**Algorithmic Operations Research**

# Integrated Software Tools for the OR/MS Classroom

Ignacio Castillo, Tom Lee et János D. Pintér

Volume 3, numéro 1, winter 2008

URI : https://id.erudit.org/iderudit/aor3_1crn01

Aller au sommaire du numéro

Éditeur(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (numérique)

Découvrir la revue

Citer cet article

Castillo, I., Lee, T. & Pintér, J. D. (2008). Integrated Software Tools for the OR/MS Classroom. *Algorithmic Operations Research*, *3*(1), 82–91.

Résumé de l'article

In recent years, a growing range of software technologies has been deployed in the classroom, as a component of business, engineering, and science curricula. These technologies enable a more direct, interactive student involvement in educational or research project development. Within this general framework, we see a strong case for using integrated scientific-technical computing systems. To illustrate this point, we review the key features of Maple, with an emphasis on its optimization model development and solver features. Maple is then used to formulate, solve, and visualize an optimization model example suitable for the classroom.

*Algorithmic Operations Research Vol.3 (2008) 82–91*

# Class Room Notes

**Expository article supporting OR/MS education.**

# *Integrated Software Tools for the OR/MS Classroom*

Ignacio Castillo, [a] Tom Lee [b] and János D. Pintér [c]

[a]School of Business & Economics, Wilfrid Laurier University, Waterloo, ON, Canada N2L 3C5
[b]Maplesoft Inc., Waterloo, ON, Canada, N2V 1K8
[c]PCS Inc., Halifax, NS, Canada B3M 1J2

**Abstract**

*In recent years, a growing range of software technologies has been deployed in the classroom, as a component of business, engineering, and science curricula. These technologies enable a more direct, interactive student involvement in educational or research project development. Within this general framework, we see a strong case for using integrated scientific-technical computing systems. To illustrate this point, we review the key features of Maple, with an emphasis on its optimization model development and solver features. Maple is then used to formulate, solve, and visualize an optimization model example suitable for the classroom.*

*Key words:* Integrated software systems in the classroom; decision models development and optimization; Maple; Global Optimization Toolbox for Maple; illustrative example.

## 1. Introduction

Since the sixties, software has been an integral part of scientific and technical curricula with the introduction of programming languages such as ALGOL, BASIC, C, FORTRAN, and PASCAL. Although rigorous studies of the educational impact of information technology (IT) are rare, one can readily infer its impact from the continuously increasing usage of IT in various academic programs worldwide.

This article makes a case for the introduction of high-performance, integrated scientific-technical computing (ISTC) software into the Operations Research / Management Science (OR/MS) classroom. In many respects, this represents an unorthodox approach. General-purpose computing systems are historically not associated with business and management students in general, and OR/MS students in particular. Rather, they are much more prevalent within the natural sciences and engineering education. Arguably, the so-called "hard" sciences have successfully established workable, effective, and consistent approaches to educational computing. We believe that many of the techniques developed for the sciences and engineering have real merit also for business and management education, and the consequent interdisciplinary pollination will enrich the OR/MS classroom.

Within this general context, in this article, we introduce the ISTC software package Maple. We review some of Maple's built-in optimization features, as well as the add-on Global Optimization Toolbox (GOT). Maple and the GOT will be then used to formulate, solve, and visualize an illustrative optimization model example. In Section 2 we briefly review the history of software pedagogy. This is followed by a discussion of ISTC applications in modeling and optimization (Section 3) and a concise topical review of Maple (Section 4). The example presented in Section 5 illustrates the ISTC advantage in terms of interactive education and student involvement. Our conclusions are presented in Section 6.

## 2. Software Pedagogy

In discussing a sound pedagogical framework for the OR/MS classroom, it is useful to review the related experience of other disciplines. In particular, the fields of

*Email:* Ignacio Castillo, [icastillo@wlu.ca], Tom Lee [tlee@maplesoft.com], János D. Pintér [jdpinter@hfx.eastlink.ca].

mathematical and engineering education offer insight to the OR/MS instructor. We give a brief overview of the mathematical curriculum reform, where the basic approach is to utilize the interactivity, and the computational and visual power of the current generation of software tools. Here we think primarily of the ISTC systems Maple (Maplesoft, 2006), Mathematica (Wolfram Research, 2006) and MATLAB (The MathWorks, 2006) that can greatly enhance the students' hands-on experience with mathematical concepts and objects, as well as with symbolic and numerical computing. In the case of engineering education, one can also observe a trend of introducing "industrial-strength" software tools such as computer-aided design (CAD) and simulation systems that not only support interactivity, but also reflect state-of-the-art industrial practice. To illustrate this point, LabVIEW (National Instruments, 2006), MathCad (MathSoft, 2006), and Simulink (The MathWorks, 2006) are mentioned here.

The current generation of ISTC software tools has evolved sufficiently, to provide intuitiveness, ease-of-use, and real-world relevance that appeal to instructors and students. The knowledge base and the interactive features of ISTC software support a substantial and attractive introduction to many aspects of realistic (educational or research) project development.

### 2.1. *Mathematics Education*

One of the most successful curricular transformations has been seen in the core calculus sequence (precalculus, calculus, and differential equations) in mathematics education. Through a wide range of initiatives over the eighties and nineties, the curriculum has been transformed from one that had essentially remained unchanged for at least 100 years to one where technology can be fully integrated in the curriculum. Indeed, the concept of the mathematics "laboratory" was born in this timeframe. Murphy (2006) offers a good outline of these developments, with a detailed list of further references. We only recall here that in the mid-eighties the U.S. National Science Foundation and other organizations have established funding to accelerate the curricular transformation, and many well-known institutions began to experiment with new technologies. In the end, the application of general purpose ISTC software systems – most notably, Maple, Mathematica, and MATLAB – have become a mainstay in modern mathematics education.

As Murphy (2006) points out, the common benefits identified are:

(1) Ability to explore more complex problems as the mechanical and often tedious programming steps are greatly reduced.
(2) Graphing and visualization can significantly increase comprehension.
(3) ISTC systems facilitate the preparation of assignments and presentations.
(4) Students become more motivated to pursue independent exercises and explorations.
(5) The ISTC systems and related pedagogy encourage collaboration.

The advantages summarized above also resonate well in the OR/MS context with the following excerpt from an interview with Harlan Crowder (INFORMS, 2006b):

"*The challenge in modeling is to make sure you are solving the right problem, at the right level of detail... The good news is that model building is a skill that can be learned. And the best way to learn is to start building models. Nobody ever learned to ride a bicycle by reading about riding a bicycle. Model building is a skill that needs to be learned and practiced.*"

### 2.2. *Engineering Education*

Engineering as a profession has been fundamentally transformed in the past three decades with more design elements automated by the new generation of software tools. These enable CAD, modeling, simulation, optimization, numeric control of machines, and so on. Accordingly, there is strong motivation for academic institutions to reflect these industrial changes and to integrate the same software techniques into the curriculum. Many key subjects of engineering education are now fully integrated with using software: consult, for example Lopez (2005). Another dominant characteristic of the engineering education is the use of industry-standard software tools. Maliniak (2002) is a good example of an "industrial call" for computer-assisted design education. Unlike in the case of mathematics education, the terms here have been established by the entire profession. Consequently, modern engineering education now includes units and courses which rely on the real software tools of industry. To illustrate this point, we refer here to Ibrahim (2002) for the case of power (electrical) engineering and to Dankwort *et al.* (2004) for the case of mechanical design and manufacture. Numerous other examples – including downloadable, fully interactive course materials and realistic applications – can be found at the websites of the ISTC system developers.

### 2.3. *OR/MS Education*

The use of ISTC systems in the OR/MS classroom is not ubiquitous – at least as of today. Just two decades ago, the effective use of desktop / laptop / network enabled personal computing in the classroom has been typically considered adventurous. Among the first advocates of this new approach, Roy *et al.* (1989) proposed the use of spreadsheet models for teaching linear programming techniques. The direct access to optimization (Frontline Systems, 2006) and simulation (Palisade Corporation, 2006) technology within spreadsheets has had a major impact on the teaching of OR/MS. Spreadsheet based modeling techniques are now widely used in the classroom: consult e.g. the textbooks by Bertsimas and Freund (2000), Hillier *et al.* (2000), Winston and Albright (2001), and articles by Erkut (1998) and Ragsdale (2001). The *Teaching of Management Science Workshop* organized annually by INFORMS (2006a) also places a significant emphasis on using spreadsheets in education. One of the key reasons for this is that spreadsheets support OR/MS education in an environment that many students are expected to use later on at the workplace.

However, spreadsheet based modeling – just as any other software tool – has its inherent limitations: thus, it is not a "universal" answer to handle all OR/MS problems, even in the educational context. A clear focus on optimization model development and solution is supported by algebraic modeling languages and environments since the late eighties. Prominent examples of such modeling systems include AIMMS (Paragon Decision Technology, 2006), AMPL (Fourer, Gay, and Kernighan, 1993), GAMS (Brooke, Kendrick, and Meeraus, 1988), the LINDO Solver Suite (LINDO Systems, 2006), MPL (Maximal Software, 2006), OPL (ILOG, 2006), and TOMLAB (2006). These systems provide advanced technology for model development, preprocessing, and consistency checking, before handing over the model to a suitable solver option. Upon return from the solver, the modeling system presents a result report in a standardized format that is ready for further import/export use. The modeling environments also support links to databases and to selected external functionality. Indeed, algebraic modeling languages and the connected solver options have been used with success in OR/MS educational, research and business environments. We believe that each of the listed tools offers some key advantages depending on the end-user's objectives, and that these aspects should be emphasized early on also in education.

## 3. ISTC Applications in Modeling and Optimization

The evolution of mathematics and engineering education offer two distinct pedagogical paths: encourage and enrich the interactive exploration of (mathematical) concepts, and let the larger (engineering) profession establish the framework. Recently, Lee (2004) suggested a hybrid approach that embodies the respective benefits of both objectives. The modern generation of general-purpose ISTC software systems supports the latter approach.

### 3.1. *The Key Advantages of Using ISTC Systems*

ISTC software systems offer a powerful combination of symbolic manipulations, numerical computing, code development (programming), visualization, and documentation capabilities. They include an extensive range of functions for computing in both numeric (floating point) and symbolic (analytical solution) forms. The latter offers the distinct advantage of mapping well to manual algebraic manipulations that one would do on a piece of paper, but without the errors or tedium involved. The addition of graphics and animation tools enables the visualization of complex phenomena, thereby enhancing the students' experience. Used appropriately, ISTC systems offer the instructor and students significant advantages over many other (more traditional) pedagogies and tools, including the following aspects and usage options.

(1) Rapid prototyping and development of course materials by instructors.
(2) In-class demonstrations: the instructor directly leads students through interactive exploratory exercises.
(3) Self-study and further explorations: for example, students can analyze the impact of certain parameter changes on systems behavior and on the final results using a given (or perhaps hidden) mathematical model. This leads to interesting and realistic "what if?" modeling exercises and educational games. For an insightful discussion of such decision-making exercises, consult Dörner (1996).
(4) Development of insight symbolically, numerically and graphically: this supports parallels between the way the mathematics is manipulated within the software and the way students, instructors, and researchers develop their understanding of modeling

and solution concepts.

(5) Use as a general productivity tool: day to day hands-on usage of mathematics, calculations, programming, visualization, document preparation, from prototyping to details.

(6) ISTC systems are now fully functional, to support the development of sophisticated assignments, technical documents, books, and presentations. As of today, hundreds of books have been written directly using Maple, Mathematica, and MATLAB (as well as some other systems).

The effective use of ISTC software enriches the educational experience by allowing instructors and students to work with more complex – and thereby often more realistic – models and problems without tedious programming requirements. Furthermore, the direct and transparent interaction with the core mathematics ensures no loss of rigor and the maintenance of the theoretical framework of the course (if desired). These are significant benefits also in the OR/MS context that other disciplines have enjoyed for years.

### 3.2. *ISTC Software in Modeling and Optimization*

Model development and optimization are the key quantitative components of OR/MS education: therefore, these subjects are the ideal portion of the curriculum to benefit from the introduction of ISTC systems. The underlying concepts and techniques are inherently mathematical and many other software tools simply do not capture and manage the relevant mathematics in an educationally "optimal" manner. For example, a fundamental benefit that ISTC systems offer over spreadsheets (similarly to modeling languages) is the clear distinction between the model and the data (of a model-instance). Immediate model scalability, seamless model visualization, animation, and model documentation options are further distinguishing features of ISTC systems, in comparison to other modeling and optimization environments.

At their core, ISTC systems are a collection of abstract symbols, together with a set of rules for assembling the symbols. The formal models that are created using these systems are purely syntactical models governed by theorems. The interpretation (the semantics) of the syntax of these models is accomplished by constructing a mapping that is used to interpret the abstract symbols and rules of a model into particular truths that govern a physical system. The mapping provides a way to interpret the theorems of a formal model as true statements of the associated physical system. The core of the formalization procedure described above is the modeling relationship (Rosen, 1991). This modeling relationship is shown schematically in Figure 1, where boxes represent a physical system $S$ governed by causality (arrow [1]) and a model $M$ governed by inference (arrow [3]), where causality and inference have the intuitive meanings. The formalization procedure requires the establishment of a mapping for encoding the observables of $S$ into the theorems of $M$ (arrow [2]) and another mapping for decoding the propositions of $M$ back to observables in $S$ (arrow [4]).

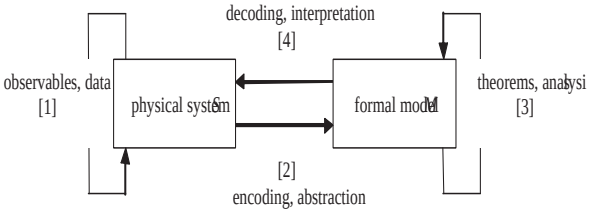


Figure 1 - The modeling relationship

Once the formalization procedure is completed, one can encode data from the physical system to create an instance of the decision problem. Data will enable a formal analysis. The interpretation of the analysis and model-based results is what allows us to make decisions pertinent to the physical system. Thus, the formalization procedure enables a course of action that supplements the use of pure intuition in making decisions. It is important to emphasize that data is not needed at the early stages of the formalization procedure. Indeed, we believe that the separation of model and data is critical at the early stages, when mathematical depth and documentation capabilities are more relevant, than instantiation details represented by the data.

Even though human judgment permeates all aspects of the formalization procedure, the use of integrated computing systems allows us to implement this procedure in a scientific fashion. Furthermore, the potential of "live" implementations is readily available, since ISTC systems already have extensive modeling capabilities, without an immediate need for additional functions and procedures. The topical functionality of ISTC software packages specifically includes also functions to solve (linear, combinatorial, local and global nonlinear) optimization models.

## 4. Using Maple in Model Development and Optimization

In the subsequent discussion, we will use Maple (Maplesoft, 2006) as a representative example of an ISTC system. Maple was conceived in the early 1980's as a system to integrate mathematical modeling and problem-solving within an interactive document (the Maple worksheet) environment. Within the worksheet, users can express mathematics, solve problems symbolically or numerically, visualize the model and the results, and insert further content such as explanatory notes and other documentation. The Maple worksheet offers a single unified environment that presents the model, data, explanation, computational and visualization options, in an intuitive and fully interactive form. The richness of the supporting mathematical library allows users to present multiple "views" of a complex concept. The Maple environment also offers word processing features to create context-rich technical documents. The key benefit for instructors and students is the ability to directly incorporate interactive mathematics and detailed comments and notes. Consequently, instructors can develop "live" course materials; students can develop, explore, and create assignments, reports and presentations within the Maple environment.

Within this article, our emphasis is focused on the modeling and optimization features of Maple. One of the key advantages of ISTC systems is the opportunity to readily accommodate the complexity of real-world systems, without paying the penalty of (arguably more difficult and tedious) traditional programming. The modeling of nonlinear systems is a prime example to illustrate this point. Nonlinear descriptive models are relevant in many areas of business, sciences, and engineering. Managing such systems naturally leads to nonlinear optimization – a subject that has been of great practical interest. For technical discussions and numerous examples, see for example the topical chapters in Pintér (1996), Chong and Zak (2001), Edgar *et al.* (2001), Pardalos and Resende (2002), Hillier and Lieberman (2005), in addition to the OR/MS textbooks cited earlier. Let us also cite here the educationally inspired comment of Grossman (2001) who makes a strong case for teaching first the basics of *general* (nonlinear) optimization before discussing the (important, but very) *special* case of linear programming. He argues that from the end user's perspective, nonlinear optimization is conceptually simpler and easier to understand, and hence should be taught before linear optimization:

*"Nonlinear optimization has four benefits to the business school management science course. First, nonlinear optimization is useful by itself, because students can use it without the added complexity of linearity. Second, nonlinear optimization is a great vehicle for delivering fundamental optimization concepts. Third, nonlinear optimization is a useful stepping-stone to the more powerful (and more difficult) techniques of linear optimization. Fourth, nonlinear optimization provides strong integration opportunities with other business school courses."*

We see a particularly strong case for using ISTC software environments to develop and solve models of nonlinear systems. Such systems are often defined by individually formulated functional relations that could be difficult to express in a compact manner (as it can be done, for example, in linear programming). Some of the model functions may also require the execution of specific computational procedures defined by special functions, integrals, systems of differential equations, external function calls, deterministic or stochastic simulation, and so on. Examples of nonlinear models that are defined by such computational procedures are abundant in various business, scientific, and engineering applications. To handle a very general class of nonlinear optimization models, Maplesoft (2004) introduced the Global Optimization Toolbox (GOT). Formally, the GOT is aimed at the numerical solution of instances of the model

(1) *min* $f(x) \quad x \in D$:=$\{x: x_l \le x \le x_u \quad g(x) \le 0\}$.

In (1) we use the notation:

- $x$decision vector, an element of the real Euclidean $n$-space $\boldsymbol{R}^n$
- $f(x)$ objective function, $f : \boldsymbol{R}^n \rightarrow \boldsymbol{R}$
- $D$non-empty set of feasible decisions; a proper subset of $\boldsymbol{R}^n$.

Specifically, $D$ is defined by the following information:

- $x_l, x_u$ explicit, finite bounds of $x$
- $g(x)m$-vector of constraint functions, $g : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$.

The concise model statement (1) subsumes a broad class of formally more general models. For example, instead of $\le$ relations, arbitrary combinations of inequality and equality relations could be used in the functions $g$; one could state explicit bounds regarding also the constraint function values; and one could even use a combination of continuous and (finitely bounded) discrete decision variables.

The core of the GOT is the customized implementation of the LGO global-local nonlinear optimization solver suite. LGO abbreviates Lipschitz(-continuous)

Global Optimizer, originally named after one of its key solver components. The theoretical foundations of LGO are presented in Pintér (1996). The GOT implementation of LGO combines Maple's aforementioned capabilities with the robustness and efficiency of the LGO solver suite. In the next section, we will use both Maple's Optimization package and the GOT to illustrate our discussion with a classroom level model example. Numerous further examples are presented in Pintér, Linder, and Chin (2006), and in the interactive electronic book (Pintér, 2006) that has been written entirely in the Maple environment.

## 5. An Illustrative Example: Portfolio Management

Before the example is presented in detail, we note that the GOT, after installation, appears within Maple as the GlobalOptimization package. In order to use the subsequently issued Maple commands directly, first we invoke the built-in (local) Optimization package, as well as the GOT by the Maple commands shown below. We will not discuss Maple programming details here, but the example presented will be easy to follow. All Maple commands will be typeset using **Courier Bold** fonts; Maple's replies will be shown simply using Times New Roman fonts, immediately following the commands. Let us also note here for clarity that the **:** symbol used in Maple input suppresses Maple output.

> **with(Optimization):**

> **with(GlobalOptimization):**

Our example illustrates a "live" demonstration and subsequent extension of the well-known Markowitz model for portfolio optimization. This model is traditionally introduced in financial engineering courses at both undergraduate and graduates levels. The model is also used as a motivating example in OR/MS courses that deal with linear and quadratic programming at the undergraduate and graduate levels. The reader is referred to Winston and Albright (2001), or to other textbooks, for additional details regarding the model. As students are often unfamiliar with the matrix-based model formulation, it may be helpful to review the finance and mathematics before the students are walked through this example. This can be easily achieved by having students work through a handout, before they participate in the "live" demonstration. This will allow them to focus on the use of the software in model development and solution, rather than the basic technicalities of the finance or mathematics background.

During a "live" demonstration, it is essential to emphasize to the audience that virtually every major portfolio manager today consults an optimization model, perhaps a more advanced variant of the Markowitz model. Managers, of course, may not follow its recommendations exactly, but they use it to evaluate basic risk and growth trade-offs. The example starts with the standard quadratic programming model formulation that balances the two competing goals of long-term growth of capital and low risk. The decision variables are the amounts invested in each asset. The objective is to minimize the variance of a portfolio's total return, subject to the constraints that i) the expected growth of the portfolio attains at least some given target level and ii) we do not invest more capital than we have. In its original form, the Markowitz model assumes no transaction costs, and thus, it is easily solved by local optimization (quadratic programming) tools. Researchers have studied numerous convex generalizations of the Markowitz model, for example, by adding convex (usually linear) transaction costs, or other linear constraints. Here we will assume – more realistically – that the investor pays concave transaction costs on asset purchases, as a function of the amounts invested. This cost structure arises in practice whenever a broker offers volume discounts on commission rates. Under these conditions, the Markowitz model becomes a non-convex optimization problem that is not easily solved using local search based optimization algorithms. We shall first formulate the original Markowitz model without transaction costs and solve it for a simple instance using quadratic programming.

Let $X = x_1, \ldots, x_n$ be the amounts that we allocate to buy the given assets $j$=1,…,$n$, $b$denotes the amount of capital that we have, $R$ is the random vector of asset returns over the time period considered, vector $r$ is the expected value of $R$, $g$ is the minimum target growth we hope to obtain, and $Q$ the covariance matrix of $R$. (All asset values, returns, and targets are expressed in dollars.) The objective function is based on minimizing the portfolio variance:

(3) $var\left(\sum_{i=1}^{n} x_i R_i\right)$

This function is equal to $X^T QX$. If, for example, $n$ = 3, then we would solve the following quadratic program:

(4) **minimize** $X^T QX$

**subject to** $x_1 + x_2 + x_3 \quad \leq \quad b$

$g \quad \leq \quad x_1 r_1 + x_2 r_2 + x_3 r_3$

$x_1, x_2, x_3$ are nonnegative

In practice, the details of the formulation will target the level of the audience. For instance, the general al-

gebraic model might not be discussed in introductory level courses. As generally done in practice, we have introduced the Markowitz model with a small instance with three assets. Suppose now that we have the following data. (For simplicity, below we use the notation $x$, $y$, $z$ instead of $x_1, x_2, x_3$.)

> **n := 3; X := <x, y, z>; b := 10000; g := 1000;**
**r := <.05, -.04, .15>;**
**Q := <<.08, -.20, .05> |**
**<-.20, .03, -.15> |**
**<.05, -.15, .45>>;**

$n$:=3

$$X := \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$b$:=10000
$g$:=1000

$$r := \begin{bmatrix} 0.05 \\ -0.04 \\ 0.15 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.08 & -0.20 & 0.05 \\ -0.20 & 0.03 & -0.15 \\ 0.05 & -0.15 & 0.45 \end{bmatrix}$$

Let us point out here that in recent years ISTC systems have dramatically improved their user interfaces. Although most of these systems began their history as rather cumbersome programming languages, they now offer intuitive, syntax-free "point and click" operations. This includes context-specific right-click menus, palette-based equation entry options, and the direct manipulation of visualization objects. In our experience, even audiences with little programming background adapt quite well to the intuitive interface and syntax of Maple. We also note that Excel Add-In packages are also available to use with Maple. Thus, it is possible for audiences with a solid spreadsheet background to be more easily introduced to the ISTC systems by using such an Add-In. Our objective function and constraints are shown below.

> **objective := expand( Transpose(X).Q.X );**
**budget := add( X[i], i=1..n ) <= b;**
**growth := add( X[i]*r[i], i=1..n ) >= g;**
**constraints := {budget, growth};**

$$objective := 0.08x^2 - 0.40xy + 0.10xz + 0.03y^2 - 0.30yz + 0.45z^2$$

*budget*:=$x + y + z$ <= 10000
*growth*:=1000 <= 0.05 $x$ - 0.04 $y$ + 0.15z
*constraints*:={$x + y + z$ <= 10000, 1000 <= 0.05 $x$ - 0.04 $y$ + 0.15 $z$}

Even though care must be taken when performing matrix operations, audiences tend to appreciate that the data, variables, and equations can be easily manipulated with their given names. As illustrated, naming conventions can be quite flexible. Moreover, the objective function could be easily expanded to explicitly uncover the quadratic structure of the objective function. Since the objective function is quadratic in the decision variables $x, y$and $z$ and the constraints are linear, we can readily use Maple's built-in quadratic programming optimizer to solve the problem.

> **originalSol :=**
**QPSolve( objective, constraints, assume = nonnegative );**

*originalSol*:=[$1.521101466610^7$,
[$x = 3606.7045794672, y = 733.3133792277, z = 5659.9820413049$]]

Notice that even though the expected return of asset $y$ is negative (-4%), its covariance with the other two assets is sufficiently negative to provide diversification benefits. Thus, its optimally allocated amount in the basic Markowitz model is positive.

Several alternatives could be investigated at this point, given the level of the course and the type of the audience. One possible alternative is to perform "what if?" analyses to gain insight regarding the model and its numerical solution as a function of the input data. Another alternative would be to investigate the application of the Markowitz model to allocation decisions across asset classes (rather than individual assets), in order to highlight the aggregation power when the number of correlations is lower and the summary statistics can be better estimated. We, however, concentrate on introducing a new aspect. Assume that we have to pay a commission of $t(x)$ dollars on the purchase of an investment for $x$dollars (the function $t$ is set by the broker), and that these costs come out of our investment budget. Then the symbolic budget constraint for the three-asset problem becomes

> **nonconvexBudget := add( X[i], i=1..n ) + add( t(X[i]), i=1..n ) <= b;**

*nonconvexBudget* := $x + y + z + t(x) + t(y) + t(z)$ <= 10000

The simplest case of the function $t(x)$ is when the commission rate is a constant percentage of the purchase amount, that is, $t(x) = cx$, with $0 < c < 1$. If, for example, the commission rate is 3% ($c$=0.03), then the budget constraint would be as follows.

> **t_const := z -> 0.03*z:**

**linearBudget := eval( nonconvexBudget,** { **t=t_const** }**);**

*linearBudget* := 1.03 $x$ + 1.03 $y$ + 1.03 $z$ <= 10000

Here, **z** is the single variable name and **0.03*z** is the result of the function acting on **z**. This notation might not be intuitive initially to everyone, but it is used here to highlight the flexibility of Maple if many different $t(x)$ functions are used in the analysis. We shall now first solve the model applying the aforementioned linear transaction cost function. As expected, the minimum variance found will increase, in comparison with the solution of the original model: since our purchasing power is reduced, we have to invest relatively more in the highest-return, highest-variance asset in order to achieve our growth target of $1,000.

> **constraints :=** {**linearBudget, growth**}**:**

> **QPSolve( objective, constraints, assume = non-negative );**

[1.5595820629 10^7,

[$x$ = 3468.2366479427, $z$ = 5664.2537697276, $y$ = 576.2474464072]]

For more advanced audiences, we can add further realism to the basic Markowitz model by supposing that the broker offers a volume discount on commissions, so that the commission is given by a concave, piecewise-linear function of the purchase amount. This cost structure is very common in practice, and its uses in other areas can be emphasized here. The straightforward Maple code implementation of the purchase cost function is omitted for brevity. However, an illustrative plot of this function is given in Figure 2.

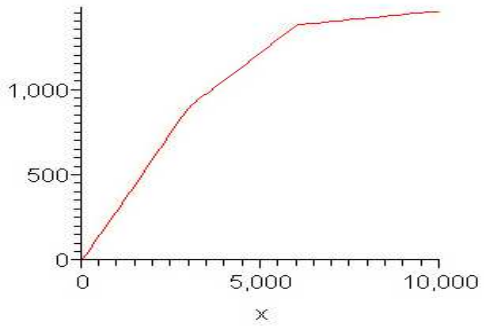


Figure 2 - Transaction cost vs. purchase amount

In Figure 3, we plot the budget constraints of the example problem under zero transaction costs (the top surface) and piecewise-linear transaction costs (the bottom surface). The implementation details of this plot are also omitted for brevity. Note that the region bounded by the red surface is convex (a tetrahedron), whereas the feasible region bounded by the blue surface is non-convex. This non-convexity makes the more realistic model a more challenging optimization problem that requires global optimization techniques.

Evidently, at this point of our "live" demonstration several concepts could be highlighted. For instance, from the perspectives of financial engineering and optimization, the discussion of convex and non-convex optimization models could lead quite well to the introduction of global optimization techniques; in particular, global optimization techniques in which ever-present derivatives of the objective function and constraints are not required. The introduction of nonlinear optimization needs could also provide strong integration opportunities with other OR/MS courses.

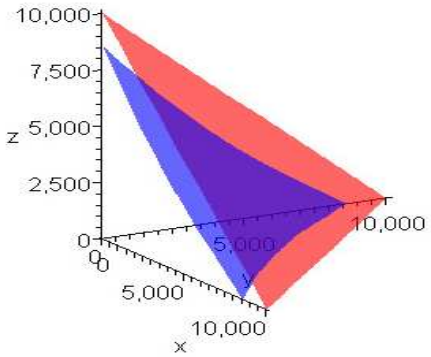


Figure 3 - Budget constraints with zero costs (red) and non-convex costs (blue)

The non-convex budget constraints are formally defined as follows in Maple.

> **pwBudget :=**

**eval( nonconvexBudget,** { **t = t_piecewise** } **):**

We first attempt to solve the non-convex problem using the local solver built into Maple, taking as initial point the optimal solution to the zero-cost problem. As it can be expected, the local solver runs into trouble. This happens because the constraints are non-differentiable at several input arguments, and most local solvers require continuous first derivatives of all model functions.

> **constraints :=** {**growth, pwBudget**}**:**

> **NLPSolve( objective, constraints, assume = non-negative, initialpoint = originalSol[2] );**

Error, (in Optimization:-NLPSolve) no improved point could be found

At the same time, the GOT with its derivative-free implementation of global and local optimization techniques handles this problem easily.

> **bounds := x=0..10000, y=0..10000, z=0..10000:**

> **pwSol := GlobalSolve( objective, constraints, bounds );**

*pwSol* := [1.76993164261 0^7, [$x$ = 2021.8978102189, $z$ = 5992.7007299270, $y$ = 0.0]]

Figure 4 shows the location of the optimal budget allocation (the dot) on the boundary of the feasible region. The surfaces representing the budget constraint (the bottom surface) and the growth constraint (the top surface) are shown as well. (This could be a good point in the lecture to mention or to recall Kuhn-Tucker theory for a technically more advance audience.)

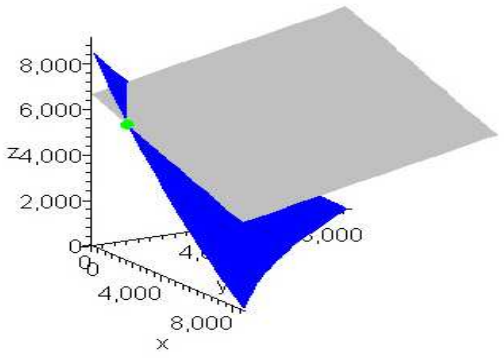


Figure 4 - Optimal budget allocation under concave transaction costs

It is clear that this "live" demonstration requires some preparation (in fact, just a little Maple programming) before it is introduced in the classroom. However, it is not far-fetched to envision the power of such a demonstration in front of the proper (student or other) audience. The key is to have in mind that we are using a tool that allows us to start with a basic (quadratic programming) model and to conclude in a few steps with a rather practical and challenging global optimization model, its solution, and supporting visualization with (a subset of) the decision variables – using only a few code lines.

We believe that this example illustrates the numerous educational benefits of an ISTC software system such as Maple. The informal student feedback at a Canadian business school supports this claim. After more than two decades of development, ISTC systems are now mature software with significant worldwide user bases. Consequently, a vast amount of high-quality, public-domain resources – including complete curriculum materials and illustrative case studies – are readily available.

## 6. Conclusions

In this article, we have presented a possible path for the OR/MS instructor who is looking at options to enrich the curriculum. ISTC systems are now permanent additions to education in engineering, science, and mathematics. We believe that the distinct benefits of such systems warrant serious consideration, and that ISTC systems will play an increasing role also in OR/MS education. Our intent is not to suggest that ISTC software would completely replace existing software packages. All computing platforms and tools discussed briefly above have respective merits and therefore should be used, when offering the best option. Indeed, we see the foreseeable future embracing both the more "conventional" and the new: ISTC systems will become a welcome complement to already existing educational and research tools, whether they are program libraries, spreadsheets, modeling languages, or custom applications. Perhaps the most pressing issue in terms of the adoption of ISTC software is the development and documentation of best OR/MS educational practices in order to pave the way for others to follow and improve. We encourage educators to contact the authors with stories and lessons learned regarding their practices.

## References

[1] Bertsimas, D. and Freund, R.M. (2000) *Data, Models, and Decisions.* South-Western College Publishing, Cincinnati, OH.

[2] Brooke, A., Kendrick, D. and Meeraus, A. (1988) *GAMS: A User's Guide.* The Scientific Press, Redwood City, CA.

[3] Chong, E.K.P. and Zak, S.H. (2001) *An Introduction to Optimization* ($2^{nd}$ Edition). Wiley, New York, NY.

[4] Dankwort, C.W., Weidlich, R., Guenther, B. and Blaurock, J.E. (2004) Engineers' CA$x$ education – it's not only CAD. *Computer-Aided Design* 36, 1439-1450.

[5] Dörner, D. (1996) *The Logic of Failure.* Perseus Books, Cambridge, MA.

[6] Edgar, T.F., Himmelblau, D.M. and Lasdon, L.S. (2001) *Optimization of Chemical Processes* ($2^{nd}$ Edition). McGraw-Hill, Boston, MA.

[7] Erkut, E. (1998) How to 'Excel' in teaching management science. *OR/MS Today* 25, 40-43.

[8] Fourer, R., Gay, D.M., and Kernighan, B.W. (1993) *AMPL - A Modeling Language for Mathematical Programming.* The Scientific Press, Redwood City, CA.

[9] Frontline Systems (2006) *Premium Solver Platform – Field-Installable Solver Engines.* Frontline Systems, Inc., Incline Village, NV.

[10] Grossmann, T. (2001) Reversing tradition: nonlinear optimization deserves more emphasis, *OR/MS Today* 28, 22-25.

[11] Hillier, F.J. Hillier, M.S., and Lieberman, G.J. (2000) *Introduction to Management Science.* McGraw-Hill, New York, NY.

[12] Hillier, F.J. and Lieberman, G.J. (2005) *Introduction to Operations Research*. ($8^{th}$ Edition). McGraw-Hill, New York, NY.

[13] Ibrahim, E.S. (2002) A comparative study of PC based software packages for power engineering education and research. *Electrical Power and Energy Systems* 24, 799-805.

[14] ILOG (2004) *ILOG OPL Studio and Solver Suite.* See also http://www.ilog.com.

[15] INFORMS (2006a) Teaching of Management Science Workshop. See recent workshops at http://meetings.informs.org/TMSWorkshop/.

[16] INFORMS (2006b) Profiles in OR/MS: Harlan Crowder (Interview). INFORMS, Hanover, MD. See also http://www.informs.org/article.php?id=413.

[17] Lee, T. (2004) The "Gray Box" approach: The potential of symbolic mathematical software in engineering analysis and education, *Proceedings of CDEN Conference 2004*. McGill University, Montreal, QC.

[18] LINDO Systems (1996) *LINDO Solver Suite.* LINDO Systems, Inc., Chicago, IL.

[19] Lopez, R.J. (2005) *Advanced Engineering Mathematics with Maple.* Electronic book edition published by Maplesoft, Inc., Waterloo, ON. See also www.maplesoft.com/products/ebooks/AEM/index.aspx.

[20] Maliniak, D. (2002) Educational software can help new engineers lacking the basics. *Electronic Design*, October, 26.

[21] Maplesoft (2004) *Global Optimization Toolbox for Maple*. Maplesoft, Inc., Waterloo, ON.

[22] Maplesoft (2006) *Maple.* Maplesoft, Inc., Waterloo, ON.

Received 10 October, 2006; revised 7 June 2007; accepted 10 June 2007

[23] MathSoft (2006) *MathCad.* MathSoft Engineering and Education, Inc., Cambridge, MA.

[24] Maximal Software (2006) *MPL Modeling System.* Maximal Software, Inc., Arlington, VA.

[25] Murphy, L.D. (2006) Computer Algebra Systems in Calculus Reform. University of Illinois, Urbana-Champain, IL.

[26] National Instruments (2006) *LabVIEW.* National Instruments Corporation, Austin, TX.

[27] Palisade Corporation (2006) *DecisionTools Suite.* Palisade Corporation, Ithaca, NY.

[28] Paragon Decision Technology (2006) *AIMMS Modeling System.* Paragon Decision Technology BV, Haarlem, The Netherlands.

[29] Pardalos, P.M., and Resende, M.G.H., Eds. (2002) *Handbook of Applied Optimization.* Oxford University Press, Oxford.

[30] Pintér, J.D. (1996) *Global Optimization in Action.* Kluwer Academic Publishers, Dordrecht.

[31] Pintér, J.D. (2006) *Global Optimization with Maple – An Introduction with Illustrative Examples.* An interactive electronic book published and distributed by PCS, Inc., Halifax, NS, and Maplesoft, Inc., Waterloo, ON.

[32] Pintér, J.D., Linder, D. and Chin, P. (2006) Global Optimization Toolbox for Maple: An introduction with illustrative applications. *Optimization Methods and Software* 21, 565-582.

[33] Ragsdale, C.T. (2001) Teaching Management Science in Spreadsheets, from Decision Models to Decision Support. *INFORMS Transactions on Education* 1.

[34] Rosen, R. (1991) *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life*. Columbia University Press, New York, NY.

[35] Roy, A., Lasdon, L. and Plane, D. (1989) End-user optimization with spreadsheet models. *European Journal of Operations* Research 39, 131-137.

[36] The MathWorks (2006) *MATLAB.* The MathWorks, Inc., Natick, MA.

[37] TOMLAB Optimization (2006) *TOMLAB Modeling System.* TOMLAB Optimization AB, Västerås, Sweden.

[38] Winston, W.L. and Albright, C.A. (2001) *Practical Management Science.* ($2^{nd}$ Edition). Duxbury press, Pacific Grove, CA.

[39] Wolfram Research (2006) *Mathematica.* See also http://www.wolfram.com.