Appendix 4

Using Analytic Solver Platform for Education

A4.1. BACKGROUND

Purchasers of this book have the option to download a Windows-based software package called **Analytic Solver Platform for Education** (**ASPE**), which has all the functionality of the commercial product Analytic Solver Platform but with restrictive limits on the size of optimization problems and "educational use" watermarks on charts. All the examples and exercises in this book can be solved within the limits of ASPE. In this Appendix, we introduce ASPE software and provide details on its use.

ASPE was developed by the same team that created Excel's Solver, and it will accommodate all models built with Excel's Solver. However, ASPE is a more powerful version of Excel's Solver and relies on a different user interface. ASPE is an integration of several software packages, including Risk Solver Platform for Education (RSPE), which is the optimization software included in ASPE. For the purposes of optimization, references to ASPE or RSPE are interchangeable.

A4.2. INSTALLING ASPE

If you've purchased this book and you are a student enrolled in a university course, you can download and install the software and use ASPE for a full semester (140 days) at no charge. To do so, visit www.solver.com/welcome-students, fill out the form on this page, and click the button to register and download. To complete the form, you'll need two pieces of information: a **Textbook Code**, which is **BOMS3**, and a **Course Code**, which your instructor can obtain for you from Frontline Systems, Inc.[1]

If you've purchased this book for self-study but you're not enrolled in a university course, you have two options: (1) Visit www.solver.com and register on the forms presented there, download and install the software, and use ASP (the full commercial product with much higher size limits) on a free trial, which is currently limited to 15 days; or (2) Contact Frontline Systems at 775-831-0300 or info@solver.com and request a Course Code that will allow you to use ASPE for 140 days. As long as you're using the software for learning rather than production use, the company's current policy is to routinely grant these licenses.

The software works with Excel 2007, Excel 2010 and Excel 2013, but if you are using the 64-bit version of Excel, be sure to download and run the Setup program for 64-bit Analytic Solver Platform, named SolverSetup64.exe. In all other cases, you'll download and run SolverSetup.exe.

---

[1] For longer-term licenses, course instructors should contact Frontline Systems at 775-831-0300 or info@solver.com

Running the Setup program is straightforward, but you'll need to pay attention to three prompts:

1. You'll be asked for an **installation password**. This will be emailed to you, at the email address you give when you fill out the registration form.
2. You'll be asked for a **license activation code**. This appears in the same email as the installation password; it determines whether you'll be using full ASP for 15 days, ASPE for 140 days, or something else.
3. You'll be asked whether you want to run initially as full Analytic Solver Platform, or a subset product. There are several choices, but for use with this book, the recommended selection is *Risk Solver Platform*. The user interface for optimization is the same in Analytic Solver Platform, Risk Solver Platform, and Premium Solver Platform. (You can change this selection later, using Help – Change Product on the ASPE/RSPE Ribbon.) For simplicity, the instructions here refer just to ASPE.

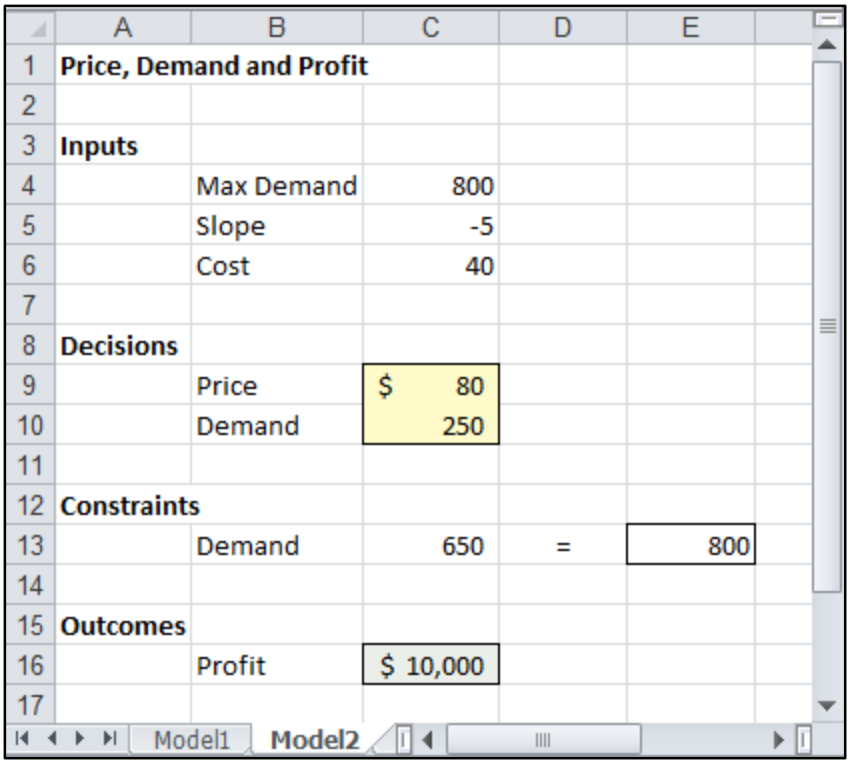
To uninstall the software, you can either re-run SolverSetup.exe, or use the Windows Add/Remove Programs feature.

A4.3. THE ASPE USER INTERFACE

In order to illustrate the use of ASPE, we return to Example 1.1 in Chapter 1. The optimization problem is to find a price that maximizes quarterly profit contribution. An algebraic statement of the problem is as follows:

Maximize	$z = (x - 40)\,y$	(objective)
subject to	$y + 5x = 800$	(constraint)

This form of the model corresponds to Figure 1.2 (reproduced here as Figure A4.1), which contains two decision variables (x and y, or price and demand) and one constraint on the decision variables. The spreadsheet model is ready for optimization.



	A	B	C	D	E
1	**Price, Demand and Profit**				
2					
3	**Inputs**				
4		Max Demand	800		
5		Slope	-5		
6		Cost	40		
7					
8	**Decisions**				
9		Price	$ 80		
10		Demand	250		
11					
12	**Constraints**				
13		Demand	650	=	800
14					
15	**Outcomes**				
16		Profit	$ 10,000		
17					

Figure A4.1. Model for Example 1.1

To start, we select the Risk Solver Platform tab and click on the Model icon (on the left side of its ribbon). This step opens the *task pane* on the right-hand side of the Excel window. The task pane contains four tabs: Model, Platform, Engine, and Output. Initially, the Model tab displays a window listing several components of the software, including Optimization. In Figure A4.2, we have expanded the Optimization entry on the Model tab. As we specify the elements of our model, they are recorded in the folder icons of this window. At the top of the model tab, five icons appear:

- Green "plus" sign, to *Add* model specifications
- Red "delete" sign, to *Remove* specifications
- Orange paired sheets with small blue arrows, to *Refresh* the display after changes
- Green checked sheet, to *Analyze* the model
- Green triangle, to *Solve* the specified optimization problem

To specify the model we first select the decision cells (C9:C10) and then on the drop-down menu of the *Add* icon, we select Add Variable. The range **$C$9:$C$10** immediately appears in the Model window, in the folder for Normal Variables. (Another way to accomplish this step without the drop-down menu is to highlight the Normal Variables folder icon and click the *Add* icon.)

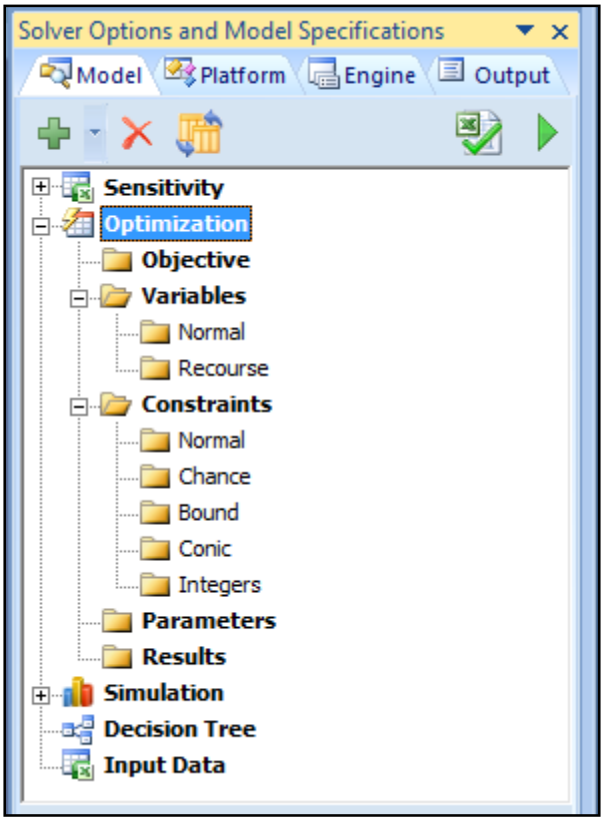


Figure A4.2. Model tab on the initial task pane

Next, we select the objective cell (C16) and on the drop-down menu of the *Add* icon, select Objective. The cell address **$C$16** immediately appears in the Model window, in the folder for Objective. By default, the specification assumes that the objective is to maximize this value. (We can also implement this step by highlighting the Objective folder and simply clicking the *Add* icon.)

Next, we select the left-hand side of the constraint (C13) and on the drop-down menu of the *Add* icon, select Add Constraint. (Alternatively, we can highlight the Normal Constraints folder icon and click the *Add* icon.) The Add Constraint window appears, with the cell address **$C$13** in the Cell Reference box, as shown in Figure A4.3. On the drop-down menu to its right, we select "=" and enter E13 in the Constraint box (or, with the cursor in the box, select cell E13.).

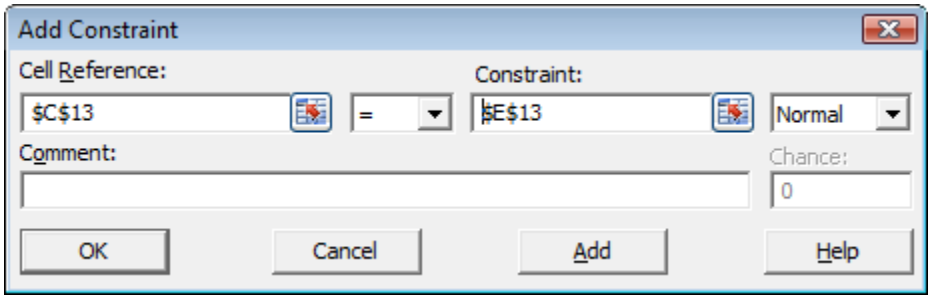


Figure A4.3. Add Constraint window

As mentioned in Chapter 1, one of our design guidelines for Solver models is to reference a cell containing a *formula* in the Cell Reference box and to reference a cell containing a *number* in the Constraint box. The use of cell references keeps the key parameters visible on the spreadsheet rather than in the less accessible windows of Solver's interface. Ideally, another person would not have to examine the task pane to understand the model. (Although Solver permits us to enter numerical values directly into the Constraint box, this form is less effective for communication and complicates sensitivity analysis. It would be reasonable only in special cases where the model structure is obvious from the spreadsheet and where we expect to perform no sensitivity analyses for the corresponding parameter.)

Finally, we press OK and observe that the task pane displays the model's specification, as shown in Figure A4.4. In summary, our model specification is the following:

Objective:	C16 (maximize)
Variables:	C9:C10
Constraint:	C13 = E13

If necessary, we can edit specifications by double-clicking on the corresponding cell address as it appears on the Model tab.

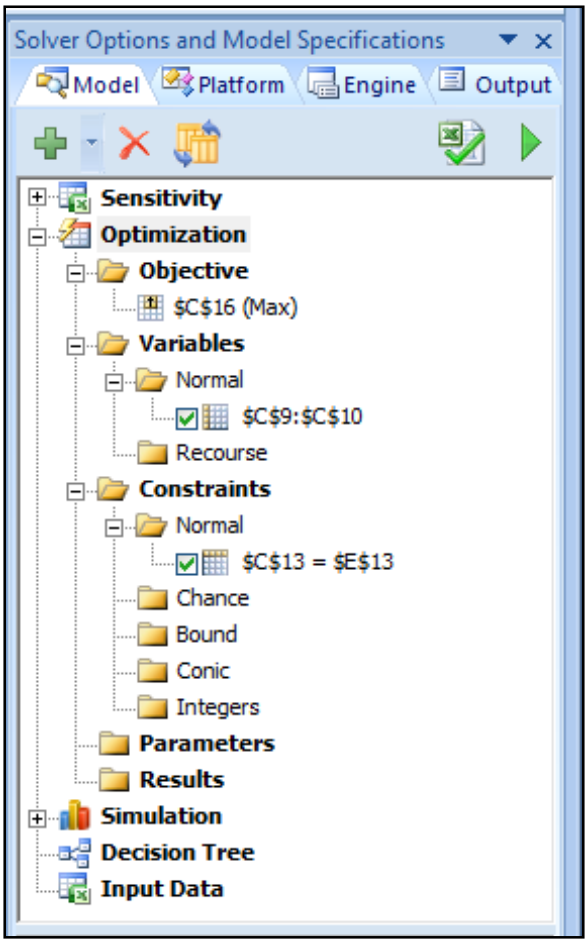


Figure A4.4. Model specification

This model is simple enough that we need not address the information on the Platform tab. (However, it is generally a good idea to set the Nonsmooth Model Transformation option to *Never*.) At the top of the Engine tab, we observe the default selection of the Standard LSGRG Nonlinear Engine, which we refer to as the *nonlinear solver*. (To ensure this selection, we uncheck the box for Automatically Select Engine.) This solution algorithm is appropriate for our optimization problem, and we do not need to address most of the other information on the tab. However, one of the options is important.

Although we may guess that the optimal price is a positive quantity, the model as specified permits the price decision to be negative. Such an outcome would not make sense in this problem, so it may be a good idea to limit the model to nonnegative prices. In fact, virtually all of the models in this book involve decision variables that make practical sense only when they are nonnegative, so we will impose this restriction routinely. On the Engine tab of the task pane, we find the *Assume Non-Negative option* in the General group and change it to True, using the drop-down menu on the right-hand side, as shown in Figure A4.5.

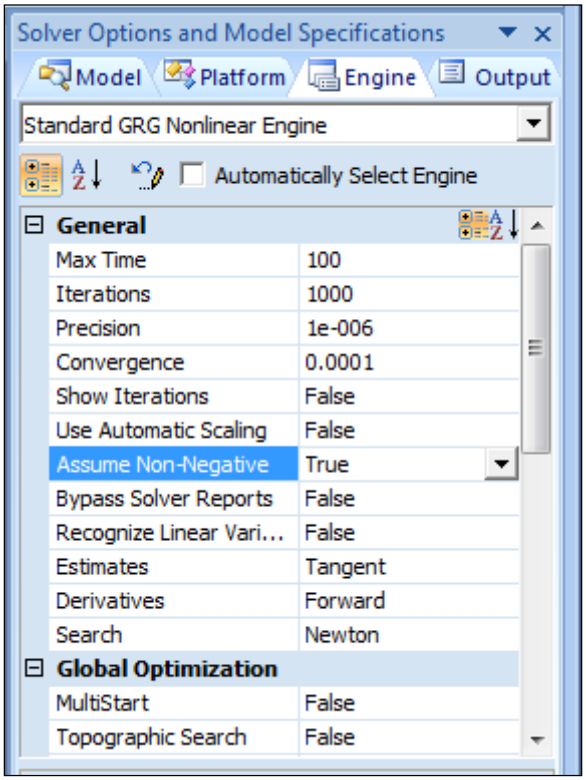


Figure A4.5. Setting the Assume Non-Negative option

Finally, we proceed to the Output tab (or return to the Model tab) and click the *Solve* icon. Solver searches for the optimal price and ultimately places it in the price cell. In this case, the optimal price is $100, and the corresponding quarterly profit contribution is $18,000, as discussed in Chapter 1.

Meanwhile, the Output tab's window displays the solution log for the optimization run. (The detail in this log is controlled by the *Log Level option* on the Platform tab, but the default

setting of *Normal* is usually adequate.) The most important part of the log is the Solver Results message, which in this case states:

> **Solver found a solution. All constraints and optimality conditions are satisfied.**

This *optimality message*, which is repeated at the very bottom of the task pane, tells us that no problems arose during the optimization, and Solver was able to find an optimal solution.

We have used Example 1.1 to introduce the user interface of ASPE. The task pane contains many user-selected options that are not a concern in this problem. Later in this appendix, we cover many of these settings and discuss when they become relevant. We also discuss the variations that can occur in optimization runs. For example, depending on the initial values of the decision variables, the nonlinear solver may generate the following result message in the solution log:

> **Solver has converged to the current solution. All constraints are satisfied.**

This *convergence message* indicates that Solver has not been able to confirm optimality. As mentioned in Chapter 1, this condition occurs because of numerical issues in the solution algorithm, and the resolution is to rerun Solver from the point where convergence occurred. Normally, one or two iterations are sufficient to produce the optimality message. We discuss some other result messages later.

To minimize an objective function instead of maximizing it, we return to the Model tab of the task pane and double-click on the entry in the Objective folder. The Change Objective window appears, as shown in Figure A4.6, and we can select the button for Min rather than Max.

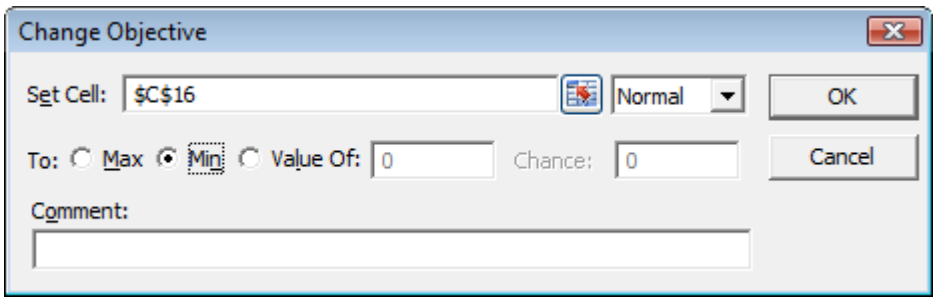


Figure A4.6. Selecting minimization of an objective

When an optimization model contains several decision variables, we can enter them one at a time, creating a list of Normal Variables in the task pane, each with its own checked box. More conveniently, we can arrange the spreadsheet so that all the variables appear in adjacent cells, as in Figure A4.1, and reference their cell range with just one entry in the Normal Variables folder. Because most optimization problems have several decision variables, we save time by placing them in adjacent cells whenever that's convenient. This layout also makes the information in the task pane easier to interpret.

A4.4. SOLVING LINEAR PROGRAMS

When solving a linear programming problem, we enter the problem specification using the task pane. This means entering information about the objective function, decision variables, and constraints, using the elements of the Model tab as described in the previous section. (Lower bound and upper bound constraints are entered in the same fashion, using the Add Constraint window. However, once they are added to the model, they are displayed in the task pane in the Bounds section.) After the model information is entered, we move to the Engine tab and set the Assume Non-Negative option to True.

At the top of the Engine tab, we uncheck the box for Automatically Select Engine and, from the drop-down menu above it, select *Standard LP/Quadratic Engine*. This is the name of the linear solver in ASPE. (The same algorithm will solve problems with a quadratic objective function; hence its name.) This algorithm is an enhanced version of the linear solver in Excel. Once the settings on the Engine tab are specified, we can run the solution algorithm by moving to the Model tab or the Output tab and clicking the Solve icon. If no difficulties are encountered, the optimality message appears (with a green background) at the bottom of the task pane, with the optimal solution displayed on the spreadsheet. The optimality message also appears on the Output tab, formatted as a link. Clicking on this link opens a Help window that explains the optimality message. The explanation may seem more extensive than necessary, but that's because the same optimality message appears when other solvers ("engines") are run. Therefore, the explanation covers several different cases that are not a concern when solving linear programs.

As described in Chapter 2, two exceptions can cause difficulties: infeasible constraints and an unbounded objective function. When the model contains inconsistent constraints, Solver detects the infeasibility and delivers the following result message in the solution log on the Output tab as well as at the bottom of the task pane:

```
Solver could not find a feasible solution.
```

Whenever this message appears, there must have been an inconsistency in the set of constraints. Chapter 2 contains a discussion about how to address this type of problem. (Keep in mind that the infeasibility message appears in the Output tab as a link to a Help window where further interpretation is available.)

The second kind of modeling error occurs when the objective function is not bounded in the direction of optimization. Solver detects the unbounded possibilities and delivers the following result message in the solution log on the Output tab as well as at the bottom of the task pane:

```
The objective (Set Cell) values do not
converge.
```

The reference to "Set Cell" provides consistency with older versions of the software, which did not use the term "objective."

A4.5. SOLVING INTEGER LINEAR PROGRAMS

As in the case of Excel's Solver, the integer solver in ASPE is suited to linear programs in which some or all of the variables must be integers, but ASPE is not appropriate for tackling nonlinear integer programs. When it comes to integer linear programs, we specify the objective function, decision variables, and constraints, and then identify certain variables as having integer restrictions. This is done with the Add Constraint window in ASPE, just as it is in Excel's Solver. In the drop-down menu of the Add Constraint window, we have the option of declaring a variable to be integer (int) or binary (bin), as in Excel's Solver. When such a declaration is made in ASPE, it will show up in the Integers section of the Constraints folder in the Model tab of the task pane. In other words, the declaration of any variable as integer or binary automatically conveys to ASPE the necessity of using an integer solver.

On the Engine tab, we again uncheck the box for Automatically Select Engine and from the drop-down menu, we can select the Standard LP/Quadratic Engine. This combination invokes the linear solver together with the branch and bound algorithm (see Chapter 6) in order to find an optimal solution. As of this writing, however, a better option is available. For solving integer programs, the user can select the Gurobi Solver Engine, which is a licensed student version of a popular third-party code for solving integer programming problems. The Gurobi Solver Engine tends to be more efficient than the Standard LP/Quadratic Engine, and for this reason the Gurobi Solver Engine is the better choice for integer programs.

Before running the integer solver (either the Standard LP/Quadratic Engine or the Gurobi Solver Engine), two settings on the Engine tab should be selected. First, the Assume Non-Negative option should be selected, just as in linear programs, at least for the variables in the model that are not constrained to be integers. (If all variables are either integer or binary, then this selection becomes redundant.) Second, the equivalent of the Integer Optimality parameter in Excel's Solver should be set. With the Standard LP/Quadratic Engine, this option is the *Integer Tolerance* parameter which appears in the Integer section of the Engine tab on the task pane). With the Gurobi Solver Engine, the comparable option is called the *Optimality Tolerance* parameter, which appears in the Engine tab (in the Tolerances section of the task pane).

A4.6. SOLVING NONLINEAR PROGRAMS

The main difference in Excel's Solver between specifying linear and nonlinear models is the selection of a solution procedure. With Excel's Solver, the nonlinear solver is the GRG Nonlinear procedure, whereas in ASPE the nonlinear solver is called the *Standard LSGRG Nonlinear Engine*. The algorithm used in ASPE is an advanced (large-scale) version of the nonlinear solver in Excel. However, although the Standard LSGRG Nonlinear Engine has more features and runs more efficiently, the outcomes that a user sees are not conceptually any different. The main results are the optimality message and the convergence message. (The convergence message tends to occur less frequently with the Standard LSGRG Nonlinear Engine because it is better able to find a solution in which the optimality conditions hold.) Of course, a poorly formulated nonlinear model could lead to the infeasibility message or the unbounded message, just as in linear programs. Finally, the Multistart option exists in ASPE and can be invoked as an option on the Engine tab (in the Global Optimization section).

Scaling problems can arise in ASPE, but the algorithms in ASPE are more robust in this respect than in Excel's Solver. However, it is still possible that a correctly formulated nonlinear

model will produce an error message that indicates infeasibility when, in fact, the issue is one of scaling. ASPE tends to be better able to overcome scaling challenges than Excel's Solver.

The nonlinear solver in ASPE is capable of solving linear as well as nonlinear programs. However, just as in the case of Excel's Solver, the linear solver is more reliable and should always be used when solving linear models.

A4.7. ALGEBRAIC TRANSFORMATIONS AND AUTOMATIC ENGINE SELECTION

Section 5 of Chapter 8 discusses the linearization of nonlinear models. To repeat a main point in that section, some problems that are formulated naturally with nonlinear functions can be reformulated as linear programs. The advantage of doing so is that the reformulation (or *transformation*) permits us to use the linear solver rather than the nonlinear solver. If we were to use the nonlinear solver, we would have to worry about encountering local optima, whereas the linear solver always leads to a global optimum. The two examples described in section 8.5 involve the use of the MAX function and the ABS function.

The appearance of the MAX function or the ABS function in an optimization model ideally triggers an error message if the linear solver is invoked: *The linearity conditions required by this Solver engine are not satisfied.* Using the techniques of Section 8.5, the user must then reformulate the model in a linear fashion. When this is accomplished satisfactorily, the linear solver can be invoked and a solution found. In ASPE, this transformation can sometimes be accomplished automatically.

The key option is the *Nonsmooth Model Transformation option*, which is located on the Platform tab of the task pane, in the Transformation section. The default setting for this option is Automatic, but it seems preferable to set the option to Never unless there is a specific reason for wanting to invoke it. When the option is set to Never for the model in Figure 8.16, the Standard LP/Quadratic Engine produces the linearity error message. However, when the option is set to Automatic or to Always, the same Engine produces the optimal solution shown in Figure 8.17.

To produce the optimal solution, ASPE first transforms the model into a linear equivalent and then uses the Standard LP/Quadratic Engine, which has previously been selected. An important feature in this case is that the transformed model is never completely visible to the user. The log that appears in the Output tab of the task pane confirms that a transformation was used, and an examination of the problem size in the Current Problem section of the Engine tab provides additional information. Before the transformation, the problem size is specified as 32 variables and 8 constraints; after the transformation, the specification becomes 38 variables and 20 constraints. None of these alterations is visible on the spreadsheet when the optimal solution is found. Moreover, the details of the transformation do not match the reformulation method described in Section 8.5, which led to a model containing 33 variables and 12 constraints.

As another example, we revisit the model in Figure 8.18, in which the objective is to minimize the sum of absolute differences among the departmental loads. As built, the model contains 32 variables and 8 constraints, just like the model in Figure 8.17. The absolute differences are computed in cells E14:E19, and they become inputs for the objective function in cell J14. Because of the presence of the ABS function, the Standard LP/Quadratic Engine cannot solve this problem when the Transformation option is set to Never. However, when the option is set to Automatic, the same Engine finds an optimal solution based on a model containing 44 variables and 44 constraints.

The Transformation option applies to the use of the MAX function, the ABS function, and some IF functions, when they appear in the objective function of a model. (The option is not

reliable when those nonlinear functions appear in constraints.) This can potentially be a convenience for the user because the model can be constructed in a straightforward manner on the spreadsheet, making use of nonlinear functions, and there is no need to implement the potentially challenging types of transformations described in section 8.5.

A4.8. SOLVING WITH THE EVOLUTIONARY ENGINE

The evolutionary solver in Excel implements a proprietary algorithm in the family of genetic algorithms. The background discussion in Section 2 of Chapter 9 provides a stylized description of a genetic algorithm, to give a sense of what the user might expect. The Evolutionary Engine in ASPE is an advanced version of this algorithm, although its details remain proprietary.

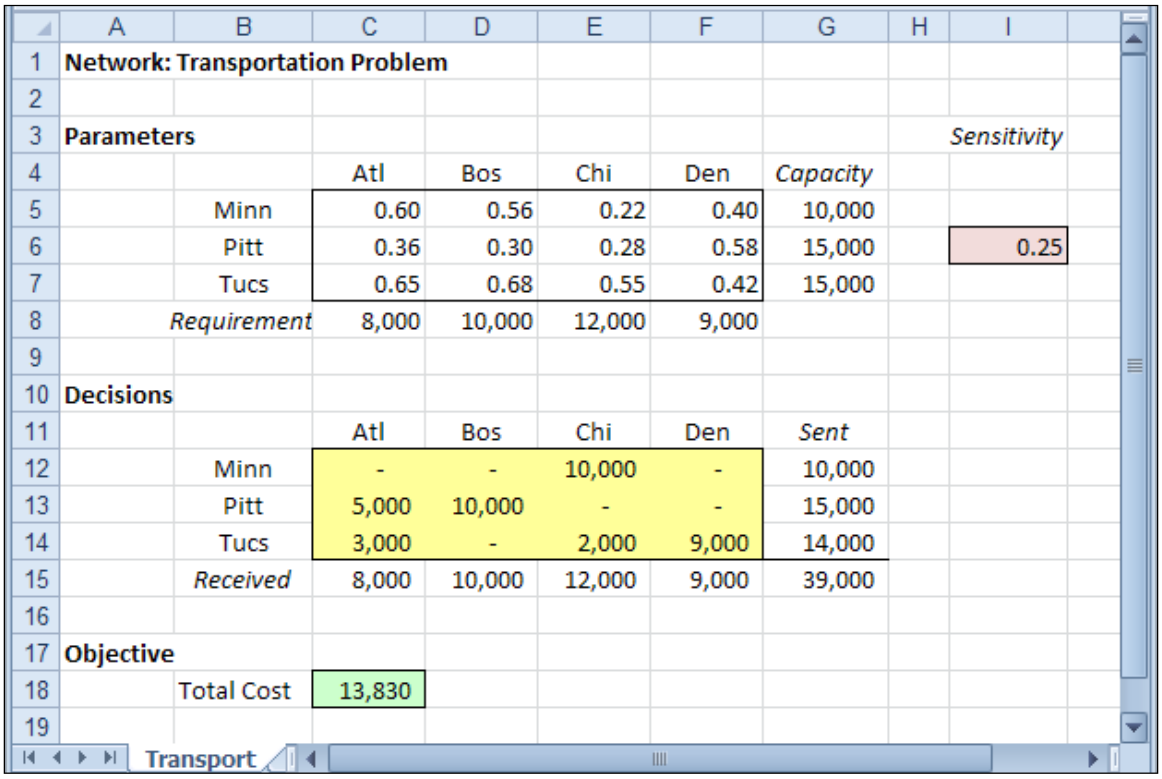
When using the evolutionary solver in Excel, the problem is specified in the Solver Parameters window, and various options are specified in the Options menu on the Evolutionary tab. These options appear in ASPE in the task pane on the Engine tab when the Evolutionary Engine has been selected. Included is the Tolerance option, which may be set to a value different from zero. One of the possible termination conditions is that the algorithm has not found a solution that improves the objective by at least the Tolerance value after the Max Time without Improvement. (In Excel's Solver, the Tolerance is implicitly set to zero, but the ASPE user can set it to a different level.)

In ASPE, an important option is the Non-Smooth Model Transformation option, which appears in the Transformation section of the Platform tab. This option should be set to Never whenever the Evolutionary Engine is used, otherwise the model may be inadvertently modified in inefficient ways. (The transformation option is discussed above.) No such option exists in Excel's Solver, so the effect of this option is a matter of concern only to users of ASPE.

A4.9. SENSITIVITY ANALYSIS

In Chapter 4 we discussed the Solver Sensitivity tool, which identifies a set of values for one or two parameters and then runs Solver under each specified value of the parameters involved. ASPE provides a similar tool that allows us to change a parameter, rerun Solver automatically, and record the impact of the parameter's change on the optimal value of the objective function and on the optimal decisions. The output of the tool is the *Parameter Analysis Report*.

As an illustration, we revisit the transportation problem introduced in Chapter 3. In Example 3.1 (Goodwin Manufacturing Company), the model allows us to find the cost-minimizing distribution schedule in a setting with three plants shipping to four warehouses. We encountered the optimal solution in Figure 3.2, which is reproduced in Figure A4.7. The tight supply constraints are Minneapolis and Pittsburgh capacity, and the optimal total cost in the base case is $13,830.



	A	B	C	D	E	F	G	H	I
1	**Network: Transportation Problem**								
2									
3	**Parameters**								*Sensitivity*
4			Atl	Bos	Chi	Den	*Capacity*		
5		Minn	0.60	0.56	0.22	0.40	10,000		
6		Pitt	0.36	0.30	0.28	0.58	15,000		0.25
7		Tucs	0.65	0.68	0.55	0.42	15,000		
8		*Requirement*	8,000	10,000	12,000	9,000			
9									
10	**Decisions**								
11			Atl	Bos	Chi	Den	*Sent*		
12		Minn	-	-	10,000	-	10,000		
13		Pitt	5,000	10,000	-	-	15,000		
14		Tucs	3,000	-	2,000	9,000	14,000		
15		*Received*	8,000	10,000	12,000	9,000	39,000		
16									
17	**Objective**								
18		Total Cost	13,830						
19									

Transport

Figure A4.7. Solution for Example 3.1

Suppose that we are using the transportation model as a planning tool and we want to explore a change in the unit cost of shipping from Pittsburgh to Atlanta, which is $0.36 in the base case, and we want to study a range of alternative values for the *PA* cost. Suppose that, as a first step, we are willing to examine a large range of values, from $0.25 to $0.75. For this purpose we create a cell and enter the formula `=PsiOptParam(0.25,0.75)`. In Figure A4.7, we have reserved column I for sensitivity parameters and entered the formula in cell I6. Then, in C6 we reference cell I6. The PsiOptParam function displays the minimum value of its range, so the value $0.25 appears in both cell I6 and cell C6. (To restore the model, we would simply enter the original unit cost of $0.36 in cell C6.)

Next, we go to the drop-down menu on the Reports icon in the Risk Solver Platform ribbon and select Optimization▶Parameter Analysis. The Multiple Optimizations Report window, shown in Figure A4.8, appears on the screen. In the Multiple Optimizations Report window, we select the results to track in the upper part of the report and the parameter(s) to vary in the lower part. In particular, we fill out the form by choosing C18 as the objective function cell and C13:F13 as the variables to track. (These four cells represent the optimal shipments from the Pittsburgh plant.) Those selections are displayed in the upper right-hand window, as shown in Figure A4.9. As a general rule, we prefer to have the objective function listed first, as shown in the figure. (The Up and Down buttons help us reorder the list.) In the bottom pair of windows, we select cell I6 as the parameter to vary.

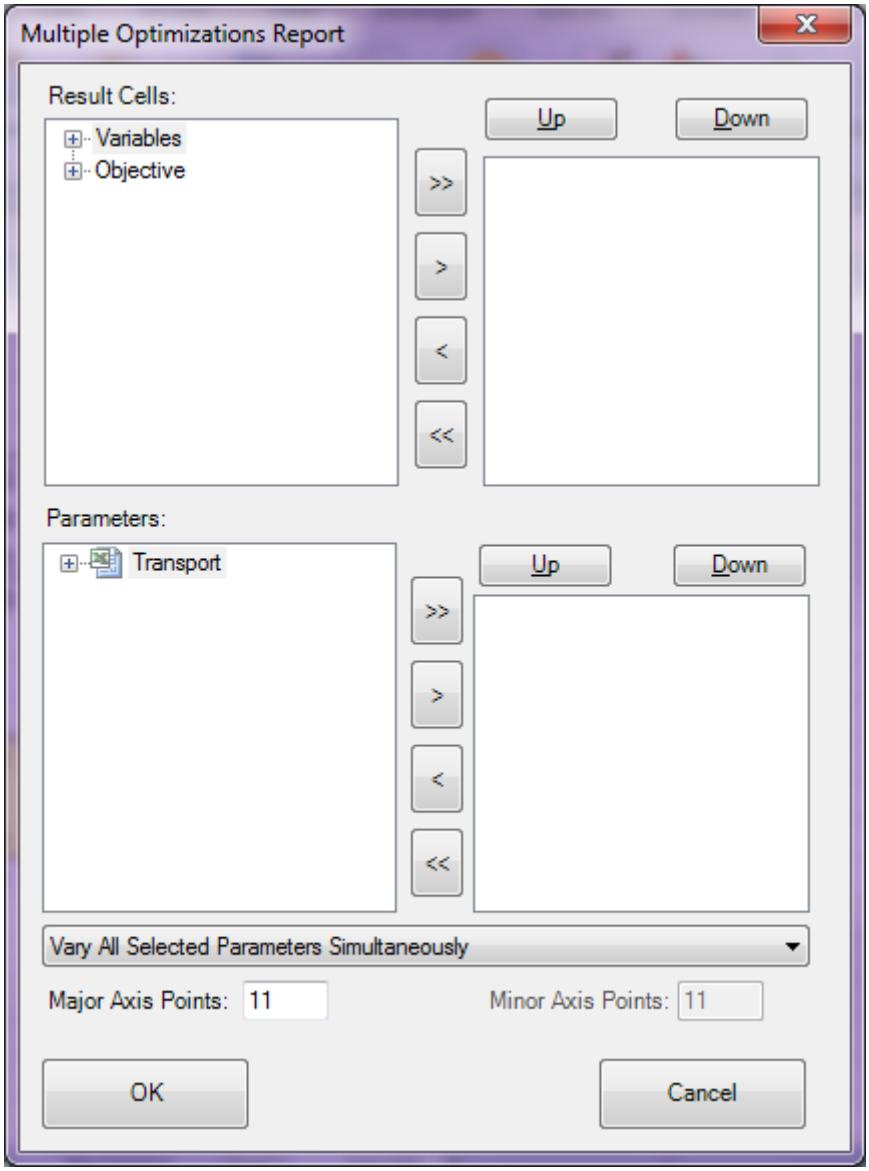


Figure A4.8. Initial Multiple Optimizations report window

Near the bottom of the window is a drop-down list of options for varying parameters; in this case we select the option to Vary All Selected Parameters Simultaneously. Next, we have to specify the number of values between the parameter's lower limit of $0.25 and the upper limit of $0.75. If we use the default of 11 Major Axis Points, as in Figure A4.9, the step size will be $0.05. (That is, starting at 0.25 and taking steps of 0.05, the eleventh step will be 0.75.).

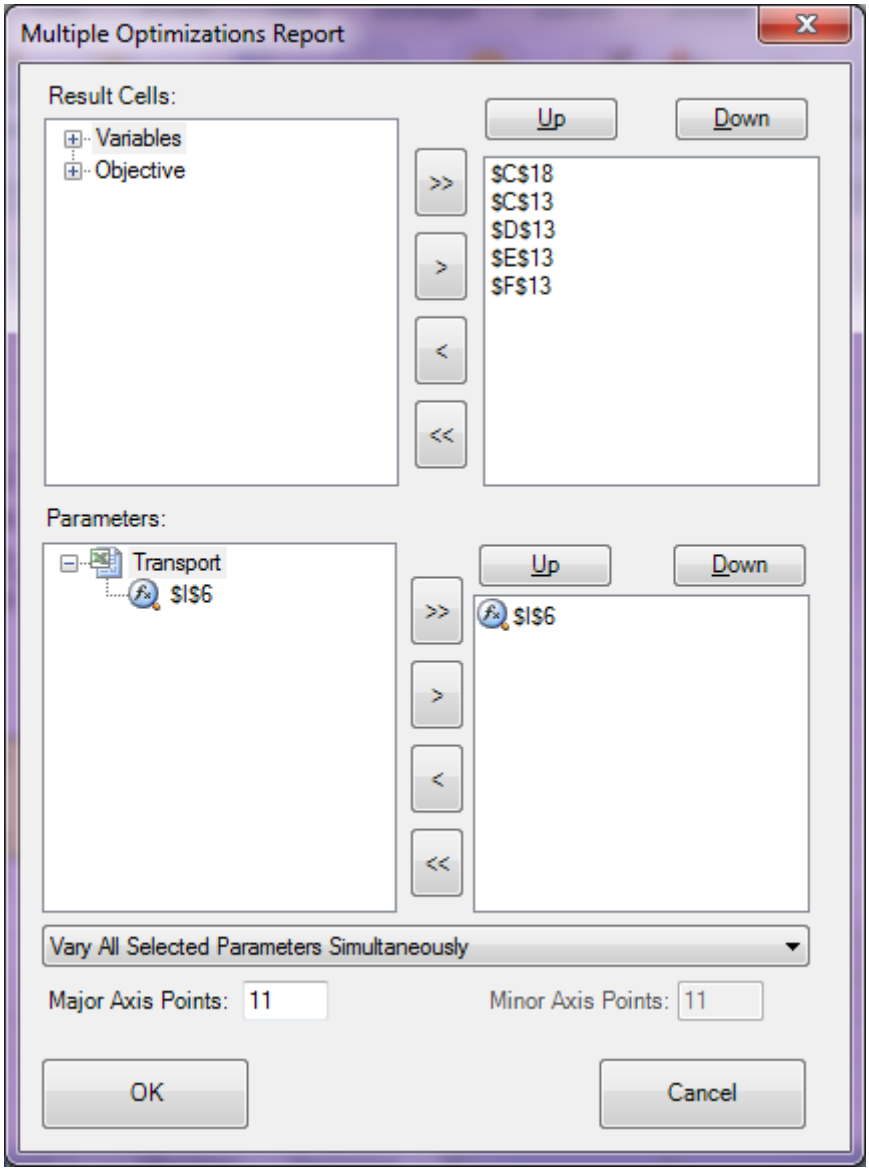
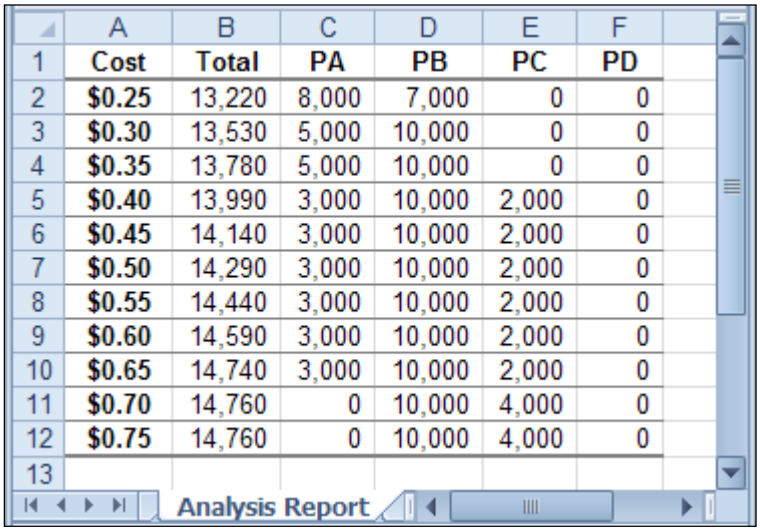


Figure A4.9. Selections in the Multiple Optimizations report

Finally, we click OK on the form. The program inserts a new worksheet in our workbook, labeled Analysis Report, and records the results. Figure A4.10 displays the Parameter Analysis Report, slightly edited for better readability. The report shows how the optimal total cost changes and how the optimal Pittsburgh shipments change as the unit cost of the *PA* route increases. The report format closely resembles the Solver Sensitivity report described in Chapter 4. To produce a report with a more refined grid (such as 0.01), we would repeat this procedure but designate a larger number of Major Axis Points.



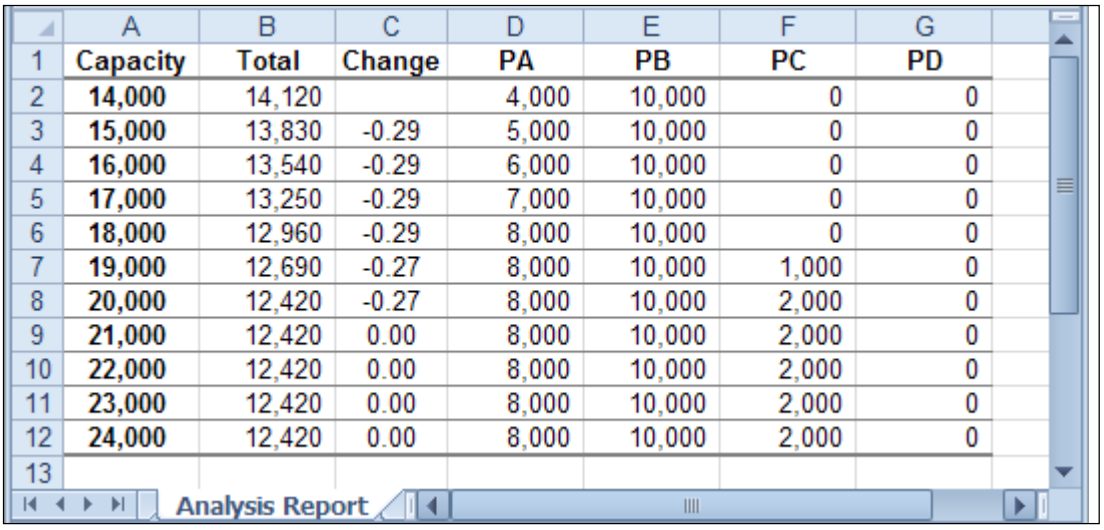
	A	B	C	D	E	F
1	Cost	Total	PA	PB	PC	PD
2	$0.25	13,220	8,000	7,000	0	0
3	$0.30	13,530	5,000	10,000	0	0
4	$0.35	13,780	5,000	10,000	0	0
5	$0.40	13,990	3,000	10,000	2,000	0
6	$0.45	14,140	3,000	10,000	2,000	0
7	$0.50	14,290	3,000	10,000	2,000	0
8	$0.55	14,440	3,000	10,000	2,000	0
9	$0.60	14,590	3,000	10,000	2,000	0
10	$0.65	14,740	3,000	10,000	2,000	0
11	$0.70	14,760	0	10,000	4,000	0
12	$0.75	14,760	0	10,000	4,000	0
13						

Analysis Report

Figure A4.10. Parameter Analysis Report for *PA* unit cost

A Parameter Analysis Report can be generated for a constraint parameter in much the same fashion. Suppose that, starting from the base case, we wish to explore a change in the Pittsburgh capacity of 15,000. To prepare for this analysis, we first have to determine the range of values to study, such as 14,000 to 24,000. Then we devote a cell in the sensitivity area of the spreadsheet, such as I7, to this parameter. In this cell, we enter the formula `=PsiOptParam(14000,24000)`, and we reference this cell in G13. (To make sure that the analysis varies only this capacity, we must remember to set cell C6 back to its original value of $0.36.)

Next, we return to the Reports drop-down menu and ask for an Optimization Parameter Analysis. Using 11 Major Axis Points, we can examine the effect of increasing Pittsburgh capacity from 14,000 to 24,000 in steps of 1,000. A slightly edited version of the report appears in Figure A4.11 with column C added manually. The entries in each row of column C represent the incremental change in the optimal objective divided by the change in the right-hand-side constant, from the row above. The formula in cell C3 is `=(B2-B3)/(A2-A3)`, and it has been copied down the column. This calculation corresponds to the Change column produced by Solver Sensitivity, thus displaying the shadow price as calculated on a discrete grid.



	A	B	C	D	E	F	G
1	**Capacity**	**Total**	**Change**	**PA**	**PB**	**PC**	**PD**
2	**14,000**	14,120		4,000	10,000	0	0
3	**15,000**	13,830	-0.29	5,000	10,000	0	0
4	**16,000**	13,540	-0.29	6,000	10,000	0	0
5	**17,000**	13,250	-0.29	7,000	10,000	0	0
6	**18,000**	12,960	-0.29	8,000	10,000	0	0
7	**19,000**	12,690	-0.27	8,000	10,000	1,000	0
8	**20,000**	12,420	-0.27	8,000	10,000	2,000	0
9	**21,000**	12,420	0.00	8,000	10,000	2,000	0
10	**22,000**	12,420	0.00	8,000	10,000	2,000	0
11	**23,000**	12,420	0.00	8,000	10,000	2,000	0
12	**24,000**	12,420	0.00	8,000	10,000	2,000	0
13							

Analysis Report

Figure A4.11. Parameter Analysis Report for Pittsburgh capacity

In addition to the one-at-a-time analysis illustrated in these figures, the Parameter Analysis Report can be built for two-at-a time analysis using the drop-down option for Vary Two Selected Parameters Independently. This option produces a two-dimensional table much like the one produced by Solver Sensitivity. For this type of report, the upper right-hand window must hold only one cell address (usually the objective function), and each cell in the report table shows its value for a different pair of values of the selected parameters.

A third option on the drop-down menu is to Vary All Selected Parameters Simultaneously. In this case, several parameters are varied at the same time, and the number of Major Axis Points applies to all of the variables at once.

The **Sensitivity Report**, which was covered in Chapter 4, is also available in ASPE, once the optimal solution has been found. From the drop-down menu on the Reports icon in the Risk Solver Platform ribbon, we select Optimization►Sensitivity. The Sensitivity Report then appears on a new worksheet, immediately before the model worksheet. The Sensitivity Report exhibits the same features discussed in Chapter 4 for Excel's Solver, except that in ASPE the report has a brief first section that reproduces the optimal value of the objective function.

ASPE offers several other forms of analysis and reports, including automated charting. Users can consult the Reference Guide or the User's Guide for details.