Implementing an Introductory Mathematical Software Course

Michael A. Karls
e-mail: mkarls@bsu.edu
Department of Mathematical Sciences
Ball State University
Muncie, IN 47306
January 23, 2010

Abstract

This paper looks at the challenges of designing and implementing an introductory mathematical software course; provides an overview of topics covered and resources used for each type of software (Excel, Mathematica, and MATLAB); gives examples of homework problems, exam questions, and project topics chosen for the course; and addresses issues that have arisen.

1. Introduction

In fall 2007, I piloted a course, entitled *MATHS 159 Introduction to Mathematical Software*, in the Department of Mathematical Sciences at Ball State University. MATHS 159 is designed to introduce mathematical science majors to "mathematical software" that can be used in both their courses and future careers. The software packages used in this three-credit hour course that includes an extra one-hour weekly lab are *Excel*, *Mathematica*, and *MATLAB*, all of which have been integrated into various courses throughout our departmental curriculum. This course, which is a direct result of work initiated in spring 2007 by Professors Irene Livshits, Richard Stankewitz, and myself, as part of a Ball State Enhanced Provost Initiative Grant, [26], became a requirement for ALL of our majors and minors, starting in fall 2008.

Our department's rationale for requiring a mathematical software course for our majors is as follows: "Modern mathematics involves the use of software, so it is important that our majors understand that computer usage is an integral part of how math is done. They should also have an understanding what computers can do and what computers cannot do. Currently, most of our math majors have difficulty with the technology they encounter in many of their classes. This in turn makes it difficult to incorporate mathematical software into our upper-level courses. One reason may be that most students in our programs don't have a solid foundation in both the mechanics of the software and the use of the software as a tool for doing mathematics. … MATHS 159 … taken early in a student's program, will provide a systematic introduction to mathematical software packages that are used in many of our upper level courses (currently *Excel*, *Mathematica*, and *MATLAB*). After completing MATHS 159 students will know how to use this software to solve problems and will be ready to apply this technology to more advanced problems and concepts." [9].

2. Departmental Overview

The Department of Mathematical Sciences is one of twenty departments in the College of Sciences and Humanities, which in turn is one of seven colleges in Ball State University. Ball State University is a public state institution located in Muncie, Indiana, with an enrollment of approximately 20,000 students, [4]. In addition to Ball State's admission criteria of three years of college preparatory mathematics in high school (two years of algebra and one year geometry), students enrolling in freshman calculus are expected to have completed courses equivalent to college algebra and trigonometry. SAT/ACT scores, departmental placement test scores, or high school records that indicate sufficient mathematical background are also accepted, [2], [3]. The Department of Mathematical Sciences offers a variety of options in mathematical science for

undergraduates: Actuarial Science, Mathematical Sciences (options in pure and applied mathematics), Mathematics Teaching, or Mathematical Economics (joint with the Department of Economics). Non-mathematical science majors can also choose a minor in Mathematics. In addition, there is a graduate program at the master's level and courses are offered for Elementary Education majors who choose a concentration in mathematics, [3], [5].

In any given semester, approximately 400 students are enrolled in departmental programs, of which, about sixty percent are in one of the undergraduate majors. Of the 240 undergraduate majors enrolled in courses for fall 2008, there were 97 Actuarial Science majors, 112 Mathematics Teaching majors, and 31 Mathematical Science majors. All departmental majors take a common core of courses which consists of three semesters of calculus, linear algebra, discrete mathematics, a senior capstone course, and as of fall 2008, MATHS 159.

3. Software Use in Departmental Courses

For our calculus sequence, there is a *Mathematica* component which has varied over the years, ranging from a set of required common labs (about three lab assignments per course) to a lab component that is required, but instructor-dependent, which is the current set-up. For linear algebra, a computer software component is included (usually *Mathematica* or *MATLAB*), with the amount included left to the discretion of the instructor.

After the calculus and linear algebra sequence, mathematical software use in courses varies by course, major, and instructor. Some instructors, such as I, use software or technology in almost all of the courses taught, while others may rarely use technology outside of a course in which it is required. Mathematics Teaching majors take a junior-level course on software/technology that can be used for their teaching, including applications of graphing calculators, *Excel*, *Logo*, and *Geometer's Sketchpad*. Actuarial Science majors use *Excel* extensively in their courses and Mathematical Science majors use *MATLAB* in their numerical analysis courses. Other courses that routinely integrate mathematical software include most of our actuarial science and statistics courses (*Excel*, *Minitab*, *SAS*, *R*), as well as junior level Mathematical Models and Numerical Analysis courses (*Mathematica*, *Excel*, *MATLAB*). Mathematical software is also used in courses such as Differential Equations, Partial Differential Equations, and Operations Research (*Mathematica*, *Excel*, *MPL*, *LINGO*).

All of the software we use is available to students in a departmental lab as well as on a set of 30 laptops on a mobile cart (Computers on Wheels or COW) that we can move to a lab or classroom as needed. Using the COW, we can turn any classroom into a laptop laboratory. In addition, most of the software is available throughout campus in public labs, many of which are open 24 hours. Finally, Ball State students can download free copies of Office 2007 and a student version of *Mathematica* for use on their own computers. *MATLAB* is not free, [36], but students can download free programs such as *Octave*, which will run most MATLAB files (including examples provided in this paper), [12].

4. Initial Challenges

For the mathematical software course, at first, all I had for a course guide/template was a course master syllabus and a list of technology topics that Professors Livshits, Stankewitz, and I had determined were essential for a mathematical science major to be able to use in their future coursework and subsequent careers. The topics were chosen after reflecting on our own personal experience as instructors and research mathematicians, as well as consulting with departmental colleagues and industry contacts. From this, I had to create a set of lecture materials, homework assignments, and projects at a level appropriate to freshman who would be concurrently enrolled in a first-semester calculus course.

For *Excel*, *Mathematica*, and *MATLAB*, there are existing textbooks and other reference materials that address each software package individually, either as a guide to use the software, including [7], [10], [11], [15], [16], [17], [33], [38], [39], and [40] or as a supplement to a Calculus course or textbook, such as [8], [13], [19], [21], and [31] but my colleagues and I were not aware of any resource that specifically targets *all three* software packages. Thus, I needed to make up my notes from scratch, with help from appropriate sources as needed. In addition to showing students how to use the software at a basic level, I wanted to incorporate useful examples that would both illustrate how to use the software, as well as teach them some basic ideas from calculus, linear algebra, and applied mathematics. All of this had to be done at a level appropriate to students who would be enrolled in a first-semester calculus course, especially first-semester freshman.

For *Excel*, I started with notes from an introductory short course given at Ball State, [16], and [38]. I also relied on *Excel*'s built-in and online help files [27], [28], an online "generic spreadsheet" tutorial [20], and web pages such as the following for using the IF function from About.com [14]. A great introduction to the *Excel* Solver Add-on package for solving simple linear programming problems can be found in [18]. For *Excel* examples that dealt with finance and simple linear programming I used material from [25]. Population growth and optimization examples came from [29] and [30]. For calculus – based examples and homework, I used material from [32]. Finally, a great example that I integrated throughout the course (as we will see below) came from the paper [6].

For *Mathematica*, my main sources were the excellent built-in help files, a brief introductory book, [7], a text version of the built-in help files, [39], Wolfram's web page [41], and online tutorials from Wolfram [42]. I also used material from a tutorial which I had written for our majors when they enroll in calculus [22], based in part on material from [13]. Many of the examples for this section were calculus – based and found in [19] and [32] (prior to the sections on antidifferentiation). Some of the topics from the *Excel* section were also revisited to see how they could be dealt with in *Mathematica*.

My main sources for learning *MATLAB* basics were the built-in help files, an introductory tutorial from the University of Utah, [37], and a more detailed primer that I found online [33]. Most of the mathematics for this section was calculus based, at the level of differentiation, Riemann sums, and integration. We also touched on programming and graphing, using examples from the previous sections. For calculus commands I relied extensively on a book by Jenson, [21].

Since *MATLAB* is matrix – based, I also had to bring students up to speed on matrix concepts, at the level of introductory linear algebra. My main sources were a linear algebra textbook, [1], and [24], which has a great introduction to matrices. To help tie the other software to *MATLAB*, we looked at how to solve linear systems in *Excel* and *Mathematica*. For *Excel*, I found a web supplement that shows how to solve a system of equations with the Solver, as well as how to subtract, multiply, and invert matrices, [35]. My goal was to make it clear (after using the other two software packages) that *MATLAB* is more natural for working with matrices.

5. Introductory Survey

One of the tools that I have found helpful in my courses is a survey handed out on the first day of class. In addition to standard questions such as major, math courses taken, or types of mathematical software used, I ask a series of directed questions tailored to the specific course. For MATHS 159, I include calculation questions that are too "tough" to complete in the allotted time (15 – 20 minutes), but all are easy to complete with software such as *Excel*. The following questions also show up as examples in my first *Excel* lecture:

1. For the integers n = 1, 2, 3, … 99, 100, find each of the following:
   a. n^2
   b. n^3

c. $4n^3 + n^2 - 1$
d. $1 + 2 + \ldots + n$ (Note: For n = 1, the sum is 1. For n = 2, the sum is 1 + 2.)

2. The data in the following table are the land area in Australia colonized by the American marine toad (Bufo marinis). The first column is year and the second column is cumulative area occupied in km^2.

Year	Area(km^2)
1939	32800
1944	55800
1949	73600
1954	138000
1959	202000
1964	257000
1969	301000
1974	584000

a. Plot this data with Year on the x-axis and Area on the y-axis.
b. Find a function that can be used to predict this growth.
c. Estimate when Australia will be overrun by the toads (Australia's area is 7,619,000 km^2).

3. Sort the following list of birthdays of famous mathematicians according to their
a. Birth Year
b. Name
c. Birth Year followed by Name

Birth Year	Name	Birth Year	Name
1584	Vernier	1840	Henrici
1646	Flamsteed	1852	Le, P.
1736	Bring	1900	Aiken
1739	Klügel	1878	Bernstein, F.
1939	Baker	1909	Black
1852	Frattini	1920	Pillai
1888	Courant	1946	Margulis
1924	Cohn	1880	Bernstein, S.
1942	Hawking	1796	Bienaymé
1814	Wantzel	1801	Cournot
1819	Adams	1867	Bôcher
1883	Keynes	1883	Schouten
1816	Wolf	1910	Turán
1905	Dubreil-Jacotin	1910	Koopmans
1906	Feller	1911	Kakutani
1922	Marchenko	1939	Kingman
1818	Joachimsthal		

6. Course Overview – Topics Covered

The following is an outline of the topics covered in the version of MATHS 159 course that I taught in fall 2008, which is very close to the versions taught in both fall 2007 and fall 2009. Topics are listed in the order presented, by lecture, with software specific concepts and commands interspersed

with mathematical examples and ideas. This is done to show how the software and mathematics go hand-in-hand. Since many of the students are concurrently enrolled in a first semester calculus course, MATHS 159's calculus topics (limit, derivative, optimization, Riemann sum, and integral) are positioned in the course to "parallel" what students will be seeing in their calculus class.

The order in which the software type is introduced is also important – *Excel* appears first since it is more likely that students entering the course will have encountered a spreadsheet program than the other two types of software. *Mathematica* is the second topic for two reasons – first, the students will have encountered it in their calculus course by the time it is introduced in this course (approximately the sixth week of the semester) and second, *Mathematica*'s initial learning curve is steeper than *Excel*'s. *MATLAB* is the least "user friendly" and requires an understanding of matrices, so it was chosen as the final software package to study.

Topics in *Excel*

Lecture 1: Definitions and Terminology

- Entering labels, constants, and formulas
- Formatting cells (font, alignment, number)
- Adjusting column width and row height
- Fill down and fill right
- Built-in functions AVERAGE and SUM
- Fill Handle
- Other cell basics (Merge and Center, Fill Color, Borders, Hyperlink)
- Making a chart with the Chart Wizard (adding a trendline, F11 creates a chart automatically)
- Importing data from a text file
- Creating a formula
- Page setup for printing
- Sorting Data (Sort Smallest to Largest button, Sort menu, AutoFilter)

Lecture 2: Finance Applications with *Excel* – Simple and Compound Interest

- Simple Interest
- Compound Interest
- Present Value
- What-If Analysis (Scenario Manager and Goal Seek)

Lecture 3: Finance Applications with *Excel* – Annuities and Amortization

- Sequence (Fibonacci, geometric)
- Series (partial sum, closed form)
- Annuities as partial sums
- Annuities (ordinary, annuities due)
- Future Value of an Ordinary Annuity
- Sinking Fund
- Using *Excel*'s built-in functions to perform these calculations
- Present Value of an Annuity
- Mortgage, periodic payments
- Amortization Schedule

Lecture 4: Logical Functions and Conditional Formatting in *Excel*

- Logical Functions (TRUE, FALSE, NOT, AND, OR, IF, IFERROR)
- Nested IF
- LOOKUP

- Conditional Formatting

Lecture 5: Mathematical Functions in *Excel*
- Common Mathematical Functions (SQRT, ABS, EXP, LN, LOG10, POWER, ROUND, SIN, COS, TAN, CSC, SEC, COT, PI, RADIANS)
- Best-Fit Lines (SUMPRODUCT, SLOPE, INTERCEPT)
- *Excel*'s Trendline Feature (number of digits, exponential)
- Rates of Change (average, instantaneous, tangent line, derivative)
- Engineering Functions (BIN2DEC, HEX2DEC, CONVERTCOMPLEX, IMPRODUCT)
- *Excel* Add-Ins

Lecture 6: Recurrence Relations in *Excel*
- Recurrence Relation (factorial function, Fibonacci sequence, compound interest)
- Initial Conditions and Order
- Closed-form solutions (compound interest, Fibonacci sequence)
- Exponential Function
- Exponential Growth and Decay
- Damped or Undamped Oscillation
- Fixed Points and Stability

Lecture 7: Population Models in *Excel*
- Finding a recurrence relation from a closed-form model (exponential growth for toad population data)
- The Logistic Model
- Using *Excel* to study growth rates with different carrying capacities and a fixed growth rate
- Fixed points in the logistic model
- Two or more populations (predator-prey, host-parasite, competitive hunter, arms race)
- Predator-Prey (foxes and rabbits) in *Excel* – recurrence relations
- Revised Predator-Prey Model which includes logistic growth

Lecture 8: The *Excel* Solver
- Maximizing an area enclosed by a fence.
- Minimizing least squares error in an approximating function (SUMXMY2)
- Maximizing office storage space (simple linear programming, naming a range of cells)
- *Excel*'s Solver Add-In

Topics in *Mathematica*

Lecture 9: Basic Syntax and Basic Commands
- Basic Syntax
- Constants and Commands
- Grouping Symbols (mathematical grouping, function syntax, list syntax)
- Types of Equals Signs (=, ==, :=)
- Calculator Commands (trigonometric functions, N)
- Symbol Manipulation (Clear, Expand, Factor, Solve, Apart, Together)
- Graphing Commands (Plot, ListPlot)
- Defining Functions ([ ], Simplify, **/;)**

Lecture 10: Entering Text, Formatting, and Using the Help File
- Entering Text and Formatting (changing text, palettes, shortcuts)

- Using the Help Browser (Plot3D example)

Lecture 11: Formatting and Working with Tables
- Table Command (TableForm, GridBox, FrameBox)
- Extracting or Working with Elements in Tables ([[ ]], Length)
- Tables of Graphs

Lecture 12: Calculus Commands
- Finding a Limit (Limit)
- Differentiating a Function (D, prime notation, higher-order derivatives)
- Integrating a Function (Integrate, NIntegrate)
- Evaluating Sums and Series (Sum)

Lecture 13: Parametric Curves and Surfaces; Animation
- Parameterized Curves and Surfaces (ParametricPlot, ParametricPlot3D)
- Real-Time 3D Graphics (rotating, zooming)
- Animation (ListAnimate, Animate, Manipulate)

Lecture 14: Importing and Exporting Data
- Import and Export (Head, $Path)
- Filetypes (CSV, DAT, XLS, etc.)
- Put and Get (FilePrint, PutAppend, Needs, Flatten)

Lecture 15: Optimization
- FindMinimum and FindMaximum (local)
- Minimize and Maximize (global)
- Constraints
- NMinimize and NMaximize
- Replacement Operator (/.)

Lecture 16: Numerical Computation
- FindRoot (Accuracy, WorkingPrecision, Chop)
- NSolve (solving polynomial equation or system polynomial of equations)
- NLimit vs. Limit
- Nintegrate (Timing, MaxRecursion)

Lecture 17: Simple Programming
- Definition of Computer Program
- Logical Operators (And, Or, Not, Xor, Nor, Nand)
- If (piecewise defined function, absolute value function example using If vs. /;)
- Differentiating each version – which works better as a plot
- Differentiating each version – which "works"
- Which (piecewise defined function – more than two "pieces", differentiating yields a function defined via Which)
- Switch
- Nest (recurrence relations)
- Print (Hello World example, new line (\n))
- Do (sum of squares, including a Print statement, adding an If statement, Return[ ])
- For (local vs. global variables, Module, Continue[ ], Break[ ])
- While (Coin Toss example, RandomInteger, using And to ensure While loop stops)

- Newton's Method Example (by "hand", simple Do loop, Do loop with intermediate outputs printed via Print, For loop, maximum 50 steps, tolerance checked via If)

Topics in *MATLAB*

Lecture 18: Systems of Equations

- System of two equations in two unknowns
- Review of substitution and elimination
- Solving a System of Equations with Technology (calculator, *Mathematica*, *Excel*)
- Using *Excel*'s Solver to solve a system of equations (2 x 2 case)
- Using *Mathematica*'s Solve command to solve a system of equations (2 x 2 case)
- General 2 x 2 system – what choices of coefficients are allowed?
- Solving general 3 x 3 case (by hand, calculator, *Mathematica*, *Excel*, Other, when a solution is guaranteed, *Mathematica* solution)

Lecture 19: Elementary Matrix Theory

- Matrix definition and examples (row, column, vector,
- Matrix Notation ($[A]_{ij} = a_{ij}$, square matrix)
- Arithmetic with Matrices (addition and subtraction, matrix multiplication, scalar multiplication)
- Identities and Inverses
- Review of Real Number Properties (additive identity, multiplicative identity, additive inverse, multiplicative inverse)
- Matrix Properties (zero matrix, identity matrix, additive inverse)
- Multiplicative Inverse (2 x 2 case, setting up algebraic system to solve to find inverse)
- When is matrix A invertible? (determinant of a matrix, how to check if a matrix is invertible via determinant.)
- Linear Systems of Equations Revisited (2 x 2 linear system in matrix form, solve via multiplication by multiplicative inverses.)

Lecture 20: Introduction to *MATLAB*

- *MATLAB* Basics (Desktop, help, lookfor, doc, type, up – arrow, clc, Command History window)
- Entering Matrices
- The Colon Operator (pulling out parts of a matrix, concatenating matrices)
- Matrix Operations (+, -, *, ^, ', .', inv, \, /)
- Array Operations with matrices A and B, scalar k (k*A, A.*B, A./B, A.^B)
- Special Matrices and Matrix Functions (eye, zeros, ones, det, clear, clear all)
- Linear Systems of Equations with *MATLAB* (2 x 2)
- Making a Table of Values for a Function
- Creating an M-File
- *MATLAB*'s Path and Running an M-File
- The Plot Command (plot, title, xlabel, ylabel, grid, legend)
- Multiple plots and piecewise-defined functions

Lecture 21: Numerical Differentiation and Integration in *MATLAB*; Function M-files

- Numerical Differentiation (linspace, diff, graphical comparison via subplot)
- Numerical Integration (Riemann sums via sum)
- Function M-Files

Lecture 22: Working with Data in *MATLAB*

- Useful built-in commands for data manipulation (fzero, sum, min, max, length, mean, median, std, sort)
- Finding minimum of a function on an interval (numerically, discretely, via fzero)
- Represent a polynomial as a vector of coefficients (polyval, polyfit)
- Fit a polynomial to a set of data via least squares
- Importing data from text file into *MATLAB* via Import Wizard
- Fit data with polynomial
- Compare to *Excel*'s Trendline or *Mathematica*'s Fit command

Lecture 23: The Symbolic Toolbox

- *MATLAB*'s built-in symbolic commands (syms, diff, int, simplify, pretty, subs, double, ezplot)
- *MATLAB*'s function calculator (funtool)

Lecture 24: Programming in *MATLAB* Examples

- Plotting Toad Data (toads1.m) (toads1.pdf)
- Making a Loop
- Working with Symbolic Functions
- Newton's Method
- Competitive Hunters Model (cancan m-file), (hunter m-file) (ode45 m-file for *Octave*) (cancan.pdf)

7. Sample Homework and Exam Questions

Here are a few examples of the types of questions I asked students on their homework or exams, broken down by software package – for more examples, see [23].

Excel

1. (*Toads Revisited*) Using the data from the Toads Example (from the survey above), create a table in *Excel* with a column for years and a column for area. Use this table to make a scatter plot that includes a best-fit line, with the best-fit line labeled.
    a. Add a third, fourth, and fifth column to the toad data table. Set up the third column to compute the area predicted by the *Excel* best-fit line. Label this column "*Excel* Best-Fit Line". In the fourth column, find the difference between the actual area and the best-fit line's predicted area. Label this column "Error". In the fifth column, find the *relative error*, which is given by *relative error = (actual value – best-fit line value)/(actual value)*. Label this column "Relative Error". *For columns that involve calculations, create a formula within Excel.*
    b. How well do your best-fit line model values and resulting errors agree with what you see in the graphs?
    c. Repeat part (a) with the linear function y = 13568.1x -2.63405*10^7, which was found with *Mathematica*'s "Fit" command. Label columns accordingly.
    d. Repeat part (b) with the linear function y = 13568.1x -2.63405*10^7.
    e. Which linear model is better – *Excel*'s or *Mathematica*'s?
    f. Why do you think one model is better than the other?
    g. Australia has a total area of 7,619,000 km^2. Use the "better" best-fit line to help estimate graphically when Australia will be overrun by toads. Hint: Change features on your graph, such as horizontal or vertical axis scales via the Format Axis Menu or use the Forecast feature in the Trendline Options Menu.

h. Show how you obtained the estimate in part (g).
i. Try different trendlines in *Excel* – do any other types of trendlines seem to "work better"? Use the same sort of numerical estimates as you did for parts (a) and (c) above. Plot the trendline you think fits the toad data the "best".
j. Use the trendline found in part (i) to estimate when Australia will be overrun by toads.
k. Is this estimate reasonable – justify your answer.

Source: The toad data comes from a College Mathematics Journal article [6] in which actual toad data is used to illustrate modelling a population via exponential growth. I chose this example because it involves a nice set of data that can be graphed with *Excel*, and then used to investigate fitting curves to data, which in turn can be used as models for population growth. The added unexpected bonus is that *Excel* rounds the constants in the equation it prints for a curve fitted to data via the Trendline feature. This rounding is significant enough (in this example) to cause a best-fit line equation that is clearly wrong graphically. I have gotten a lot of mileage out of this set of data. Other questions and examples related to the toad data include: actually calculating the best-fit line slope and y-intercept several different ways (formulas, built-in *Excel* functions, minimizing the sum of the squares for error with *Excel*'s Solver); treating toad growth as a recurrence relation; and fitting the data to exponential, logistic, or polynomial models (in all three types of software).

2. (*Buying a House*) You purchase the home of your dreams for $185,000 with 20% down. The current mortgage rates are 6.375% for a 30-year fixed mortgage and 6.00% for a 15-year fixed mortgage. In addition to the 20% down payment, there are *closing costs* of $2436 for bank fees, surveying, appraisal, and taxes that must be paid up front at the time of the loan. One way to do so is to add the closing costs to the loan amount.
    a. Create an amortization schedule for each loan, assuming the closing costs are paid out of pocket.
    b. How much total interest is paid for each loan?
    c. Add the closing costs to the loan amount and repeat part (a). How much more do you pay for each loan (remember to include the closing costs in the loan costs in part (a))?
    d. If closing costs were paid out of pocket, as in part (a), how much sooner could you pay off each loan if you paid one extra payment per year by dividing the payment into twelve equal payments that are applied directly to the principal? What would be the total interest for each loan in this case?
    e. Suppose you have taken out the loan for 15 years as in part (a), and at the end of the 6^{th} year, you need a new roof on your house, which will cost $9500. At the time you need the roof, the current mortgage rates are 5.125% for a 15-year fixed loan (or any lesser amount of time). Closing costs to refinance will be $3705. Find and justify the best investment strategy from among the following, with the requirement that the house will still be paid off in 15 years total.
        i. Borrow the money for the roof by refinancing, i.e. adding the roof cost and closing costs to the principal and taking out a new mortgage on this amount. (The first mortgage is paid off in this process.)
        ii. Pay for the roof out of pocket and refinance, with closing costs added to the new loan amount.
        iii. Pay for the roof out of pocket and don't refinance!

Source: This is a "standard" amortization schedule problem that can be done in *Excel*, with a few "twists" that are based on my own experiences with owning a house. At some point I needed to put on a new roof and had to decide whether or not it would be better to borrow money to pay for the roof by refinancing or to pay out of pocket. This example also brings in the idea of closing costs, which may not always be addressed in amortization examples.

Mathematica

1. (*Tangent Lines and Secant Lines*) Choose a function y = f (x) defined on an interval [c,d] and use this function to illustrate graphically the idea of tangent line at the point (a,f(a)) as a limit involving secant lines. Your example should include graphs that can be animated to illustrate the concept. Label your graphs appropriately. Also, your a – value should be easily changeable, i.e., make it a global variable. Explain in words what is happening. Here is an example with the function $f(x) = x^3+x-4$ and a = 1 (TangentLineExample.nb) (TangentLineExample.pdf). **Choose a different function and a – value than those given in the example!**

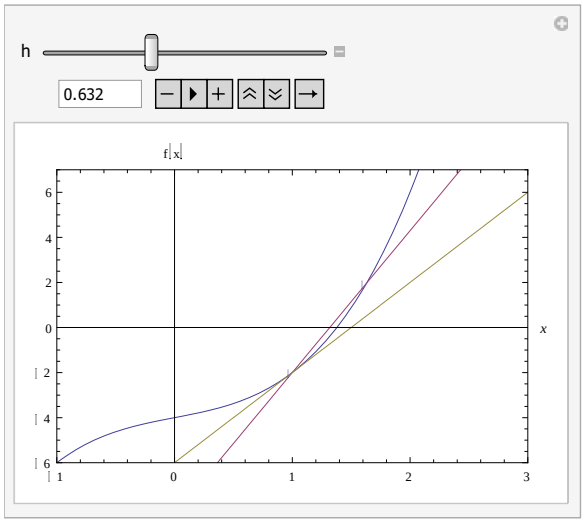

Figure 1: tangent line as limit of secant lines illustration via Mathematica's Manipulate command, using $f(x) = x^3+x-4$ and a = 1.

Source: This is a combination of the "standard" examples of looking at a tangent line as the limit of secant lines, finding the equation for a tangent line to the graph of a function at a point on the graph, and plotting both the function and the tangent line on the same graph. We have already seen how to do this in *Excel* with a fixed function, fixed a – value, and an estimate for derivative via a difference quotient calculation. What makes this example nice is that it can be changed by redefining the function or a – value and a new graph will result, which can be changed dynamically (via the slider) to see how the secant lines approach the tangent line. *Mathematica*'s Manipulate function is used to build in the animation to the plot.

2. (*Creating Piecewise Functions Revisited*)
   a. Let f1[x]=x^(1/3). Using *Mathematica*, find each of the following: f1[1], f1[-1], f1'[x], f1'[1], and f1'[-1]. Then graph f1[x] and f1'[x] on the x-interval [-3,3]. What do you notice?

b. One way to fix the "problem" with the cube root function in *Mathematica* is to use a piecewise definition for the function. Define f2[x] with the following commands:

```
f2[x_]:= x^(1/3) /; x > 0
f2[x_]:= -(-x)^(1/3) /; x < 0
```

Repeat part (a). Is there any improvement?

c. Using commands such as If, Which, or Switch, create a cube root function r[x] that works the way the cube root is supposed to work, i.e. *Mathematica* can find r[1], r[-1], and plot both r[x] and r'[x] correctly.

d. Repeat part (c) for f[x]=(Abs[x]-2)^(1/3). You should get these graphs for y = f[x] and y = f'[x]:

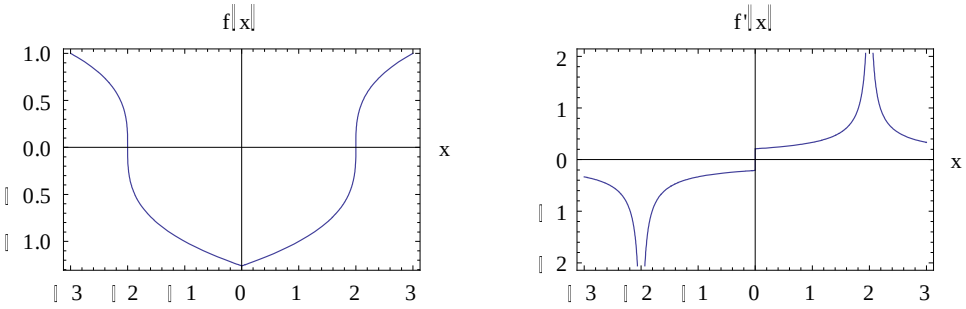

Figure 2: graphs for part (d)

Source: In one of the class assignments on optimization (from [19]), critical numbers of the function f[x]=(Abs[x]-2)^(1/3) are investigated. Since the function involves cube roots, one runs into one of *Mathematica*'s "bugs" – it treats all roots as complex and so when one computes the cube root of -1 with *Mathematica*, one gets the principal cube root 1/2 + i*√3/2 instead of -1. It is suggested in [19] to include the add-on package Miscellaneous`RealOnly` to fix this "problem", but for versions 6.0 and higher of *Mathematica*, this package is obsolete. My "calculus – level" solution to deal with this problem is to define the cube root function as a piecewise function for which cube roots of negative real numbers are computed correctly. I made this problem up to investigate how to create a piecewise-defined function that correctly evaluates all numbers in its domain, can be plotted correctly on its domain, can be differentiated correctly symbolically, and whose derivative can be evaluated and plotted correctly as well.

MATLAB and Linear Algebra Basics

1. *(Matrices in Excel)* Given the matrices $A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 6 \\ 0 & 1 & 4 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 1 \\ 3 & -1 \\ 0 & 2 \end{pmatrix}$, use *Excel* for each of the following.
   a. Find AB.
   b. Find det(A).
   c. Find A^{-1}.
   d. Show $AA^{-1} = A^{-1}A = I$.
   e. Solve AX = b, for b set equal to each column in B (Hint: this can be done in one step).
2. (*Matrices in Mathematica*) Repeat question 1 with *Mathematica*. Can you find a way to make matrices look like matrices?

Source: After looking at matrix basics, I want the class to understand the basics, see how to perform these calculations in *Excel* by "hand" or with built-in functions (MMULT and MINVERSE), and appreciate how well-suited *MATLAB* is for doing matrix calculations. I got the idea for the *Excel* material from [35] as the result of an internet search on matrix operations in *Excel*. The *Mathematica* questions are included so students can see how *Mathematica* uses lists for matrices, which is more cumbersome than arrays in *Excel*, but the commands to compute products and inverses are much better in *Mathematica* – thus there are tradeoffs for each type of software. For example, *MATLAB* is better for matrix computations, but *Mathematica* is better at displaying the output in exact form.

3. (*Calculus with MATLAB – appears on an in-class exam*) Let $g(x) = \exp\left[-\frac{1}{2}x^2\right]$.
   a. Find the actual derivative of this function, g'(x). You may do this by hand if you wish.
   b. Find the *numerical* derivative of g(x) on the interval [-1, 1], using an x – vector that contains 1000 points and includes the endpoints -1 and 1. Save the values of this numerical derivative as the vector *gprime*. Make a table with two columns that compares the first ten values of gprime (column 1) with the first ten values of g'(x) (column 2). *The table does not need any headings!*
   c. Use a Riemann sum to *numerically* estimate the definite integral $\int_{-1}^{1} g(x)\ dx$.

      Subdivide the interval [-1, 1] into 50 equal parts and choose the right endpoint of each subinterval as the sample point.

   Extra Credit: Find the integral $\int_{-k}^{k} g(x)\ dx$, with g(x) from problem 3 (c), *symbolically*. Then find the value of this integral if we replace k with infinity, two ways – first by letting k = 1, 10, 100, and 1000 and finding the integral's value in each case as a numeric rather than symbolic value (hint – look up "double") and second by finding the limit of the integral as k approaches infinity (hint – look up "inf" and "limit").

   Source: This problem illustrates some of the nice features of *MATLAB* for numerical differentiation as well as numerical integration via arrays of inputs and corresponding function values. The concept of Riemann sum is emphasized – in *MATLAB* these are "easy" to compute. In addition, the first part of this problem can be done symbolically within *MATLAB* with the Symbolic Toolbox (diff and int) or the Function Calculator. The idea for this problem comes from similar examples in [21].

8. Final Project Guidelines and Topics

Part of a student's course grade is based on a final project which includes a class presentation. The class is broken into four groups, with each group responsible for one of four possible projects chosen by me, taken from [34], to be presented during the last class period and final exam period.

Project	Title	Topic	Calculus Ideas Used	Area of Application
1	Fitting Lines to Data	Minimizing quadratic polynomials finds the regression line to a data set	Derivatives to find local extrema, logarithm and exponential functions	Statistics
2	Somewhere within the Rainbow	Minimizing Trigonometric functions to explain properties of a rainbow	Derivatives of trigonometric functions, finding maxima and minima	Optics, Meteorology
3	Measuring Voting Power	Integrals of polynomials give a measure of voting power	Average value of a function, integration of polynomials, mathematical induction	Political Science
4	How to Tune a Radio	Trigonometric integrals explain tuning a radio	Differentiation and integration of sines and cosines	Signal Analysis

Table 1: projects for fall 2008

Each project applies ideas from calculus to real-world problems or situations and can be investigated with the software tools studied in the course. Each group's responsibility is as follows:

- Read and understand the material outlined in the project handout.
- Answer all questions in the project handout's exercises, using at least one type of software discussed in the course, as appropriate.
- By no later than the beginning of the Final Exam period turn in a single set of solutions to the project handout's exercises, with all work done as neatly and completely as possible.
- Present the topic to the rest of the class (it is up to the group to determine how to do this). The presentation should include a handout for the rest of the class summarizing the project topic (for example, the handout could include key ideas and examples). The overall presentation should involve application of at least one type of software discussed in the course. *Each person in a group is expected to give a five – minute presentation on a portion of the material related to their group's project!*

9. Course Logistics

Each time I've taught this course, it has met three times per week on MWF, with a double period on Wednesday to account for a one-hour lab. The pilot version of the course in fall 2007 had twelve students, but since fall 2008 there have been 25 students enrolled in each section (course cap is 25 students), with two sections running each semester. I have found that reserving Fridays for lab days works best, with content taught on Mondays and Wednesdays. In addition, Wednesdays are used for in-class exams, so students have 100 minutes to take an exam, vs. 50 minutes. A student's course grade is based on the total points earned out of 600 possible, from homework and labs (250 points), three in-class exams (50 points each), class participation (100 points), and the final project (100 points).

The way I teach the material is via an interactive lecture format in which a concept is introduced via PowerPoint or other appropriate electronic format, such as *Mathematica* notebooks. Throughout the lecture examples are set up and solved by each student on their own computer, with me guiding them through the solution using my laptop connected to an overhead projection system. Lectures and files for examples are posted on line before each class so students can follow along – either on their computer or the overhead screen. I also use chalk/whiteboard to further illustrate ideas and answer questions.

One issue that has to be dealt with is students being at different levels of technical proficiency, both with computers and the software. To help with this, I have a graduate assistant (GA) assigned to the course. The GA attends each class period, learns the material, and helps with student questions which are technical in nature, as well as mathematical when appropriate (especially in the lab portion of the course). In addition to assisting in the class itself, the GA helps with grading of lab assignments. Labs are assigned on a Friday, due the following Friday, and turned in and graded electronically. This method saves paper and helps to keep track of which students have turned in their work, as well as the time it was turned in.

After completing material for each software package, I give an in-class exam with questions that require students to demonstrate mastery of the software at a basic level and use it to solve problems similar to those from lecture or lab assignments. For each exam students are allowed one side of a 5" by 8" index card of personal notes. The cards are turned in at the end of the exam. In addition, they can use the help files within *Excel*, *Mathematica*, or *MATLAB*, but are not allowed to access other materials such as old homework or labs, class lectures, or class notes. Most work and answers to questions on an exam can be done within the software, but some questions require a hand-written answer. Once a student has completed their exam, they email me and my GA (as a backup) their files with answers to questions including supporting work, turn in their note cards, and exam copies with hand-written answers. For grading of exams, I have found it works best to look at the student work within the electronic files and put comments and scores on the hard copies of the exam.

Any other work done by students, such as Final Projects or Group Projects, is turned in via email to me. Since these projects are done by four groups per class, I print out and grade hard copies of a group's submitted work. The only exception to electronic submission of work has been three small homework assignments at the beginning of each software section that ask students to list ten things they learn from a tutorial about the software.

10. Course Issues/Course Solutions

Since the course is different than "traditional courses", I have had to work out bugs as they arise – most of which can be classified as technological, logistical, software, or mathematical in nature.

a. Technological issues are those related to using computers in general, such as not being able to log into a network to get to the class web page, not knowing where a file is saved, trouble with saving a file in a certain format, and not knowing how to email a file.
b. Logistical issues are those related to running the course, such as finding ways to effectively collect, grade, and return student lab work and exams.
c. Software issues are those that relate directly to specific course software, such as getting errors while trying to work out an example in class or on homework problem.
d. Mathematical issues are those that relate to the course content, especially questions on labs and exams, such as being able to interpret a question, knowing how to start a question, figuring out how the question relates to the course lectures, and using the software to answer the question.

"Solutions": For each issue that has arisen, I have tried to address it right away if possible, or for the next version of the course, as appropriate to the specific issue.

a. Many of these issues occur only once and are unpredictable, but can usually be "fixed" immediately. Having a GA in the class to assist with these types of questions as they arise has worked well.
b. For the pilot version of the course, I tried using Blackboard for student submissions, with assignments due by the beginning of class. This worked fine until the *Mathematica* section, as Blackboard did not recognize *Mathematica* notebook files. After this, I had students turn in hard copies of their work, which was then graded and returned by hand.

For exams, students saved a copy of their exam to their COW laptop and emailed me a copy. For subsequent versions of the course, I modified schemes implemented by Rich Stankewitz when he taught the course (using my notes) in spring 2008 for assignment and exam collection, grading, and distribution. These are outlined above in Section 9: Course Logistics.

c. Most problems with the software boil down to entering incorrect syntax and are "easily" remedied once the syntax error is found. I have noticed that as each software section (and semester overall) progresses, these errors occur less frequently – this tells me the course is "working" at least on the software level.
d. Ignoring syntax issues, mathematical issues tend to arise from lab or exam questions that students do not understand or know how to interpret. Since I am making up or getting the questions from various sources (all at the level of first semester calculus or below) and tailoring them to this course, the confusion can sometimes be due to a poorly worded or stated question. With this in mind, I revise, rewrite, or throw out questions as needed and provide liberal hints in lab, office hours, and via email.

11. Future Plans for the Course

Overall, I feel that the course has been a success. This is based on feedback from students both in person as well as on evaluations, comments from colleagues who have taught the course, student performance on exams, student pass rate (77% of my MATHS 159 students have earned a grade of C or higher, vs. 43% earning a grade of C or higher in my freshman calculus courses), departmental support for the course, and my perception of how students have matured both mathematically and with the software as the course has progressed – for example by the end of the course they are much more likely to use *Mathematica* to attack a problem (this is what most have chosen for their final projects). Since MATHS 159 is in its infancy, no formal assessment of the course's impact on other courses has been made – informal input from colleagues who teach courses that use the software suggest that some changes have occurred in student preparation for software use – in particular more students in our introductory statistics course report familiarity with *Excel* and our students in numerical analysis are more confident and fluent with *MATLAB* than in the past.

One issue that has arisen that is related to the mathematical issues above, as well as to the placement of the course in our curriculum is that a student's perceptions of the course may depend on whether the student is freshman or upper class. The general sentiment from students who took the course in fall 2007 was that the course was valuable to them, they learned a lot, and they wished it would have been offered earlier in their college career (this is the course with almost all upper class majors). The main complaint for that year (which I get for many of my courses) was essentially that I gave too much work. For the second version of the course in fall 2008, there were twice as many students, of which all but one were first – semester freshman. Even though students did well in the course, the general feeling I got from the few comments on evaluations this time was that students did not see the value of the course to their major. It will be interesting to see how students feel about the courses I taught in fall 2009, as there is a mix of about half freshman and half upper class students.

Another unexpected issue that has popped up is tied directly to departmental resources available to run the course – requiring all of our majors to take MATHS 159 as a freshman coupled with having to cap the course at 25 students per section to run an effective course and only being able to run two sections per semester has led to a potential "bottleneck" for students entering our program. This is exacerbated by the fact that if a student is not successful in their first semester of calculus, they may change majors – these students are taking a spot in MATHS 159 that could be filled by a student who will actually complete a mathematics degree.

For this reason, our department has implemented curriculum revisions so that MATHS 159 will be taken in a students' sophomore year after taking a semester of calculus and a semester of

discrete mathematics. Starting in fall 2010, this new version of the course will be offered as MATHS 259. We are currently looking at what adjustments (if any) need to be made to the course to address its repositioning in our programs. In addition to being able to offer the course to more of our majors, this move should also help with students' perceived value of the course to their major. As the course progresses, in whatever form it takes, many more examples, homework problems, exam questions, and final projects will have to be created. I look forward to the challenge of continuing to find ways to help increase students' awareness of the usefulness of software for their mathematical careers!

12. Supplemental Electronic Files

Mathematica

- Karls, M. (2009). Tangent Line Example. *Mathematica* notebook for illustrating the tangent line to a function at a point as a limit of secant lines through the point. Requires *Mathematica 6.0* or higher, [41]. (TangentLineExample.nb)

MATLAB

- Karls, M. (2010). Competitive Hunters Example. MATLAB M-files for illustrating a competitive hunters model as well as the idea of calling a function M-file from within another M-file. Requires *MATLAB*, [36]. (cancan m-file), (hunter m-file)
- Karls, M. & Stankewitz, R. (2010). Plotting Toad Data. *MATLAB* M-file for showing how to plot toad data along with approximating linear and exponential curves that fit the data. Requires *MATLAB*, [36]. (toads1.m)

Octave

- Karls, M. (2010). Competitive Hunters Example. *Octave* M-files for illustrating a competitive hunters model as well as the idea of calling a function M-file from within another M-file. Requires *Octave*, [12], as well as Octave M-file ODE45.M, by T. Treichl, listed below. (cancan m-file), (hunter m-file)
- Karls, M. & Stankewitz, R. (2010). Plotting Toad Data. *Octave* M-file for showing how to plot toad data along with approximating linear and exponential curves that fit the data. Requires *Octave*, [12]. (toads1.m)
- Treichl, T. (2009). ODE45.M. Octave M-file that mimics MATLAB M-file ODE45.M. This file is required in addition to CANCAN.M and HUNTER.M in order to run the Competitive Hunters Example with the Octave M-files CANCAN.M and HUNTER.M listed above. Requires *Octave*, [12]. (ode45 m-file for *Octave*)

PDF

- Karls, M. (2009). Tangent Line Example. PDF version of *Mathematica* notebook TangentLineExample.nb. (TangentLineExample.pdf)
- Karls, M. & Stankewitz, R. (2010). Plotting Toad Data. PDF version of *MATLAB* or *Octave* M-file toads1.m. (toads1.pdf)
- Karls, M. & Treichel, T. (2010). Competetive Hunters Example. PDF version of MATLAB or Octave m-files CANCAN.M, HUNTER.M, and ODE45.M. (cancan.pdf)

References

1. Anton, H. & Rorres, C. (1994). *Elementary linear algebra* (7th ed.). New York, NY: John Wiley and Sons, Inc.
2. Ball State University. (2010). 2008-10 Undergraduate Catalog. [WWW page]. URL http://www.bsu.edu/web/catalog/.
3. Ball State University. (2010). Admission requirements. [WWW page]. URL http://cms.bsu.edu/AdmissionsLanding/UndergraduateAdmissions/AdmissionRequirements.aspx.
4. Ball State University. (2009). Fact book. [WWW page]. URL http://www.bsu.edu/factbook/.
5. Ball State University. (2010). Graduate Catalog. [WWW page]. URL http://cms.bsu.edu/Academics/CollegesandDepartments/GradSchool/Academics/GraduateCatalog.aspx.
6. Blanchard, P. (1994). Teaching differential equations with a dynamical systems viewpoint. College *Journal of Mathematics, 25* (5), 385-393.
7. Burkhardt, W. (1993). First *steps in Mathematica.* Berlin, Germany: Springer-Verlag.
8. Cohen, J. and Hagin, F. (1995). *Calculus explorations with Mathematica*. Englewood Cliffs, NJ: Prentice Hall.
9. Department of Mathematical Sciences. (2007). *2008-10 Course and program changes overview*. Memo to Dean Susan Johnson and the College of Sciences and Humanities Curriculum Committee, March 14, 2007.
10. Davis, T. & Sigmon, K. (2004). *MATLAB primer* (7th ed.). Boca Raton, FL: Chapman and Hall/CRC.
11. Don, E. (2000). *Schaum's outline of Mathematica*. Boston, MA: McGraw-Hill.
12. Eaton, J. W. *Octave*. [WWW page]. URL http://www.gnu.org/software/octave/
13. Emert, J. & Nelson, R. (1992). *Calculus and Mathematica*. Fort Worth, TX: Saunders College Publishing.
14. French, T. *Excel IF functions - Using IF functions in Excel spreadsheets*. About.com. [WWW page]. URL http://spreadsheets.about.com/od/excelfunctions/a/if_funct_hub.htm
15. Gilat, A. (2004). *MATLAB: An introduction with applications* (2cd ed.). Hoboken, NJ: Wiley.
16. Greg, H. (2007). *Excel 2003 for dummies*. Hoboken, NJ: Wiley.
17. Hanselman, D. & Littlefield, B. (2004). *Mastering MATLAB 7.* Upper Saddle River, NJ: Prentice Hall.
18. Hillier, F. S. & Lieberman, G. J. (2005). *Introduction to operations research* (8th ed.). New York, NY: McGraw-Hill.
19. Hollis, S. (2003). *CalcLabs with Mathematica*. Pacific Grove, CA: Brooks/Cole.
20. James, B.W. (2009). *Excel tutorial*. [WWW page]. URL http://people.usd.edu/~bwjames/tut/excel/.
21. Jenson, G. (2000). *Using MATLAB in calculus*. Upper Saddle River, NJ: Prentice Hall.
22. Karls, M. (2008). *Mathematica Tutorial*. [Web page]. URL http://www.bsu.edu/web/mkarls/MathematicaTutorial08.nb.
23. Karls, M. (2008). MATHS 159: Introduction to mathematical software. [WWW page]. URL http://www.bsu.edu/web/mkarls/159fa08.htm.
24. Lewand, R.E. (2000) *Cryptological Mathematics*. Washington, DC: Mathematical Association of America Press.
25. Lial, M. L., Miller, C. D., & Greenwell, R. N. (1993). *Finite Mathematics and Calculus with Applications* (4th ed.). New York, NY: Harper Collins College Publishers.
26. Livshits, I, Karls, M., and Stankewitz, R. (2006). *Computational mathematics initiative – Student involvement in research and outreach*. Ball State University Enhanced Provost Initiative Grant Proposal. Submitted October 17, 2006. Awarded January, 2007.

27. Microsoft Office Online. (2009). Microsoft *Excel* – math and trigonometry functions. [WWW page]. URL http://office.microsoft.com/en-us/excel/CH100645361033.aspx
28. Microsoft Office Online. (2009). Microsoft *Excel* – list of worksheet functions. [WWW page]. URL http://office.microsoft.com/en-us/excel/HP100791861033.aspx
29. Mooney, D. & Swift, R. (1999). *A first course in mathematical modeling*. Washington, DC: Mathematical Association of America Press.
30. Olinick, M. (1978). *An Introduction to mathematical models in the social and life sciences*. Reading, MA: Addison-Wesley.
31. Piascik, C. (1999). *Applied calculus with Microsoft Excel*. Pacific Grove, CA: Brooks/Cole.
32. Stewart, J. (2003). Calculus *(early transcendentals)*. (5th ed.). Belmont, CA: Brooks/Cole.
33. Sigmon, K. (1993). *MATLAB primer* (3rd ed.). Gainesville, FL: Department of Mathematics. University of Florida.
34. Straffin, P. (Editor). *Applications of calculus: Volume 3*. *MAA Notes (29)*. Washington, DC: Mathematical Association of America Press.
35. Systems of linear equations and matrices *Excel* files. (2008) Supplement to Finite Mathematics, (7th ed.), by Margaret Lial et al. [WWW page]. URL http://wps.aw.com/aw_lial_finitemath_7/0,1769,12520-,00.html.
36. The Math Works. (2010). *MATLAB* and Simulink for Technical Computing. [WWW page]. URL http://www.mathworks.com/.
37. University of Utah. (2009). *MATLAB* tutorial. [WWW page]. URL http://www.math.utah.edu/lab/ms/matlab/matlab.html/.
38. Walkenbach, J. (2007). *Excel 2007 bible*. Hoboken, NJ: Wiley.
39. Wolfram, S. (1991). *Mathematica: A system for doing mathematics by computer* (2cd ed.). Reading, MA: Addison-Wesley.
40. Wolfram, S. (2003). The *Mathematica book* (5th ed.). Champaign, IL: Wolfram Research.
41. Wolfram Research. (2009). [WWW page]. URL http://www.wolfram.com/.
42. Wolfram Research. (2009). Wolfram screencast and video gallery. [WWW page]. URL http://www.wolfram.com/broadcast/.