in steps of `di`. The function

```
Evaluate[expr]
```

causes the expression `expr` to be evaluated. Finally the function `Plot[ ]` is the one we have used before. Thus, the statement

```
In[]:=      Plot[Evaluate[Table[{p1d ,p1s},{T,1,1.2,0.1}]],
            {x1,0.01,0.1},
            AxesLabel → {"x1", "p1"},
            PlotLabel → "Market for x1"]
```

first generates a list of three elements, one corresponding to each value of the technology parameter `T`, then evaluates the expression in each element of the list, and finally generates the plot shown in Figure 3.10.

Figure 3.11 shows the result of a similar experiment, but with the demand function share parameter ρ changed in the following way:

```
{ρ,0.5,0.9,0.2}
```

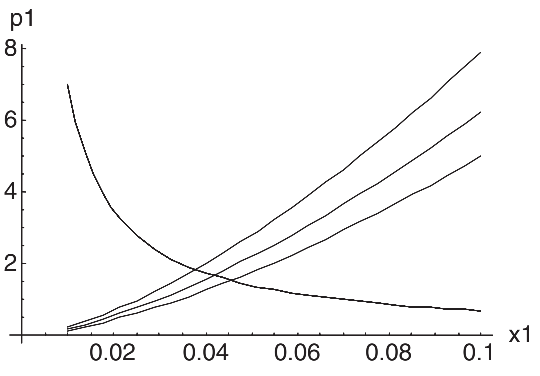

Figure 3.10. Comparative statics changing parameter T.

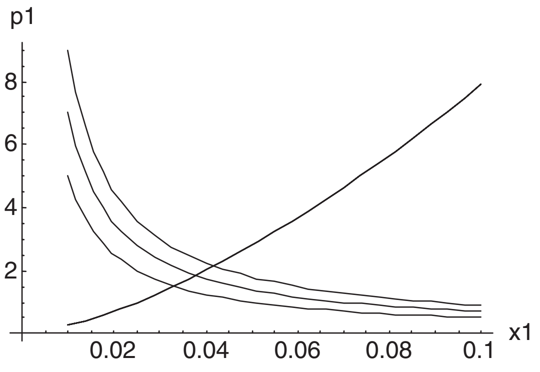

Figure 3.11. Comparative statics changing parameter ρ.

Finally, we perform the same comparative static exercise now with an animated plot using the following statement:

```
In[]:=     Table[Plot[{p1d ,p1s},
           {x1,0.01,0.1},
           PlotRange → {0,8},
           AxesLabel → {"x1", "p1"},
           PlotLabel → "Market for x1"],{T,1,1.2,0.1}]
```

Note that here we have a `Plot[ ]` function nested within a `Table[ ]` function. Thus, the table contains a sequence of plots controlled by the evolution of the `T` parameter. The output of the statement is such a sequence. Double click on the first graph of the sequence and you see the resulting animation. You can control the speed of the animation with the buttons that appear at the bottom of the notebook.

Note that here we fixed the range for the vertical axis with the option `PlotRange`. Otherwise, each plot may generate variable values for that range, creating the false impression that the demand curve is also shifting (to see this, eliminate that option from the statement and see what happens). Also note that if you perform other comparative static exercises changing any of the parameters other than `T`, you may have to adjust the `PlotRange` option accordingly, as well as the range for `x1`, setting different minimum and/or maximum values.

3.5. EXPERIMENTS

A simple set of experiments would be to perform more comparative static exercises changing some of the model parameters. You may also want to add parameters to the model (e.g., taxes) and see how this affects the outcome of the comparative statics.

Another popular function used to represent preferences or technology is the constant elasticity of substitution (CES) function

$$f(x_1, x_2) = (x_1^\alpha + x_2^\alpha)^{\frac{1}{\alpha}}$$

As we did with the Leontief and Cobb-Douglas functions, you may want to generate the contour plot of this function and see what happens as the parameter α goes from a value near zero to one near minus infinity.

Finally, you may want to develop an analysis analogous to the one we did in this chapter substituting the CES function for the Cobb-Douglas function.

3.6. FURTHER READING

For an introduction to Mathematica see Wolfram (2003). Consumer theory and the theory of the firm as well as competitive market equilibrium are at the core of most microeconomics textbooks. Later in this book we deal with duopoly models in Mathematica and general equilibrium models in GAMS.