SPREADSHEET PROGRAMS

BRIEF HISTORICAL OVERVIEW

At its most fundamental a spreadsheet is an electronic grid consisting of rows and columns where each cell in the grid contains either data or the relationship between the contents of other cells. As new data are entered or existing data are amended the spreadsheet recalculates the relationships between the cells to reflect the most recent changes.

In its original incarnation the spreadsheet was presented as an electronic version of the accountant's ledger with automated basic operations, such as sum, count, average, maximum, and minimum. The first electronic spreadsheet, Visicalc, was created by Dan Bricklin and Robert Frankston for the Apple II in 1978. It sold for $150. Visicalc was followed in rapid succession by SuperCalc, Multiplan and eventually Lotus 123 in 1983. Lotus Corporation became the spreadsheet market leader and set the spreadsheet standards for several years after Release 2 of Lotus 123 in 1985. Standard features included an increasingly large library of mathematical functions, easier graphing and printing, elementary database manipulations and the ability to customize and program via the macro language. By the mid-to late-1980's, spreadsheets permitted reasonably complex and sophisticated models to be built at the user's desktop. Although accounting and financial analysis software was not new (and available on mini and mainframe computers of the time), spreadsheets were targeted specifically for the rapidly evolving personal computers. Their success can be attributed largely to providing end users with control of a powerful calculation and decision-aiding tool at their desktop. Spreadsheets became the "killer application" for microcomputers and gave rise to a cottage industry of a myriad of add-in applications, including more fonts, landscape printing, memory extenders and managers, and display and publishing features.

In 1987, two new spreadsheet packages started to eat into Lotus's dominance of the spreadsheet market, namely Microsoft Excel (Microsoft Corporation, Redmond, WA) and Quattro Pro (Borland, Cupertino, CA). The latest release of each package leapfrogged its competitors for a short time with incremental improvements and new features: Spreadsheets could be composed of multiple sheets, links could be made to other files, easier and more sophisticated graphing facilities could be used, and the graphical user interface (GUI) could be applied with its shortcut buttons and customization options. With time, the add-in packages became more sophisticated and many were integrated into the spreadsheet itself. For example, statistical analyses can today be performed very simply from directly inside the spreadsheet; Solver (Frontline Systems, Inc., Incline Village, NV) is a full-functioned mathematical programming package that will calculate optimal values for decision variables.

By the early 1990's, Excel emerged as the market leader and continues to be the dominant spreadsheet, so much so that "Excel" has entered the lingua franca to mean "spreadsheet." With the arrival of the Internet, Excel provided hyperlinks and Web publishing features as part of its continual evolution. While Excel maintained its market dominance, other spreadsheets continued to exist mostly within competing office suites to Microsoft Office, for example, Quattro as part of Corel's WordPerfect Office (Corel Corporation, Ottawa, Ontario, Canada) and Sun Microsystems's open source Star Office (Sun Microsystems, Inc., Santa Clara, CA). More recently, in 2005, Web spreadsheets sprung up, which allow users to upload, create, and edit spreadsheets online and collaborate with others, in real time, and track changes. Google, Inc. (Mountain View, CA) entered this market in mid-2006 with its free offering of Google Docs & Spreadsheets, which might signal the direction for the future. A detailed history of the early development of spreadsheets is provided by Power (1).

SPREADSHEET PACKAGES AND USERS

At one end of the spectrum a spreadsheet can be viewed as a large and powerful calculator. However, modern spreadsheets provide a sophisticated modeling environment with an interface that can be productively used by an end-user novice and an operations research/management scientist (OR/MS) expert alike. Users of spreadsheets can broadly be classified into two categories, those that use spreadsheets in a "static" fashion typically for all kinds of business data processing such as financial statements, inventory tracking, sales management, and budgeting applications to more complex decision support applications, including optimization, forecasting, simulation, and strategic planning. The wide range of applications and base of end-users has made spreadsheets a universal modeling platform. By developing an application in the spreadsheet environment, the model can be circulated among a wide range of users without having to worry about specialized software packages and learning curves. Although the different spreadsheet packages and versions are not identical, they are compatible enough that they can support the basic function of each other's models and thereby not hurt the widespread use of a developed model. The examples used throughout this article have all been modeled with Excel 2003.

SPREADSHEET FEATURES

A modern spreadsheet consists of a series of worksheets, each of which contains a grid of rows and columns. Each cell can contain data in various formats, typically numbers, text, dates, or formulas, which state the relationships between the contents of other cells. Many standard mathematical, logical, statistical, engineering, and financial operations are available as built-in functions, and these can be combined to express complex relationships.

The spreadsheet interface is very user friendly with many shortcut keyboard key combinations and special buttons and an undo and redo feature. Commands (such as save, delete, insert, edit, graph, format, copy, move, and sort) can be applied to individual cells, ranges of cells, or the whole file so that the spreadsheet can be customized as required. The level of customization and sophistication of the

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*.

resulting spreadsheet model readily accommodates the requirements of the novice and expert user alike: and therein lies the secret of spreadsheet popularity and extensive user base. For example, a novice user would simply enter data on which calculations are performed, whereas a more advanced user would link to the original data, perhaps in other files or, after sorting or extracting from a database. A novice user would create formulas that referred to cells by their row and column references (e.g., F23), whereas a more advanced user would create range names for single cells or blocks and use absolute and relative notation judicially so that the spreadsheet formulas would not be compromised by any future dimensional changes or reorganization. Also, the advanced user may create formulas using meaningful data headings (as opposed to cell addresses or range names); e.g., =sum(Western) will sum the column called "Western." The advanced user may also create larger and more complex models that require circular references (a situation that is common, for example, with financial proforma models) and need Excel's Calculation Iteration menu option to resolve the circularity.

Data in the spreadsheet can be graphed in a variety of formats [e.g., line graphs, *x*—*y* scatter plots, pie, area, and bubble charts, in two-dimensional (2-D) and three-dimensional (3-D) representations]. The graphs can be customized by logarithmic and scaled axes, mixed format graphs (e.g., bar and line graphs) and displaying points, lines, and backgrounds in different colors and textures. Data in the spreadsheet can be treated as a flat 2-D database, which can be queried and reports can be produced. Pivot tables allow the user to summarize data in a variety of arrangements by providing cross-tabulations of data and summary statistics.

All spreadsheets today include a powerful programming language (e.g., Visual Basic for Applications, VBA, in Excel), which in effect provides an unlimited forum for user customization. For novice users, no programming or even knowledge of the existence of VBA is required to record and store sequences of keystrokes and commands so that they can be played back at a later time. In this way, repetitive tasks can be automated. However, a user who can program in VBA can build special-purpose applications with their own look and feel (i.e., menus and commands) and that involve decision points, branching, loops, and user interaction. User-defined functions can also be created using VBA to complement the existing library of available Excel functions.

Other enhancements include features that help document and control a model's integrity such as cell annotation, graphical pointers to succeeding and preceding cells (to help with debugging and understanding complex spreadsheets), and scenario management where complex "what-if" scenarios can be organized and tracked. These features help users create more structured models rather than the "quick and dirty" models that have been historically built with spreadsheets and that are becoming more unacceptable as the spreadsheet medium becomes the universal base for more multi-user-oriented models.

LIMITATIONS OF SPREADSHEETS AND SPECIAL CONSIDERATIONS

The availability and extensive use of spreadsheets in all walks of life has spurred concern regarding the accuracy and integrity of the results produced by a spreadsheet. Large organizations have thousands of spreadsheets distributed across the enterprise that have been developed by independent end-users in an uncontrolled environment. Ironically, it is the same ease of use and availability of spreadsheets, which makes them so popular, that also makes them susceptible to errors. Many accounts of errors in spreadsheets exist [see, for example, Panko (2) and Cragg and King (3)], and empirical studies have found that up to 90% of all spreadsheets in an organization contain errors, often of a costly nature. Laboratory and field studies have found that spreadsheet developers make errors in 2% to 5% of all formulas, regardless of their experience. These mistakes can range from mechanical errors (such as referring to a wrong cell or entering a wrong number) to logic errors (such as entering a wrong formula). Both the error rate and the impact of the inaccuracy increase with the complexity of the model. Galletta et al. (4) describe how it is difficult to detect errors once created, partly because of the highly polished presentation of the results afforded by the spreadsheet. H.M. Customs and Excise (5) states that "detailed testing can be extremely laborious" even with specialized spreadsheet auditing software. Section 404 of the Sarbanes-Oxley Act of 2002 targets this accuracy problem by mandating that firms increase controls related to the development and maintenance of spreadsheets. Sarbanes-Oxley requires companies to be able to justify what has happened to the data it presents in its corporate accounts and how it got there. This legislation along with the high risk of spreadsheet errors has generated a recent increased focus on auditing tools and processes as well as on spreadsheet management within organizations.

Spreadsheet programs lack the embedded logic and data controls necessary to prevent errors, and organizations must apply manual or automated control processes to help mitigate these risks. For developed models, this requires processes for controlling changes to a spreadsheet, maintaining input data integrity, documenting functions and objectives, and controlling access to the most recent version of the model. Martin (6) suggests that high-risk spreadsheets be converted into server-based applications to provide automated control measures. Current research [e.g., Kruck (7) and Freeman (8)] focuses on designing new techniques, expanding testing and inspection procedures, and adapting general programming techniques, such as the System Development Life Cycle, to improve the initial development of accurate spreadsheets. The HM Customs and Excise report (5) outlines procedures for assessing the risk that is associated with each spreadsheet so that the organization can concentrate upon auditing the spreadsheets that have the largest implications for the business.

Some limited auditing tools do currently exist in Excel, but they merely display the dependencies of cells on other cells; it is up to the user to determine whether these are in error. More sophisticated add-in auditing tools, such as Spreadsheet Advantage, Spreadsheet Professional, and

XL Analyst, are now available that provide the ability to identify differences between two versions of a spreadsheet model, map out the structure of each worksheet and blocks of cells that contain the same formulas in a model, identify circular references, and analyze the structure and complexity of a spreadsheet. Auditing protocols still need to be implemented to utilize these tools to find errors in the most efficient, reliable, and effective way possible [e.g., Butler (9)]. Cragg and King (3) suggest that the first step is to "communicate the fact that there are serious problems with independent, uncoordinated and undisciplined approaches to spreadsheet development which can lead to managers making decisions based on dubious data."

PROFILES OF TYPICAL USERS

The literature contains many discussions on the virtues and benefits of the spreadsheet environment, e.g., Pirlot (10), Roy et al. (11), Vazsonyi (12), Carraway and Clyman (13), and Powell (14). Spreadsheets provide a natural interface for model building; are easy to use in terms of inputs, solutions, and report generation; and allow users to perform what–if analysis. Bodily (15) stated that these key spreadsheet properties could provide a stepping stone for end users to the operations research/management scientist (OR/MS) discipline. The spreadsheet serves as a point of convergence for the non specialist user, who through spreadsheets has discovered modeling and its benefits, and the OR/MS specialist, whose models previously lacked the immediacy and impact necessary to respond to the end users' needs. Bodily (15) identified prospects for OR/MS tools in the spreadsheet medium and predicted that the convergence of the end user and the OR/MS specialist in these areas would form a powerful union that would ultimately result in greater rigor in model building and improved productivity.

Today, spreadsheets are the de facto modeling medium for OR/MS educators and researchers. Most, if not all, introductory OR/MS texts are now spreadsheet based [e.g., Balakrishnan et al., (16), Moore and Weatherford (17), and Ragsdale (18)]. The almost unanimous adoption of spreadsheets in OR/MS education by about 2000 prompted Gass et al. (19) to argue against the benefits of spreadsheets in OR/MS courses, where they state that "striving to get the spreadsheet right is taking precedence over learning what is right in modeling." Others [Seal and Przasnyski (20) and Troxell and Aieta (21)] have commented that too much class time is often spent on teaching tools or software, which detracts from concentration on OR/MS concepts. Another concern is that the powerful tools now potentially at the end users' disposal may undervalue the simple tool for the simple task [e.g., Berry (22)].

The final spreadsheet users are increasingly often the model builders. Spreadsheet models provide a widely understood format and have a more natural interface than algebraic models. The end users therefore have greater confidence in the models and in model generation. Solution procedures are readily integrated, and they offer decision support system (DSS) facilities and automatic what–if analysis. A survey of practitioners by Leon et al. (23) showed that a variety of OR/MS tools are being used in spreadsheet applications by end users across a wide spectrum of functional areas; see Figs. 1 and 2.

A literature analysis of the application of OR/MS tools in spreadsheet models by Seal et al. (24), classified applications by the OR/MS tools used, the functional areas involved, and the level of implementation performed. The level of implementation was categorized into three types of papers. In Type 1 papers, the spreadsheet model was implemented and used by a distinct and well-defined client and the papers included a description of the model and an account of the improvements or effects of implementation. In Type 2 papers, the spreadsheet model was implemented to address a problem or issue raised or generated specifically by the researchers. The resulting model was documented and reproducible, but it was not implemented to solve a client's specific problem. Type 3 papers described or proposed a small or trivial spreadsheet model.

Table 1 shows the number of papers describing spreadsheet implementations by functional areas and points to the ubiquity of the spreadsheet. Although the bulk of Type 1 spreadsheet implementations was in manufacturing and administration, several other functional areas are well represented. In the same paper, the authors observed a steady increase over time of Type 1 papers using the most popular OR/MS tools, namely decision support systems, mathematical programming, inventory, simulation, statistics, and forecasting. The strength of the spreadsheet medium lies in providing end users with a dynamic decision-making environment and the aforementioned tools are quite well suited for that purpose as evidenced by the fact that most applications were developed not to solve the traditional static OR/MS problem but to support a client's dynamic decision-making process. The most significant motivations or benefits identified for using spreadsheets in these studies were: 1) the dynamic sensitivity analysis or "what–if" capabilities, 2) the user-friendly interface, 3) end-user familiarity with the spreadsheet environment, 4) the integrative capabilities, and 5) the ease of modeling that exists because of a spreadsheet's flexibility with its selection of modeling tools. In most cases, the spreadsheet models resulted in greater productivity, just as Bodily (15) anticipated. The flexibility, user friendliness, and availability of the interface were perceived very positively, and the resulting implementations usually claimed significant improvements in productivity and efficiency as measured by various yardsticks particular to that application.

However, despite the documented successes, the use of OR/MS tools in spreadsheets may not be appropriate for all cases, and the everyday use of hitherto specialized tools by end users is not without some reservations. Spreadsheets may be perceived as too limited or too slow for large or complex applications, or such applications could require excessive (VBA) programming. Indeed, it may simply be easier to use an established specialized package rather than to build and validate a complex spreadsheet model for certain types of problems. Although many authors extol the virtues of spreadsheets, some at the same time warn that "certain applications are predisposed for spreadsheet treatment and others are not" [for example, Freeman (25)]. Several authors stress that the strengths of these approaches are

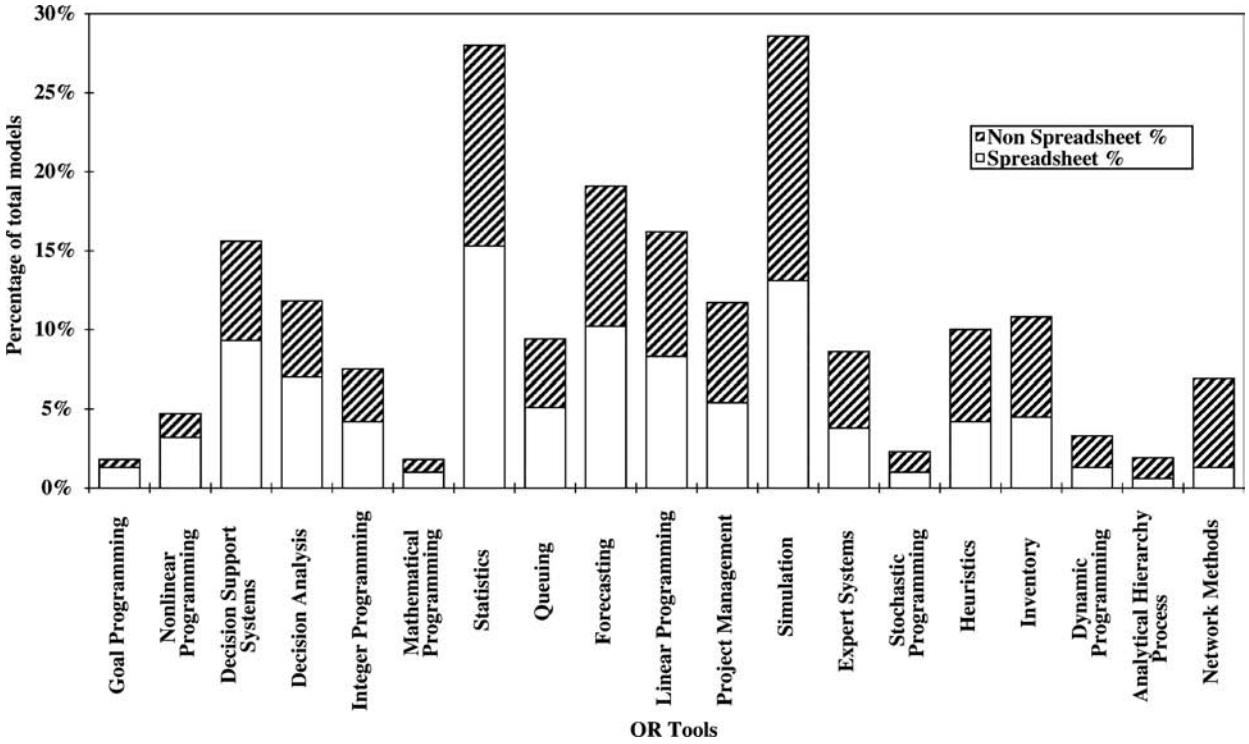

Figure 1. The OR tools arranged in decreasing order of the ratio of percentages between spreadsheet and non-spreadsheet models show that OR tools are being used in the spreadsheet environment. The non-spreadsheet percentage is equal to the number of non-spreadsheet models using the OR tool divided by the total number of non-spreadsheet models. The spreadsheet percentage is equal to the number of spreadsheet models using the OR tool divided by the total number of spreadsheet models.

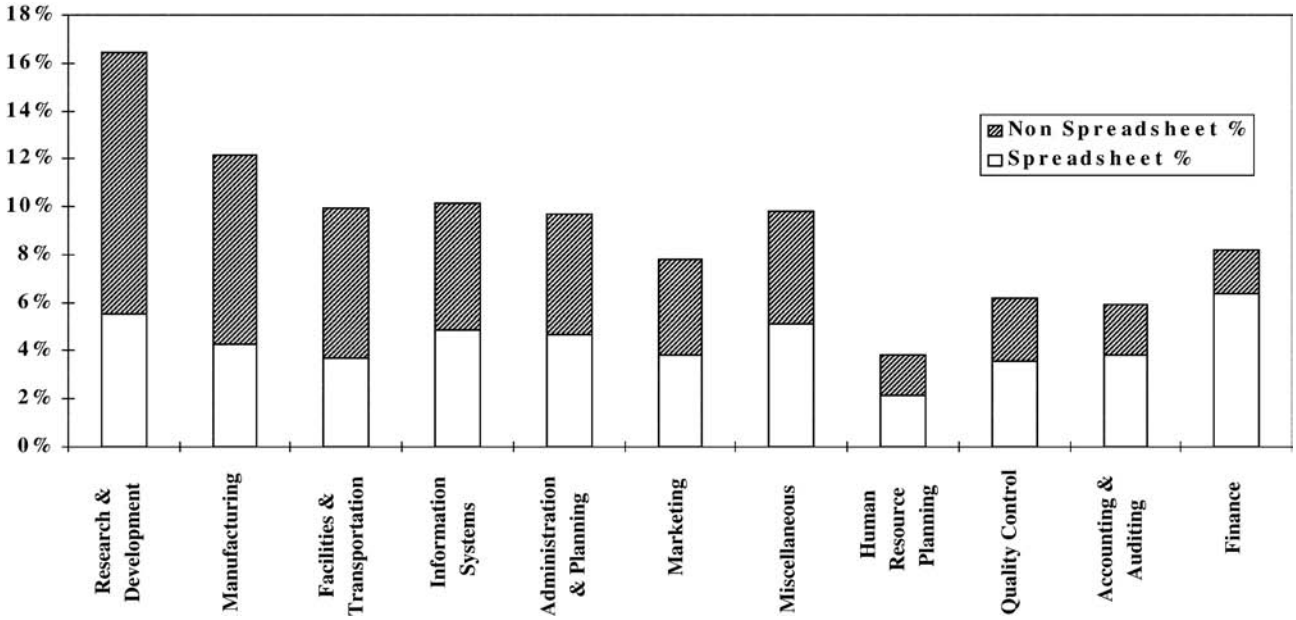

Figure 2. The percentage of total spreadsheet and non-spreadsheet implementations across 11 functional areas shows acceptance of spreadsheets across all functional areas. The non-spreadsheet percentage is equal to the number of non-spreadsheet models in each functional area divided by the total number of spreadsheet and non-spreadsheet models for all areas. The spreadsheet percentage is equal to the number of spreadsheet models in each functional area divided by the total number of spreadsheet and non-spreadsheet models for all areas.

Table 1. Functional Areas Where Spreadsheets Were Used (Sorted by Number of Type 1 Papers)

Functional Area	Number of Type 1 Papers	Number of Type 2 and 3 Papers
Manufacturing	25	51
Administration	10	4
Finance	7	24
Other	6	6
Transportation	5	4
Accounting	5	10
Research and Development	5	2
Human Resources	4	5
Marketing	3	1
Information Systems	2	1
Quality Control	1	4
Education	0	3
International Business	1	0

Table 2. Options Available in Excel's Solver

Option	Description
Precision	Specifies how near to each other two trial solutions must be before an optimal solution is declared.
Estimates	Additional solution methods are Tangent and Quadratic. Use Quadratic if the worksheet involves complex formulas that are highly nonlinear.
Derivatives	Specifies the method of partial derivatives, using Forward or Central differencing. Central differencing can take longer but may result in a closer solution.
Search	Specifies a quasi-Newton or Conjugate gradient method of searching.

the decision *aid* as opposed to the decision-*making* aspects [e.g., Pirlot (10) and Roy et al. (11)]. Some users expect a "black box" solution and get frustrated by the required interactions or questionable outputs. Concern exists that providing such powerful tools to the novice without sufficient training can result in misuse of a model or misinterpretation of the results, thereby producing erroneous or low-quality decisions [e.g., Troxell (26)].

Starting in 2001, organizations such as the Spreadsheet Productivity Research Interest Group (SPRIG) of The Institute for Operations Research and the Management Sciences (INFORMS) have been established to study the discussed issues and limitations associated with spreadsheet productivity. The mission of the Spreadsheet Productivity Research Interest Group (http://sprig.section.informs.org/) is to "inspire, support, promote and disseminate innovations in practice, research and teaching related to the use of spreadsheets and other end-user software for modeling and analysis. SPRIG will develop and maintain close relationships with non-academic spreadsheet leaders."

DSS TOOLS AND APPLICATIONS

Mathematical Programming

Roy et al. (11) described spreadsheet optimization applications developed by end -users with little or no previous optimization experience. They concluded that many people who are unfamiliar with optimization methods and are uncomfortable with algebraic models can formulate and solve large, real-life problems in spreadsheets without an OR/MS specialist. This observation has been substantiated by other researchers [Pirlot (10), Powell (14), Carraway and Clyman (27)]. The easiest type of problem to solve is a linear programming (LP) model based on the efficient Simplex solution algorithm for this class of problems. This calculation is now an integral part of Excel as Solver, originally developed by Frontline Systems. Many problems can be modeled that meet the linearity assumptions required for a LP problem [see any standard OR/MS text books, e.g., Taha (28), Anderson et al. (29), or Ragsdale (18)].

One of the main advantages of spreadsheet-based optimization models is that the models are created in a format that is natural to the end-user as opposed to algebraic expressions that may not be so familiar or understandable for many end-users. The results are reported in this same intuitive format familiar to the user as opposed to a typical LP package output format, which must be interpreted and reorganized into meaningful information for the user. As a spreadsheet model is often initially built to answer what–if questions, the model may be then optimized using Solver as a follow-through or additional analysis.

As an example, consider the classic multiperiod production scheduling problem. A manufacturer has forecast the demand for a product for the next six months along with the monthly sales prices and manufacturing and holding costs. The firm can produce as many units as it wants in any given period with a one-month lead time (i.e., units pro-

	A	B	C	D	E	F	G	H
1	**MONTH**	**January**	**February**	**March**	**April**	**May**	**June**	
2	Target Sales	80	90	80	60	50	50	
3	Sales Price per Unit	$80	$80	$60	$70	$80	$90	
4	Manufacturing Cost per Unit	$60	$60	$50	$60	$70	$75	
5	Holding Cost per Unit	$2	$2	$2	$2	$2	$2	
6								
7	Beginning Inventory	100	100	90	60	70	50	
8	***Number of Units Sold***	***80***	***90***	***80***	***60***	***50***	***50***	
9	***Number of Units Produced***	***80***	***80***	***50***	***70***	***30***	***0***	
10	Ending Inventory	100	90	60	70	50	0	
11	Inventory Capacity	100	100	100	100	100	0	
12	Minimum Inventory	10	10	10	10	10	0	
13								**TOTAL**
14	Revenue	$6,400	$7,200	$4,800	$4,200	$4,000	$4,500	**$31,100**
15	Cost	$5,000	$4,980	$2,620	$4,340	$2,200	$0	**$19,140**
16	**Profit**	**$1,400**	**$2,220**	**$2,180**	**-$140**	**$1,800**	**$4,500**	**$11,960**

Figure 3. Six-month multi-period production model. The user varies the data in rows 8 and 9 by trial and error to determine the overall profit.

duced in January are available for sale in February), but its operation is limited by the size of its warehouse, which can hold a maximum of 100 units. The company would like to keep a safety stock of 10 units each month, except for the last month where it would like to reduce inventory to 0. The problem is to determine how many units to produce and sell each month so as to maximize the six-month total profit. The basic spreadsheet model for this problem without considering optimization is shown in Fig. 3. With this basic model, the user can experiment with different numbers of units sold and produced by changing the cells in row 8 and 9 and can watch the impact these decisions will have on total profit in cell H16 as well as on monthly inventory levels in row 10. While changing the production and sales quantities, the user will want to make sure that the inventory levels in row 10 do not drop below the minimum in row 12 or above the capacity in row 11. The user will also want to make sure that the units sold in any month do not exceed the beginning inventory for that month as the current month's production units will not be available until next month. After a certain amount of trial and error, the user may arrive at a solution similar to the one found in Fig. 3.

This basic spreadsheet model can be easily turned into an optimization model by setting up the Solver dialog box shown in Fig. 4, which communicates the nature of the constraints that the user was manually trying to enforce. Cell H16 is identified as the objective cell to maximize by changing the decision variable cells (B8:G9) subject to the cell constraints that follow. Using the Options button in Fig. 4, the non-negativity assumption for the decision variables and the assumption of a linear model can be checked off in the dialog box shown in Fig. 5. The solution that is obtained in the spreadsheet model as a result of running Solver is shown in Fig. 6. For advanced users, standard LP sensitivity analysis output (i.e., shadow prices and reduced costs) can also be generated on new worksheets in the workbook by selecting the appropriate options when Solver displays the message that it has found a solution.

The natural reporting format of the spreadsheet makes it easy for users to identify mistakes in the optimization model logic and makes the necessary corrections. For example, if the user had not originally entered the constraint B8:G8<=B7:G7 (do not sell more than the on-hand amount at the beginning of each month), the solution obtained would be as shown in Fig. 7. The user should realize that s/he cannot sell the number of units shown in February and March as the units produced in those months will not be available until the next month. The user would then need to make an adjustment to handle this constraint. This type of insight would not be possible with traditional algebraic solver packages.

Some disadvantages exist to using spreadsheet optimization models. The standard Solver package currently built-in with Excel can currently handle up to 200 variables and 200 constraints on one worksheet. Upgrades can be purchased from Frontline Systems, Inc., the developer of Solver (http://www.solver.com), which solve much faster and accommodate models that are spread across multiple worksheets with up to 8000 variables and 8000 constraints. With Frontline's more powerful spreadsheet-based platforms and assortment of specialized Solver engines, spreadsheets can handle more large-scale systems, with some users reporting successful implementations of LP problems that included 2.4 million variables. A shortcoming for large-scale problems, however, still is the absence of indexing and dimensioning capabilities with spreadsheets. Unlike optimization packages such as GAMS, which allow a modeler to dimension variable indices easily, the spreadsheet model must be manually created and modified. Laying out a worksheet carefully can help to a certain extent in that some dimension changes can be easily done by inserting/deleting columns or rows and copying formulas. However, this process is prone to errors and not all modifications are a matter of simply copying existing formulas.

Other implementation problems exist with the simplicity of the spreadsheet interface and optimization models that many novice modelers may encounter. First, novice

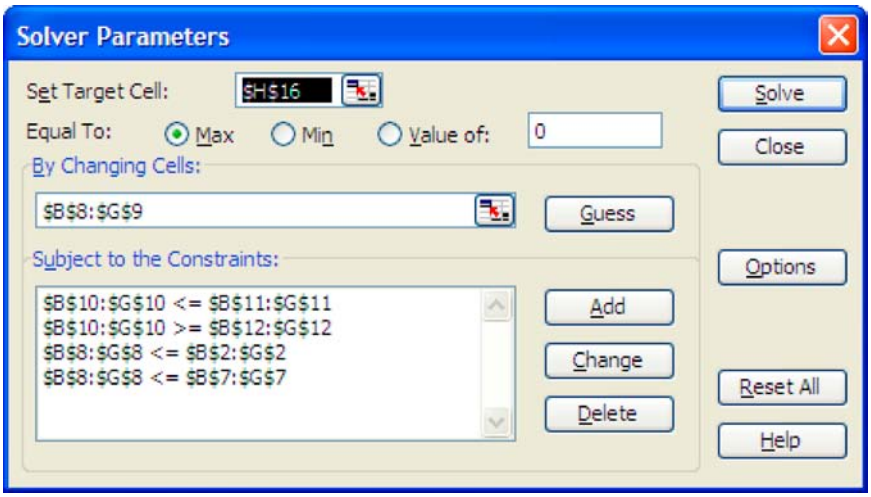

Figure 4. Solver dialog screen for the six-month multi period production model shows the objective to be maximized and the constraints.

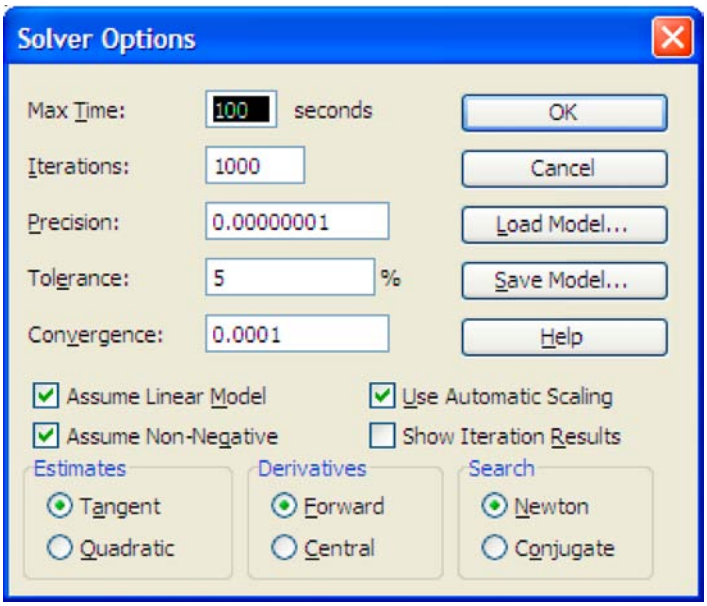

Figure 5. The "options" screen in Solver can be used to choose and control the type of solution procedure.

	A	B	C	D	E	F	G	H
1	**MONTH**	**January**	**February**	**March**	**April**	**May**	**June**	
2	Target Sales	80	90	80	60	50	50	
3	Sales Price per Unit	$80	$80	$60	$70	$80	$90	
4	Manufacturing Cost per Unit	$60	$60	$50	$60	$70	$75	
5	Holding Cost per Unit	$2	$2	$2	$2	$2	$2	
6								
7	Beginning Inventory	100	90	10	100	100	50	
8	***Number of Units Sold***	***80***	***90***	***10***	***60***	***50***	***50***	
9	***Number of Units Produced***	***70***	***10***	***100***	***60***	***0***	***0***	
10	Ending Inventory	90	10	100	100	50	0	
11	Inventory Capacity	100	100	100	100	100	0	
12	Minimum Inventory	10	10	10	10	10	0	
13								**TOTAL**
14	Revenue	$6,400	$7,200	$600	$4,200	$4,000	$4,500	**$26,900**
15	Cost	$4,380	$620	$5,200	$3,800	$100	$0	**$14,100**
16	**Profit**	**$2,020**	**$6,580**	**-$4,600**	**$400**	**$3,900**	**$4,500**	**$12,800**

Figure 6. The optimal solution to the six-month multi-period production model after being run through Solver.

	A	B	C	D	E	F	G	H
1	**MONTH**	**January**	**February**	**March**	**April**	**May**	**June**	
2	Target Sales	80	90	80	60	50	50	
3	Sales Price per Unit	$80	$80	$60	$70	$80	$90	
4	Manufacturing Cost per Unit	$60	$60	$50	$60	$70	$75	
5	Holding Cost per Unit	$2	$2	$2	$2	$2	$2	
6								
7	Beginning Inventory	100	20	10	100	100	50	
8	***Number of Units Sold***	***80***	***90***	***80***	***60***	***50***	***50***	
9	***Number of Units Produced***	***0***	***80***	***170***	***60***	***0***	***0***	
10	Ending Inventory	20	10	100	100	50	0	
11	Inventory Capacity	100	100	100	100	100	0	
12	Minimum Inventory	10	10	10	10	10	0	
13								**TOTAL**
14	Revenue	$6,400	$7,200	$4,800	$4,200	$4,000	$4,500	**$31,100**
15	Cost	$40	$4,820	$8,700	$3,800	$100	$0	**$17,460**
16	**Profit**	**$6,360**	**$2,380**	**-$3,900**	**$400**	**$3,900**	**$4,500**	**$13,640**

Figure 7. Solution to the six-month multi-period production model if the constraint "do not sell more than the on-hand amount at the beginning of each month" was omitted.

users may not realize that alternative optimal solutions exist as Solver does not automatically present this information. Turner et al. (30) describes an efficient and straightforward procedure that can be used to locate the alternative optimal solutions with Solver. Second, a poorly scaled model can create solution problems for Solver. Troxell (26) defines a poorly scaled model as "one in which the values used among the objective function and constraint functions, including the initial values for the algorithm, differ by several orders of magnitude." Severe build-up of round-off errors can occur during the solution process, generating error messages or, on rare occasions, sub-optimal solutions that are erroneously presented as optimal. Solver does have an option to Use Automatic Scaling (see Fig. 5), which will attempt to scale the values of the objective and constraint functions internally in order to minimize the effects of a poorly scaled model. Finally, it is easy to unintentionally create nonlinear spreadsheet models. Users who are not OR/MS experts do not realize that IF, MIN, MAX, and many financial spreadsheet functions are piecewise linear or nonlinear functions. Use of these functions in the optimization model's logic can make the problem nonlinear and identified solutions could include local optimal points as well as the global optimal point. Many users will not question the solution they get as they treat the model as a type of black box that will always give them the best answer. These are all examples of how powerful tools at the disposal of non-experts can result in erroneous models and interpretation of results.

In a nonlinear program (NLP), the objective function and/or at least one constraint of the problem will be a nonlinear function of the decision variables. For example, an IF function, which a modeler might use in a spreadsheet to model price discounts as a function of volume sold, can make the model nonlinear if these prices were used in the objective to calculate revenue. Nonlinearity is required for many engineering design applications, which often involve high-order polynomials or calculus. When a problem is nonlinear, the spreadsheet built-in optimizers must be switched from the stable, efficient simplex solver engine for LPs to a set of solver engines that will find the best solution given the current starting solution presented in the spreadsheet. Depending on the curve and the starting solution point used in the NLP, the optimal solution found may or may not be the global optimal solution [see any standard text, e.g., Ravindran et al. (31) for descriptions of nonlinear programming methodology].

As an example of a nonlinear programming spreadsheet model, consider the following engineering design application. A company wishes to build a steel tank to store 2500 cubic yards of gasoline. Steel costs $22.50 per square yard, and the company needs enough steel to cover the top, bottom, and side of a cylinder. What are the best dimensions so as to minimize the cost of steel used and provide the necessary volume? Obviously the amount of steel required to build the cylinder is $2\pi R^2 + 2\pi RH$, which is a nonlinear function of the radius R and the height H. The resulting volume is $\pi R^2 H$, which is also a nonlinear function of R and H.

A spreadsheet model can be easily set up to solve this problem as a nonlinear program. In Fig. 8, the cylinder design is set up so that for any height or radius entered in cells D5 or D6, the necessary amount of materials needed to cover the different parts of the cylinder (cells D7 to D9) and the resulting volume (cell D11) are calculated. The resulting costs are shown in cell D15. The user can experiment with different radiuses and heights to find a good initial starting point, such as the one shown in Fig. 8. This problem can then be converted in an NLP by running Solver as shown in Fig. 9. The objective is to minimize the cost of steel needed, in cell D15, subject to the constraint that the resulting volume in cell D11 be greater than the required capacity in cell C13. The Non-negativity Assumption option must also be selected, but the Assume Linear Model option should be turned off. Solver will then use the selected search options shown in the bottom of Fig. 5 along with the starting point (such as the one in cells D5 and D6 of Fig. 8) to determine the optimal solution shown in Fig. 10. Table 2 provides a brief description of the options shown in Fig. 5, which are available with the Solver en-

	A	B	C	D
1	***Tank Design Example***			
2				
3	CYLINDER DESIGN:			
4				
5		Height (in yds):		9.5
6		Radius (in yds):		9.5
7		Top:		283.53
8		Bottom:		283.53
9		Sides:		567.06
10		Total Area:		1134.11
11		Volume:		2693.52
12				
13	Required Capacity:		2500 cubic yards	
14	Steel Cost per Squared Yard:			$ 22.50
15	Total Cost of Proposed Cylinder:			$25,517.59

Figure 8. Nonlinear programming model: engineering design of cylindrical gasoline tank.

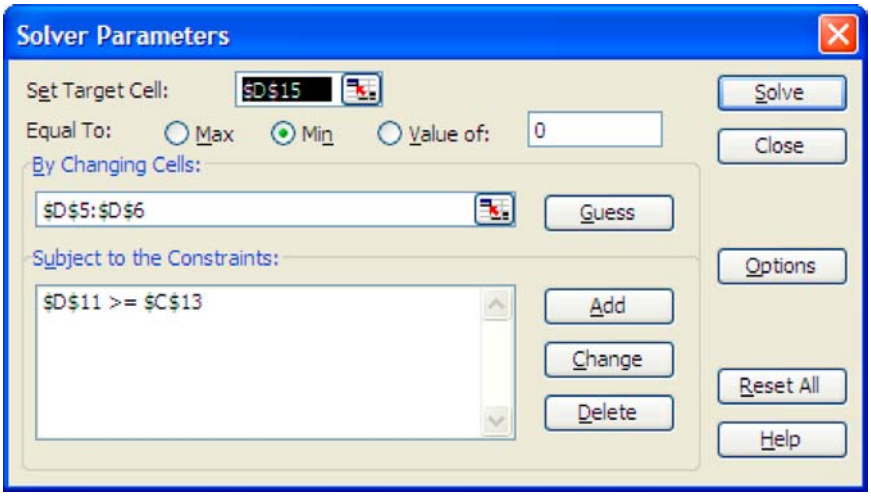

Figure 9. Solver dialog screen for the engineering design of a cylindrical gasoline tank model shows the objective to be maximized and the constraints.

	A	B	C	D
1	***Tank Design Example***			
2				
3	CYLINDER DESIGN:			
4				
5		Height (in yds):		14.71
6		Radius (in yds):		7.36
7		Top:		169.95
8		Bottom:		169.95
9		Sides:		679.80
10		Total Area:		1019.71
11		Volume:		2500
12				
13	Required Capacity:		2500 cubic yards	
14	Steel Cost per Squared Yard:			$ 22.50
15	Total Cost of Proposed Cylinder:			$22,943.36

Figure 10. The optimal solution to the engineering design of the cylindrical gasoline tank model after being run through Solver.

gines in Excel.

In this example, the local optimum identified is the global optimum. This problem is a relatively simple NLP on which the gradient search solvers can quickly converge to the one optimum from a reasonable starting point. If the starting point had included both a radius and a height of 0, Solver would have difficulty converging and would result in error messages. It is important, therefore, for the user to

understand the significance of the starting solution shown in the spreadsheet model before Solver is run. It will determine whether a solution will be found and how many iterations it will take for that solution to be found. In cases where there are several local optima, the starting point will also impact the chances of identifying the global optimum.

The spreadsheet has tools that can be useful for identifying good starting points. By creating the NLP in a spreadsheet environment, the user has a model on which what –if analysis can be easily performed to see how possible solutions will perform. The Data Table feature of spreadsheets can be used to help automate this what –if process and collect information on output results for specified input values. For example, in the cylinder design illustration, the two-way data table structure shown in Fig. 11 can be set up with different test radii and height sizes listed in the first row and column of the table. A Data Table menu option systematically inputs the different combinations of radii and heights specified in the first row and column of the data table into cells D5 and D6 of the NLP model and records the resulting volume as a table entry for the corresponding row and column. For example, cell D23 in Fig. 11 is the volume that will occur if a 14-yard high and 8-yard radius cylinder is built. From the data table, the user can see the range of design configurations that will generate sufficient capacity of 2500 cubic yards. A similar table could be set up for different radii and heights to see possible impacts on the cost of steel. Based on these results, the user could identify a good starting point for R and H.

For larger problems with more variables and constraints, the same initial prescreening can be done to change more inputs simultaneously by using scenario manager in Excel. This would allow the user to generate different what–if scenarios for the decision variables, summarize their possible results, and make comparisons to identify good starting points. Finally, Frontline's upgrades for Solver have a MultiStart Search option that will resolve the starting point problem just described so that the user does not have to use Data Tables or other tools to test different starting points. Upgraded versions of Solver also include an evolutionary solver engine, which uses genetic algorithm methods such as mutation, cross over, selection, and constraint repair to identify a solution that is better in comparison with other known solutions. It does not guarantee that the solution it finds is an optimal solution, even though it often identifies the correct optimal solution. Genetic algorithms work well for models that contain the nonlinear spreadsheet functions that can derail linear and nonlinear solvers as well as for the scientific applications that involve calculus and high-order polynomials, such as maximizing response rates or yields in a process.

One main disadvantage of using NLP models in the spreadsheet environment is the amount of interaction required on the part of the user to determine the global optimum. Most spreadsheet users are looking for a black box solution similar to LP models. Users do not always understand why it can take so long to get a solution and are often frustrated by the strange solutions that the solver engines can generate. They do not know how or do not want to spend the required time to find a good starting point nor to experiment with different solver engines. For many novice users, the required interaction on their part overshadows the benefit of using this technique. Some advanced modelers would prefer to use a specialized package with which they are already familiar and where the search algorithms can be customized to their application requirements so as to give them more power and speed. Nevertheless, improvements in technology and advances in Frontline's solver engines have made spreadsheets more appealing to advanced modelers in recent years.

Integer programming is another mathematical programming tool available in the spreadsheet environment. Converting an LP into an integer program or mixed integer program involves adding a solver constraint that certain changing cells (decision variables) be integer or binary. Once again the user is given a powerful tool, the branch-and-bound search algorithm, in a black box that one must be careful not to misuse. Every extra variable specified as an integer may require a significant amount of additional computation time necessary to solve the problem. Problems that could be solved immediately without integer restrictions can end up taking significant amounts of time to solve even on systems with fast processors and large amounts of RAM or never deliver a final recommendation at all depending on the number of integer variables selected. The standard Solver is currently limited to 200 integer constraints where upgraded Solver versions can handle up to 2000 integer constraints. Adjusting Solver options, such as the tolerance option in Excel's solver, can help speed up calculations at the expense of identifying a limit on the suboptimal solution that is acceptable.

Some mathematical programming applications such as capital budgeting, scheduling, fixed charge, uncapacitated facility location, and cutting stock problems exist where integer programming techniques are necessary to model key relationships. Baker and Camm (32) have studied how integer programming problems can be remodeled in spreadsheets with nonlinear functions and then solved with the evolutionary solver. They found that evolutionary solver works better for some types of applications than others, with better being defined as the percent of time that it finds the optimal solution as well as the average suboptimality. They even found one application type, the weighted tardiness problem, where the spreadsheet model solves faster and better with evolutionary solver than by integer programming.

Simulation

Monte Carlo simulation is a tool that aids decision-makers by understanding the uncertainty involved in a decision better. It is a descriptive technique as opposed to optimization, which is prescriptive. Given a possible course of action, simulation can be used to describe the risk in the strategy by considering many sources of uncertainty simultaneously. A spreadsheet with its natural format and its what–if capabilities is an ideal medium for this tool for many small-to medium-size applications. For larger and more complex applications such as facility layout problems or process strategy decisions, a specialized simulation language such as Simula or SIMSCRIPT or a commercial, simulation program with animated graphical interfaces such

	A	B	C	D	E	F	G
1	***Tank Design Example***						
2							
3	CYLINDER DESIGN:						
4							
5		Height (in yds):		14.00			
6		Radius (in yds):		8.00			
7		Top:		201.06			
8		Bottom:		201.06			
9		Sides:		703.72			
10		Total Area:		1105.84			
11		Volume:		2814.87			
12							
13	Required Capacity:		2500 cubic yards				
14	Steel Cost per Squared Yard:			$ 22.50			
15	Total Cost of Proposed Cylinder:			$24,881.41			
16	***DATA TABLE:***						
17	*HEIGHT*			*RADIUS*			
18	2814.9	2	5	8	11	14	17
19	2	25.1	157.1	402.1	760.3	1231.5	1815.8
20	5	62.8	392.7	1005.3	1900.7	3078.8	4539.6
21	8	100.5	628.3	1608.5	3041.1	4926.0	7263.4
22	11	138.2	863.9	2211.7	4181.5	6773.3	9987.1
23	14	175.9	1099.6	2814.9	5321.9	8620.5	12710.9
24	17	213.6	1335.2	3418.1	6462.3	10467.8	15434.6

Figure 11. Two-way data table for the engineering design of a cylindrical gasoline tank model evaluates the volume for various combinations of the cylinder's radius and height.

as SLAM II or Micro Saint would be more appropriate. These alternatives are costly and involve a much steeper learning curve, however. For smaller applications, therefore, the spreadsheet is a user-friendly option to consider and many successful examples of spreadsheet simulations are recorded in the literature, for example, Bookbinder (inventory and distribution in the fine paper industry) (33), Pope and Cross (load size and routing schedule for ocean shippers) (34), Bobby (cost structure of an oil distributor) (35), Schuster and Finch (production scheduling and inventory management in a juice manufacturing company) (36), and Poshyanonda et al. (scheduling for a feed mill company) (37).

To introduce uncertainty into an existing spreadsheet model, the user must quantify risk distributions for the input cells, which are uncertain. Spreadsheets have built-in random number generators to help do this. To simulate a random number between 0 and 1, a user must simply enter the RAND() spreadsheet function into a cell. Alternatively, the RANDBETWEEN function will generate a random integer between any specified lower and upper limit. Random numbers can be used with other spreadsheet functions, such as NORMINV, to simulate values from specified distributions. If the user's probability background is strong enough, the user can also define formulas to describe other distribution functions, with the discrete, triangular, or betapert distributions being examples of some of the easier distributions to create. By pressing the F9 recalculate key, the user will simulate possible scenarios of inputs and be able to view the resulting outputs directly in the model layout, one iteration at a time.

To collect data about the possible outcomes and summarize the data into meaningful risk analysis information, the data table option can be used. Each row of the defined data table can be used to represent an iteration, with the column headings representing the key outputs that the user would like to understand better. The data collected in the table can then be summarized with the spreadsheet's descriptive statistics menu options to communicate the potential risks of the strategy being considered.

If the user is serious about using simulation in a spreadsheet environment, there are worthwhile add-in packages such as Decisioneering Inc.'s Crystal Ball and Palisade Corporation's @Risk that can be used to help automate the simulation calculations. These packages provide numerous programmed input distributions, a user-friendly interface for setting simulation options, and detailed statistical output and graphs, thus removing most of the calculation burden from the user.

A risk analysis can be performed on the production scheduling problem of Fig. 6 as an example of the decision aid provided by Crystal Ball. Using, for example, the optimal production solution found earlier by LP and Solver, the model is modified to incorporate uncertainties regarding the target sales values shown in row 2 of Fig. 6. Assuming that the target sales are normally distributed each month with mean and standard deviation as shown in rows 2 and 3 of Fig. 12, the Crystal Ball formula =CB.Normal(B2,B3) entered in cell B4 and then copied through to G4 will randomly generate target sales for each month from the assumed distributions. [Without Crystal Ball, the formula *=NORMINV(rand(),mean target sales, standard deviation of target sales)* each month could be used instead.] Alter-

	A	B	C	D	E	F	G	H
1	MONTH	January	February	March	April	May	June	
2	Target Sales - mean	$80	$90	$80	$60	$50	$50	
3	Target Sales - std dev	$10	$10	$10	$10	$5	$5	
4	Target Sales	$77.82	$97.87	$79.91	$67.38	$52.92	$48.32	
5	Sales Price per Unit	$80	$80	$60	$70	$80	$90	
6	Manufacturing Cost per Unit	$60	$60	$50	$60	$70	$75	
7	Holding Cost per Unit	$2	$2	$2	$2	$2	$2	
8								
9	Beginning Inventory	100	92	10	100	93	40	
10	*Number of Units Sold*	78	92	10	67	53	40	
11	*Number of Units Prod uced*	70	10	100	60	0	0	
12	Ending Inventory	92	10	100	93	40	0	
13	Inventory Capacity	100	100	100	100	100	100	
14	Minimum Inventory	10	10	10	10	10	0	
15								TOTAL
16	Revenue	6,225.83	7,374.17	600.00	4,716.85	4,233.99	3,572.24	**26,723.08**
17	Cost	4,384.35	620.00	5,200.00	3,785.23	79.38	0.00	**14,068.97**
18	**Profit**	**1,841.47**	**6,754.17**	**-4,600.00**	**931.62**	**4,154.61**	**3,572.24**	**12,654.11**

Figure 12. Six-month multiperiod production scheduling model where target sales are distributed normally.

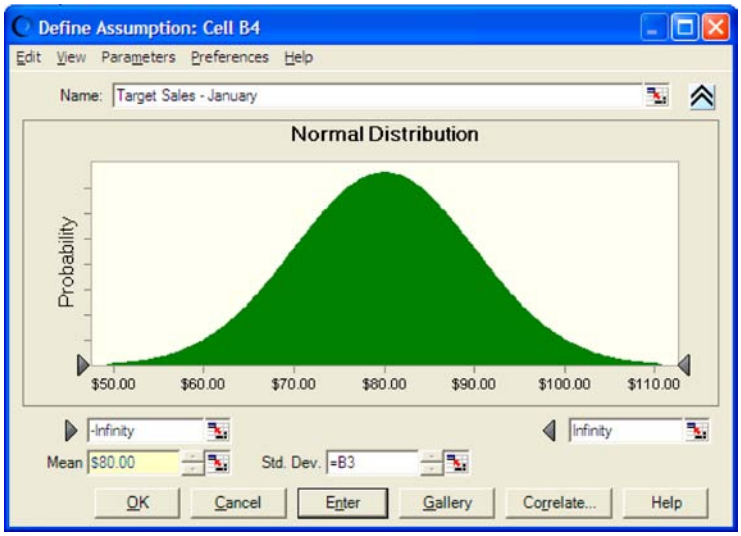

Figure 13. Crystal Ball input definition window showing how the normally distributed target sales for January are entered.

natively, Crystal Ball can provide more visual assistance by entering cell values for the distribution parameters through interactive input windows as shown in Fig. 13. The number of units sold in row 10 of Fig. 12 now has a formula entered that will modify the solution to changes in target sales in order to keep the solution realistic and feasible. For example, if demand is lower than expected, the number of units sold will have to be lowered to that amount. Figure 12 shows the results of one possible iteration with these modifications.

The user can then select the total profit in cell H18 and the ending inventories in cells B12:G12 as the outputs to be studied and run the simulation (default is 1000 iterations but can be extended to any number) and view the numerous descriptive statistics that will be collected for these variables. All resulting statistics (as well as the raw data from each iteration) are tabulated in spreadsheet cells for additional analysis or manipulation, if required. In addition, Figs. 14 and 15 show two types of graphical outputs that Crystal Ball will produce showing the possible ranges of profit and ending inventory levels that could occur if the user follows the production strategy prescribed by the LP solution strictly. Figure 14 shows the distribution of total profit and associated summary statistics based on 1000 trials. In Fig. 15, the user can see that a significant risk exists that the ending inventory in April will exceed the maximum capacity of 100 and that some sort of adjustment will need to be made. The exact probabilities and inventory levels can be found by viewing the detailed descriptive statistics provided by Crystal Ball and interacting with the results windows; e.g., the display in Fig. 14 shows that approximately a 46% chance exists that the total profit will be between $11,500 and $12,500.

In addition, an optimizing simulation can be run using the OptQuest module, which is a component of Crystal Ball Pro. Here, the user specifies permissible ranges for some decision variables (and whether the variable should be considered as continuous or discrete) and an objective

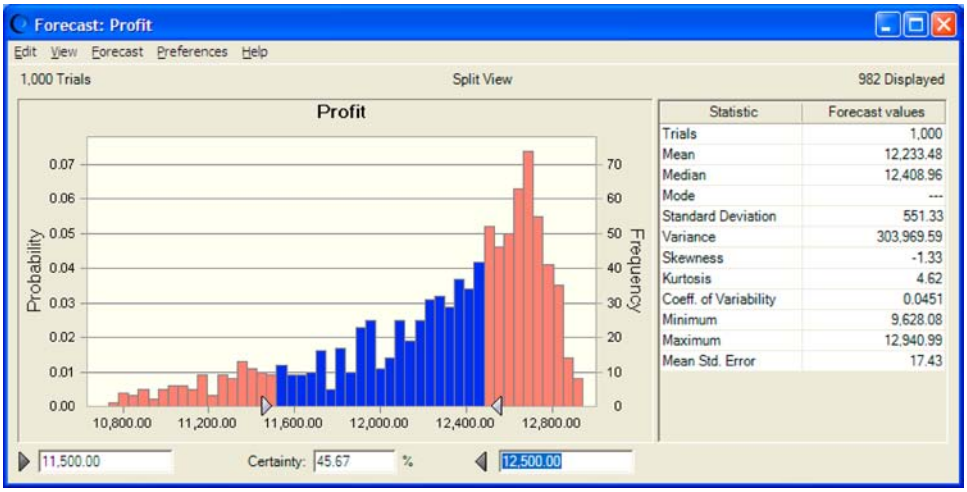

Figure 14. Crystal Ball output window showing the distribution of total profit for the simulation when target sales are normally distributed, in the six-month multiperiod production model.

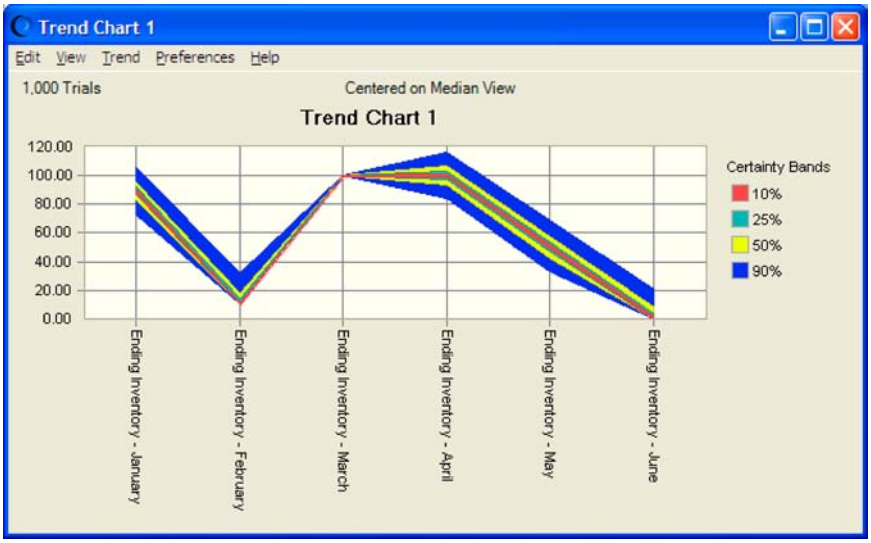

Figure 15. Crystal Ball Trend chart showing the trend of the ending inventory for the simulation when target sales are normally distributed, in the six-month multiperiod production model.

to be optimized (where a choice from a variety of statistical measures such as mean, median, mode, standard deviation, kurtosis, standard error, among others, can be made). OptQuest runs an entire simulation for combinations of values of the decision variables and chooses the best, until no improvement can be found or the search is terminated by the user.

An illustration of such an approach can be explained by an extension of the production planning simulation described above and presented in Fig. 12. Previously, we simulated the effect of uncertainty in the target sales on just one production strategy, whereas now we can define the number of units produced each month as decision variables and run OptQuest to maximize, for example, the median of the Total Profit. Running two OptQuest optimizing simulations may not yield exactly identical results because the underlying process is stochastic, so OptQuest ranks the near-optimal solutions it obtained according to the optimizing criterion to provide the user with insights into the underlying variability. One solution proposed by OptQuest is summarized in Fig. 16. Note that the optimal value for total profit is very close to that obtained in the original deterministic version of this problem obtained by Solver, as shown in Fig. 6. Although this production planning problem was simple enough to lend itself to both mathematical programming and simulation approaches, OptQuest provides a relatively straightforward approach to stochastic optimization and is a meaningful option for problems too complex to address by formal mathematical programming methods.

An alternative simulation approach has been integrated into the spreadsheet itself recently by Frontline Systems, the developers of Solver, which is currently in public beta testing. As with the Crystal Ball approach, the user specifies uncertain input cells based on well-known distributions or customized as required. The difference is that

OptQuest result	January	February	March	April	May	June
Number of Units Produced	70	10	100	60	0	0

Total Profit	
Mean	12,216.61
Median	12,406.86
Mode	---
Standard Deviation	582.94
Variance	339,815.4
Skewness	-1.51
Kurtosis	5.85
Coeff. of Variability	0.0477
Minimum	8,714.39
Maximum	12,967.79
Range Width	4,253.40
Mean Std. Error	18.43

Figure 16. Results after OptQuest optimization for the six-month multiperiod production planning model when target sales are normally distributed.

their Polymorphic Spreadsheet Interpreter (PSI) technology, which is also incorporated in the most recent version of Crystal Ball and the more advanced industrial strength Solver platforms, permits extremely fast spreadsheet updating, so that in Risk Solver a 1000 (or user specified) trial simulation is performed every time the spreadsheet is refreshed. The consequences for all the cells depending on the uncertain inputs are almost instantaneously summarized. Essentially a stochastic simulation is performed on each spreadsheet refresh, which allows the user to do almost instantaneous what–if analysis for uncertain data, as has hitherto been possible for deterministic data. We suspect that Risk Solver will become entirely integrated in future versions of Excel, just as Solver was before it, which will expand the modeling possibilities within Excel.

Forecasting

Forecasting the future values of associative or time series data is frequently required in business and other applications in order to predict the future behavior of a system. For example, a tile manufacturing company used a spreadsheet for forecasting the annual sales volume for different product families [Miller and Liberatore (38)]. Special-purpose forecasting packages (such as Forecast Pro) will import spreadsheet data, automatically select the best forecasting method for that data, and compute the results. Any changes in the original data in the spreadsheet will require rerunning of the forecasting package. This kind of black box approach is appealing to certain types of users with no knowledge of forecasting methodology, who simply want an answer to a specific problem. However, for users who have some knowledge of forecasting methods, the spreadsheet medium provides an opportunity to gain additional insights and understanding of their data by providing an environment where the model sensitivity can be visually inspected by using the spreadsheet's graphing features. All types of times series analyses can be easily set up: e.g., all variants of moving averages and exponential smoothing, with identification of trend and seasonality with just an elementary familiarity of the method. The TREND function is useful if a linear or quadratic forecasting method is desired. The sophistication of the sensitivity analyses and consequently the quality of the resulting forecasts made is proportional to the user's appreciation of forecasting theory and methods [see any forecasting text, e.g., Makridakis et al., (39) for discussion of forecasting methods and applications].

A nice compromise between the black-box and build-it-from-the ground-up-in-Excel approaches is provided by CB Predictor, a module of Crystal Ball Pro. This module has the same kind of functionality as an external black-box forecasting package directly in the spreadsheet with a suite of time series methods for stationary, trend, and seasonal data (see Fig. 17 for the user screen displaying methods available). The user can let CB Predictor choose the best method for a particular data set–all that is required is for the user to specify which error measure, RMSE (root mean square), MAD (mean absolute deviation), or MAPE (mean absolute percentage error), to minimize. Alternatively, a more hands-on user can choose both the method and the method parameters (such as smoothing constants) to investigate or compare the forecasts based on insights other than minimizing historical errors.

As all spreadsheet packages contain a regression module that uses the least-squares approach, associative forecasting reduces to a simple procedure of highlighting the independent (x) and dependent (y) variable ranges. For example, a vendor is attempting to forecast the sales of ice cream (y) by using that day's temperature (x) as the explanatory (and easy to measure) variable. These data are displayed as a scatter plot in Fig. 18. Using the least-squares regression approach provides the output shown in Fig. 19 and the resulting best straight line $y = 32.335 + 1.765x$ is plotted in Fig. 20. This can be used to carry out the forecast. Multiple regression forecasting models based on more than one independent variable can similarly be set up by highlighting all independent variable ranges. Depending on the sophistication of the user, the regression output (see Fig. 19) contains information that could be used to provide error estimates, test the significance of the regression coefficients, and determine the confidence intervals for the forecasts.

Excel also provides a graphical curve fitting module (called Add Trendline) that can be used to determine the

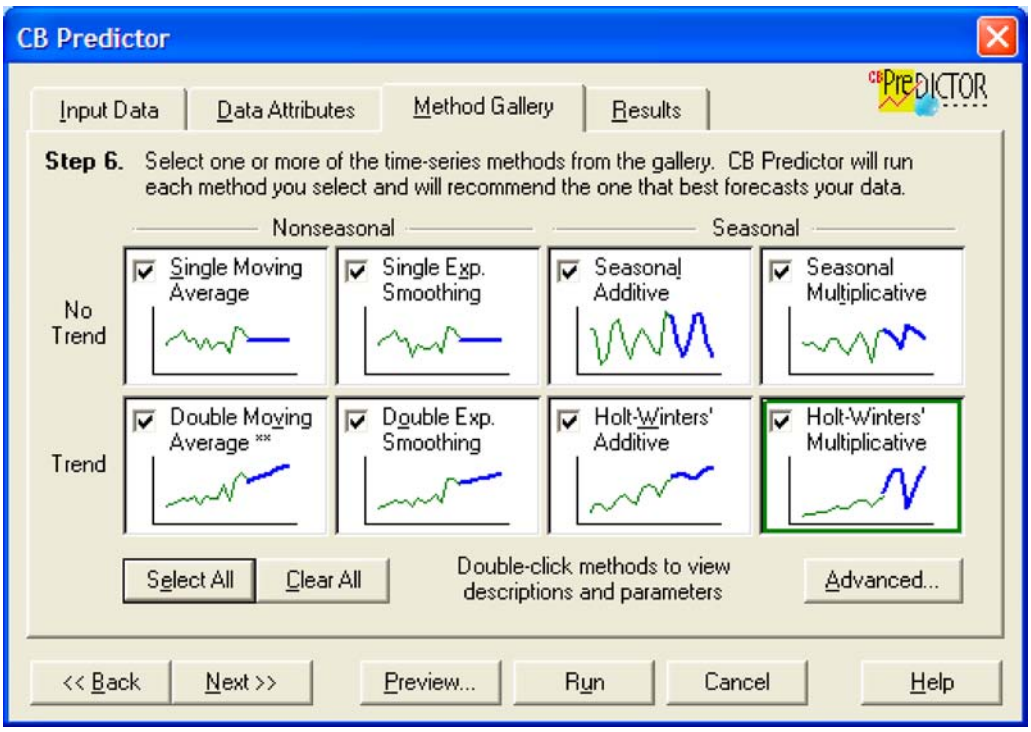

Figure 17. The forecasting methods available in CB Predictor.

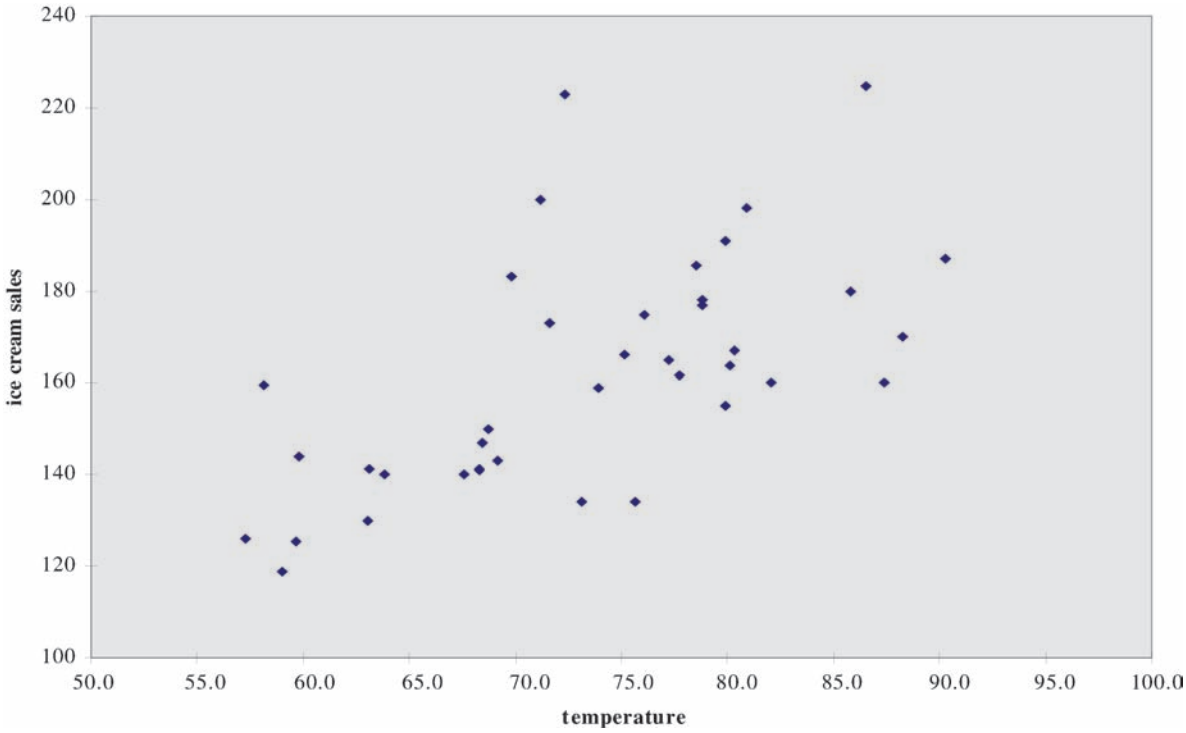

Figure 18. Scatter plot of sales of ice cream versus temperature data.

best fit equation, which can then be extrapolated to make the forecast. Curves that can be fitted include up to sixth-order polynomials, moving averages, logarithmic, and exponential.

Strategic Planning

Decision support systems should allow a user to interact with the model in order to gain better insight into the relationships, issues, and key assumptions that exist for a problem. This is exactly the type of aid that strategic planners

Regression Statistics	
Multiple R	0.61899973
R Square	0.38316066
Adjusted R Square	0.36648933
Standard Error	20.2356574
Observations	39

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	9411.2046	9411.2046	22.9832045	2.6605E-05
Residual	37	15150.8277	409.481829		
Total	38	24562.0323			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	32.3353604	27.2386805	1.18711185	0.24274941	22.855448	87.5261691	-22.855448	87.5261691
temperature	1.76456341	0.36807132	4.79408015	2.6605E-05	1.0 878008	2.51034675	1.01878008	2.51034675

Figure 19. Regression output for forecasting ice cream sales.

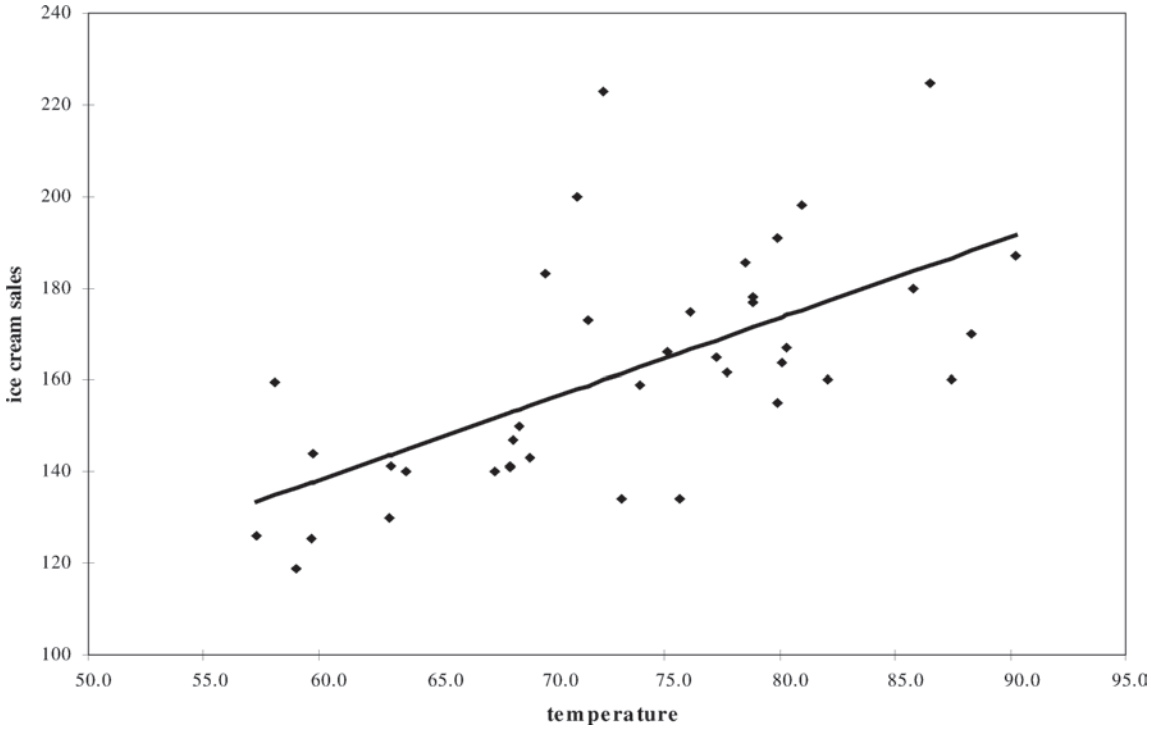

Figure 20. Scatter plot of sales of ice cream versus temperature data with best least-squares straight line superimposed.

require with their complex, uncertain applications. Most strategic planners are familiar with spreadsheets and feel at ease interacting with models created in this environment. The natural what–if facility allows them to investigate scenarios that they are considering; the Operations Research tools presented provide powerful support for answering more complex what–if and how–to questions.

Some sophisticated strategic planning models have been developed in the spreadsheet environment. Carraway et al. (40) developed a spreadsheet decision support for identifying the optimal mixture of satellite and cable technology in an international telecommunications network. This strategic model helped the telecommunications company understand the relationship between satellite and cable efficiency, network configurations, and demand forecasts. Eppen et al. (41) developed a spreadsheet-based model for General Motors to aid in making strategic decisions about capacity for four of their auto lines. The tools incorporated in the model allowed the planners to understand the tradeoffs between risk and profit in their multiproduct, multi-plant, multi-period capacity planning problem. The information obtained from the model confirmed the planner's intuition that GM had more capacity than it should for a specific product line. It also provided insight into product mix issues that the planner would not have otherwise identified for additional study.

FUTURE DEVELOPMENTS

It would be presumptuous and reckless to make definitive or specific statements regarding the future of spreadsheets,

because this is an area where advances in technology occur on an accelerating basis. However, it is safe to say that spreadsheets are an established and popular modeling medium that we expect to continue to mature, given the constant improvements in hardware relating to computation speed and available memory. Two recent advances may point to possible future trends: First, Web-based spreadsheets make spreadsheet sharing easy and accessible for multiple users, and second, Risk Solver, which performs stochastic simulation directly in a spreadsheet, opens up new modeling opportunities. In any case, more complex and advanced decision support will become a reality for the end user. We therefore strongly suggest caution and careful evaluation of their appropriate use as well as implementation of spreadsheet management control processes within an organization.

BIBLIOGRAPHY

1. Power, D. J. A brief history of spreadsheets, DSS-Resources.COM, World Wide Web, version 3.6. http://dssresources.com/history/sshistory.html (accessed August 30, 2004).
2. Panko, R. Facing the Problem of Spreadsheet Errors. *Decision Line* 2006, **37**, pp 8–10.
3. Cragg, P. B.; King, M. Spreadsheet Modelling Abuse: An Opportunity for OR? *J. Oper. Res. Soc.* 1993, **44**, pp 743–752.
4. Galletta, D. F.; Hartzel, K. S.; Johnson, S. E.; Joseph, J. l.; Rustagi, S. Spreadsheet Presentation and Error Detection: An Experimental Study. *J. Manage. Inform. Syst.* 1996/ 1997, **13**, pp 45–64.
5. H. M. Customs and Excise Computer Audit Service, Methodology for the Audit of Spreadsheet Models, 2001. http://customs.hmrc.gov.uk/channelsPortalWebApp/downloadFile?contentID=HMCE_PROD_009443.
6. Martin, A. F. Get Spreadsheets under Control. *Internal Auditor* 2005, **62**, pp 31–34.
7. Kruck, S. E. Testing Spreadsheet Accuracy Theory. *Inform. Softw. Technol.* 2006, **48**, pp 204–213.
8. Freeman, D. How to Make Spreadsheets Error-Proof. *J. Accountancy* 1996, **181**, pp 75–77.
9. Butler, R. J. Is this Spreadsheet a Tax Evader? How H.M. Customs and Excise Test Spreadsheet Applications; *Proc. of the Thirty-Third Hawaii International Conference on System Sciences*; 2000.
10. Pirlot, M. A Case Study in Transportation Network Optimization Using a Microcomputer. *Eur. J. Oper. Res.* 1990, **45**, 251–259.
11. Roy, A.; Lasdon, L.; Plane, D. End-User Optimization with Spreadsheet Models. *Eur. J. Oper. Res.* 1989, **39**, 131–137.
12. Vazsonyi, A. Where We Ought to Be Going: The Potential of Spreadsheets. *Interfaces* 1993, **23**, 26–39.
13. Carraway, L. D.; Clyman, R. D. Integrating Spreadsheets into a Case-Based MBA Quantitative Methods Course: Real Managers Make Real Decisions. *Inform. Trans. Educ.* 1997, **1**. http://ite.informs.org/Vol1No1/Carraway/index.php.
14. Powell, S. G. The Teachers' Forum: From Intelligent Consumer to Active Modeler, Two MBA Success Stories. *Interfaces* 1997, **25**, pp 88–98.
15. Bodily, S. Spreadsheet Modeling as a Stepping Stone. *Interfaces* 1986, **16**, pp 34–52.
16. Balakrishnan, N.; Render, B.; Stair, R. M. *Managerial Decision Modeling with Spreadsheets*, 2nd ed.; Prentice Hall: Englewood Cliffs, NJ, 2007.
17. Moore, J.; Weatherford, L. *Decision Modeling with Microsoft Excel*, 6th ed.; Prentice Hall: Englewood Cliffs, NJ, 2001.
18. Ragsdale, C. *Spreadsheet Modeling and Decision Analysis: A Practical Introduction to Management Science*, 5th ed.; South-Western, Mason, OH, 2006.
19. Gass, S.; Hirshfeld, D.; Wasil, E. Model World: The Spreadsheeting of OR/MS. *Interfaces* 2000, **30**, pp 72–81.
20. Seal, K. C.; Przasnyski, Z. H. Using Technology to Support Pedagogy in an OR/MS Course. *Interfaces* 2003, **33**, pp 27–40.
21. Troxell, D. S.; Aieta, J. Teach Yourself Solver. *Proc. of the NEDSI*; Atlantic City, NJ, 2000.
22. Berry, T. The Trouble with Spreadsheets. *Personal Comput.* 1989, **13**, pp 61–63.
23. Leon, L.; Przasnyski, Z. H.; Seal, K. C. Spreadsheets and MS/OR Models: An End-User Perspective. *Interfaces* 1996, **26**, pp 92–104.
24. Seal, K. C.; Przasnyski, Z. H.; Leon, L. A Literature Survey of Spreadsheet Based MS/OR Applications: 1985-1999. *OR Insight* 2000, **13**, pp 21–31.
25. Freeman, J. Spreadsheet Gaming and Management Skills Development. *OR Insight* 1993, **6**, pp 9–13.
26. Troxell, D. S. Optimization Software Pitfalls: Raising Awareness in the Classroom. *Inform. Trans. Educ.*, 2002, **2**. http://ite.informs.org/Vol2No2/Troxell/index.php.
27. Carraway, L. D.; Clyman, R. D. Managerial Relevance: The Key to Survival for OR/MS. *Interfaces* 1997, **27**, pp 115–130.
28. Taha, H. A. *Operations Research: An Introduction*, 8th ed.; Prentice Hall, Upper Saddle River, NJ, 2007.
29. Anderson, D. R.; Sweeney, D. J.; Williams, T. A. *An Introduction to Management Science-Quantitative Approaches to Decision Making*, 11th ed. Thomson South-Western, Mason, OH, 2005.
30. Turner, S.; Aieta, J.; Saber, J. Determining the Set of Alternative Optimal Solutions using Excel Solver. *Math. Comput. Educ.* 2000, **34**, pp 129–147.
31. Ravindran, A.; Phillips, D. T.; Solberg, J. J. *Operations Research: Principles and Practice*, 2nd ed.; John Wiley and Sons: New York, 1987.
32. Baker, K. R.; Camm, J. D. On the Use of Integer Programming versus Evolutionary Solver in Spreadsheet Optimization. *Inform. Trans. Educ.* 2005, **5**. http://ite.pubs.informs.org/Vol5No3/BakerCamm/.
33. Bookbinder, J. H.; McAuley, P. T.; Schulte, J. Inventory and Transportation Planning in the Distribution of Fine Papers. *J. Oper. Res. Soc.* 1989, **40**, pp 155–166.
34. Pope, J. A.; Cross, E. M. The optimal load size for ocean shippers. *Logist. Transport. Rev.* 1988, **24**, pp 299–315.
35. Bobby, G. Modelling the Cost Structure of UK Oil Distributors. *Int. J. Phys. Distribution Mater. Manage.* 1989, **19**, pp 24–29.
36. Schuster, E. W.; Finch, B. J. A Deterministic Spreadsheet Simulation Model for Production Scheduling in a Lumpy Demand Environment. *Prod. Invent. Manage.* 1990, **31**, pp 39–43.
37. Poshyanonda, T.; Dagli, C. H.; Omurtag, Y. Smart Shop Floor Scheduling Using Knowledge Based Simulation. *Comput. Industr. Eng.* 1989, **17**, pp 107–112.
38. Miller, T. C.; Liberatore, M. J. Production and Distribution Planning in a Process Firm. *Prod. Invent. Manage.* 1989, **30**, pp 44–48.

39. Makridakis, S.; Wheelwright, S. C.; Hyndman, R. J. *Forecasting Methods and Applications*, 3rd ed.; John Wiley and Sons: New York, 1998.
40. Carraway, R. L.; Cummins, J. M.; Freeland, J. R. Solving Spreadsheet-Based Integer Programming Models: An Example from International Telecommunications. *Decision Sci.* 1990, **21**, pp 808–824.
41. Eppen, G. D.; Martin, R. K.; Schrage, L. A Scenario Approach to Capacity Planning. *Oper. Res.* 1989; **37**, pp 517–527.

LINDA A. LEON
ZBIGNIEW H. PRZASNYSKI
Loyola Marymount University,
Los Angeles, CA