Spreadsheets in Education (eJSiE)

Volume 2 | Issue 1 Article 9

January 2006

Using Spreadsheets to Learn Numerical Methods

Christopher J. Van Wyk
Drew University, New Jersey, USA, cvanwyk@drew.edu

Recommended Citation
Van Wyk, Christopher J. (2006) Using Spreadsheets to Learn Numerical Methods, *Spreadsheets in Education (eJSiE)*: Vol. 2: Iss. 1, Article 9.
Available at: http://epublications.bond.edu.au/ejsie/vol2/iss1/9

Using Spreadsheets to Learn Numerical Methods

Abstract
The common spreadsheet is an excellent tool for students in a first course in numerical methods. It makes it easy to construct illustrative examples, to experiment with them, and to plot results in graphi-cal form. For the students in a class in numerical methods who are majoring in a subject other than computer science (e.g., physics, engineering, or economics, spreadsheets offer an attractive alternative to conventional programming that allows ready experimentation with numerical algorithms. All stu-dents, in fact, no matter what their major, seem to appreciate the immediate display of results that spreadsheets allow.

Keywords
numerical methods

Using Spreadsheets to Learn Numerical Methods

Christopher J Van Wyk
Department of Mathematics and Computer Science
Drew University
Madison, NJ 07940
cvanwyk@drew.edu

Abstract

The common spreadsheet is an excellent tool for students in a first course in numerical methods. It makes it easy to construct illustrative examples, to experiment with them, and to plot results in graphical form. For the students in a class in numerical methods who are majoring in a subject other than computer science (e.g., physics, engineering, or economics), spreadsheets offer an attractive alternative to conventional programming that allows ready experimentation with numerical algorithms. All students, in fact, no matter what their major, seem to appreciate the immediate display of results that spreadsheets allow.

1 Introduction

Most textbooks on numerical methods present algorithms in an Algol-like pseudocode (e.g., [1, 4]), although some use symbolic-algebra systems (e.g., [2]) or specialized mathematical-modeling software (e.g., [3]). Many numerical algorithms are readily expressed as loops – either indefinite (**while** error is still large **do** …) or definite. Unrolling either kind of loop into successive rows of a spreadsheet makes it easy to watch numerical algorithms at work. Sections 2 – 7 use spreadsheets to illustrate some important basic points in numerical methods. Section 8 summarizes the lessons and mentions some topics that are *not* well suited to implementation on spreadsheets.

A Microsoft Excel 2002 spreadsheet that contains all of the illustrative examples in this paper is available at www.users.drew.edu/~cvanwyk/NA_SS_examples.xls. It is best to read this paper with an open spreadsheet program into which you can type the examples.

2 Floating-point Numbers

The sequence of negative powers of two, $x_i = 2^{-i}$, is a familiar example of a sequence of real numbers that converges to, but never reaches, zero. To see the very different behavior of floating-point numbers, fill in two cells of a spreadsheet as shown in Spreadsheet Formula 2.1, then copy the shaded box down (as indicated by the dark arrow) to fill the next 1023 or so rows. Because the formula in cell A2 contains a relative reference, the copy operation leaves the number on each line equal to one-half the number above it, so row i should contain $x_i = 2^i$. On row 1023, though, the number is zero! (See Spreadsheet 2.2, which displays the contents of column A to the maximum possible

Spreadsheet Formula 2.1. Negative powers of two

	A
1	=1/2
2	=A1/2
3	↓

Spreadsheet 2.2. Rows 1020-1023 from Spreadsheet Formula 2.

1020	8.900295434028810000000000000000E-308
1021	4.450147717014400000000000000000E-308
1022	2.225073858507200000000000000000E-308
1023	0.000000000000000000000000000000E+00

Spreadsheet Formula 2.3. Positive quantities vanish

	A	B
1	=1/2	=1+A1
2	=A1/2	↓
3	↓	↓

precision.) The place just before the numbers in column A vanish indicates the lower end of the *range of exponents in floating-point representation*.

The sum of two positive real numbers is always a real number that is larger than either of them. To see that this does not hold for floating-point numbers, add a second column to the above spreadsheet, as shown in Spreadsheet Formula 2.3. After around fifty rows, the numbers in column A are still positive, but the numbers in column B stop being larger than one. The number in column A of the row just before this happens is often called *machine epsilon*.

2.1 Exercises

- Try Spreadsheet Formulas 2.1 and 2.3 on different software packages or different machines. Discuss the implications of different exponent ranges and numerical precisions for writing portable software.
- The numbers in Spreadsheet 2.2 suggest that the output routines that convert internal representations to (printable) decimal form did not work past the 15th decimal digit, even though more digits were requested. Devise a formula to retrieve more of the significant digits of the numbers in Spreadsheet 2.2, or at least to ascertain that those digits are, in fact, non-zero.

Spreadsheet Formula 3.1. Bisection algorithm

	A	B	C	D	E	F
1	a	=(A1+C1)/2	b	=f(A1)	➔	➔
2	=IF(SIGN(D1)<> SIGN(E1),C1,B1)	↓	=IF(SIGN(D1)<> SIGN(E1),C1,B1)	↓	↓	↓
3	↓	↓	↓	↓	↓	↓

3 Root-Finding

If f is a continuous real-valued function on real numbers that is positive at one end of an interval and negative at the other end, then f has a root somewhere in the interval. By examining the sign of f at the midpoint of the interval, we can halve the length of the interval known to contain the root. This simple observation leads directly to the *bisection algorithm* shown as Spreadsheet Formula 3.1.

Here is how to use this spreadsheet formula:

- Enter into cells A1 and C1 the left and right endpoints of the interval $[a,b]$ that contains the root.
- Enter the formula for the midpoint in cell B1 and copy it down to cell B2.
- Enter the formula for f in cell D1. For example, if $f(x) = x^2 - 2$, type =A1*A1 – 2 in cell D1.
- Copy cell D1 across to cells E1 and F1 and down to D2 through F2; because D1 uses relative references, E1 and F1 will contain the values of function f at B1 and C1, respectively.
- Enter into cells A2 and C2 the formulas for the endpoints of the half-interval that contains the root.
- Copy cells A2 through F2 down for as many lines as you like.

Making column B wide and formatting it so that many digits are displayed will reveal that, every three or four lines, one more digit of the answer stops changing. This happens because bisection is *linearly convergent* – each iteration yields one more bit of the correct answer, and one decimal digit corresponds to between three and four bits (since $3 < \log_2 10 < 4$).

If f is differentiable and x_0 is a good initial guess at the location of its root, we can use *Newton's method* to get a better estimate of the root, as shown in Spreadsheet Formula 3.2. The initial guess goes into cell A1, and the formulas for f and its derivative go into cells B1 and C1, respectively; then these cells are copied down their columns as before.

Spreadsheet Formula 3.2. Newton's method

	A	B	C
1	x_0	=f(A1)	=f '(A1)
2	=A1-B1/C1	↓	↓
3	↓	↓	↓

Making column A wide and formatting it to display many digits reveals that after a couple of iterations, the number of digits of the correct answer approximately doubles with each additional row. This happens because Newton's method is *quadratically convergent* when the initial guess x_0 is close to a root. (Unfortunately, if the initial guess is not close to a root, Newton's method can diverge, and the values in column A may display chaotic behavior.)

3.1 Exercises

- Spreadsheet Formula 3.1 evaluates f several times at the same point. Revise it to calculate f only once at each different value of its parameter.
- Program another root-finding algorithm, such as the *secant method* or *regula falsi*. Try to set up the spreadsheets so that the function f whose root is sought can be changed simply by replacing the formulas in a single contiguous block of cells (as one can for Spreadsheet Formulas 3.1 and 3.2).
- Add to the spreadsheet a graph that shows the values of the function at successive approximations to the root. Such graphs can be especially entertaining when an algorithm fails to converge.

4 Taylor Series

Spreadsheets make it convenient to observe infinite series in the act of convergence. For example, Spreadsheet Formula 4.1 can be used to compute the exponential function on the value in cell B1. For most positive values of B1, the shaded cells won't need to be copied down for very many lines, because the Taylor series for $e^x = \sum_{i\geq 0} \frac{x^i}{i!}$ converges so rapidly. The formula in cell C4 contains an absolute reference to \$B\$1 so that each row of column C includes one more power of x than the row above it. This technique, as well as the reference to column A in the denominator, illustrates the *incremental construction of the summation expression*, which is important to avoid the moral equivalents of `pow(x,i)` and `fact(i)` that are suggested by the conventional mathematical notation of the Taylor series.

Spreadsheet Formula 4.1. Taylor series for exponential function

	A	B	C	D
1	x=	x		
2	n		TS nth term	TS nth order
3	0		1	=C3
4	=A3+1		=C3*$B$1/A4	=D3+C4
5	↓		↓	↓

Spreadsheet Formula 4.2. Slowly converging Taylor series for $\ln\left(\frac{1}{1+x}\right)$

	E	F	G	H
2	alternator	nth power	TS nth term	TS nth order
3	1	1	0	=G3
4	=0-E3	=$B$1*F3	=E4*F4/A4	=H3+G4
5	↓	↓	↓	↓

Of course, not all Taylor series converge so quickly. Spreadsheet formula 4.2, for example, displays the very different *convergence behavior of the Taylor series* for $\ln\left(\frac{1}{1+x}\right) = \sum_{i>0} \frac{(-x)^i}{i}$. This series converges when $|x| < 1$, but not very quickly when x is near one. When B1 contains 0.5, for example, the numbers in column E stop changing after about twenty rows. But when B1 contains 0.999, the fourth digit after the decimal point in column E is still changing after a thousand lines.

4.1 Exercises

- Compare the convergence of the Taylor series for e^{-1} with the convergence of the reciprocal of the Taylor series for e.
- Program the Taylor series for $\sin\theta = \sum_{i\geq 0} \frac{(-1)^{i+1}\theta^{2i+1}}{(2i+1)!}$.

5 Polynomials

For calculation, it is better to write the polynomial $p(x) = a_2x^2 + a_1x + a_0$ in *nested multiplication* form as $p(x) = (a_2x + a_1)x + a_0$. Spreadsheet Formula 5.1 is a row-wise implementation and generalization of this algorithm that leaves row N+2 of column B containing the value of the Nth-degree polynomial $p(x)$ when cell B1 contains the value

Spreadsheet Formula 5.1. Horner's rule

	A	B
1	X=	x
2	a_n	=$A2
3	a_{n-1}	=$A3+B2*B$1
	…	↓
N+2	a_0	p(x)

Spreadsheet Formula 5.2. Neville's algorithm
[The contents of the slashed cells are ir-relevant to the computation.]

	A	B	C	D	E	F	G
1						X=	x
2	x_0	y_0	=(B3-B2)/(A3-A2)	=(C3-C2)/(A4-A2)	=(D3-D2)/(A5-A2)	=G$1-$A2	=$B$2+G3
3	x_1	y_1	↓	↓		=G$1-$A3	=($C$2+G4)*F2
4	x_2	y_2				=G$1-$A4	=($D$2+G5)*F3
5	x_3	y_3				=G$1-$A5	=$E$2*F4

of x and column A contains the coefficients of polynomial $p(x)$. The absolute references to column A and row 1 in column B make it convenient to evaluate $p(x)$ at several different places: simply copy column B into columns C, D, and so on, then type a different value of x into row 1 of each column.

Given a set of points (x_i, y_i) with distinct x-coordinates, the *divided-difference table* shown in Spreadsheet Formula 5.2 can be used to construct the *interpolating polynomial* that passes through the points. If cell G1 contains a value x, we can find the value of the interpolating polynomial $p(x)$ as follows. First, fill column F with the differences $x - x_i$; next, perform nested multiplication *upwards* on the differences in column G; finally, cell G2 will contain $p(x)$. As in Spreadsheet Formula 5.1, the formulas in columns F and G can be replicated in columns to the right in order to evaluate the interpolating polynomial at several abscissae. Of course, it is always prudent to verify that the polynomial does indeed pass through the points, i.e., $p(x_i) = y_i$.

5.1 Exercises

- Generalize Spreadsheet Formula 5.2 to construct a quartic polynomial that passes through five given points, or a quintic that passes through six. Plot the resulting polynomials and observe their oscillation between the given points.
- Implement one of the many schemes for *numerical integration* that are based on interpolating a low-degree polynomial through carefully chosen points on the curve of the function being integrated.

Spreadsheet Formula 6.1. Gauss-Seidel iteration

	A	B	C	D	E
1	matrix			calc RHS	given RHS
2	a_{11}	a_{12}	a_{13}	=SUMPRODUCT (A2:C2,$A$6:$C$6)	b_1
3	a_{21}	a_{22}	a_{23}	↓	b_2
4	a_{31}	a_{32}	a_{33}		b_3
5	vector				
6	=E2-D2+A2*A6)/A2	=(E3-D3+B3*B6)/B3	=(E4-D4+C4*C6)/C4		

6 Systems of Linear Equations

A spreadsheet program that allows formula cells to include circular references can use Gauss-Seidel iteration to solve this linear system:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} A6 \\ B6 \\ C6 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

The equation above follows the familiar convention of writing the solution vector as a column vector. Spreadsheet Formula 6.1 writes it horizontally, however, so it can use the built-in SUMPRODUCT function to calculate the dot product of each row of the matrix with the solution vector, forming the calculated right-hand side in column D. Note that Gauss-Seidel iteration, which updates each element of the solution vector as soon as a better estimate is available, is especially well suited to spreadsheet implementation.

While iterative methods are not guaranteed, in general, to converge to a solution, there are conditions under which they are known to converge. An important example of such a condition is when the coefficient matrix is *diagonally dominant* – on each row, the sum of the off-diagonal elements is smaller in magnitude than the diagonal element.

6.1 Exercises

- *Jacobi iteration* updates the solution vector only after all new elements are available. Discuss why it is not as well suited as Gauss-Seidel iteration to spreadsheet implementation.
- *Cubic splines* are piecewise interpolating polynomials whose first and second derivatives match where consecutive polynomials meet. The matrix that determines cubic spline coefficients is diagonally dominant and *tridiagonal* – its only nonzero elements lie on the main diagonal and on the two diagonals on either side of it. Show how to use Gauss-Seidel iteration to solve a tridiagonal system. (Hint: the three non-zero diagonals can be stored in three adjacent spreadsheet columns, leading to an implementation that is much more easily generalized to an $n \times n$ system than the matrix form in Spreadsheet Formula 6.1.)

Spreadsheet Formula 7.1. Euler's method

	A	B	C	D
1	delta-t =	Δt		
2	t	y	y'	y"
3	0	y_0	v_0	-9.8
4	=A3+$A$1	=B3+C3*$B$1	➔	⬇
5	⬇	⬇	⬇	⬇

7 Differential Equations

The techniques of numerical methods can be brought to bear to find *numerical solutions to differential equations*. To solve the second-order system

$$y'' = -9.8\mathrm{m/s}^2,\ y'(0) = v_0,\ y(0) = y_0,$$

Spreadsheet Formula 7.1 uses Euler's method to approximate $y(t+\Delta t) \approx y(t) + y'(t)\Delta t$ and $y'(t+\Delta t) \approx y'(t) + y''(t)\Delta t$. For this simple example, the approximate solution and the exact solution $y(t) = -4.9t^2 + v_0 t + y_0$ can be plotted on the same graph and compared.

7.1 Exercises

- Euler's method uses a first-order Taylor series to advance in time. Implement a higher-order approximation.
- Implement a solution to the Lotke-Volterra system of differential equations, including a plot showing changes in predator and prey populations. Experiment with different parameter settings.
- Implement a solution to the diffusion equation showing the temperature at points evenly spaced along a rod. Use both a *marching method* (where the approximation at each point depends only on the values computed at earlier time steps) and an *implicit method* (where the approximation at each point depends on values computed at nearby points; e.g., Crank-Nicolson).

8 Summary

These examples show how spreadsheets can reveal the inner workings of a variety of numerical methods. Some textbooks, in fact, include tabular displays to show how a method changes the variables in a particular problem. *Creating spreadsheets that reproduce the tables in a textbook* can help students to understand how the values of variables are used in a numerical method. After doing this, students can experiment further with the numerical methods thus implemented, watching what happens when any number in the problem – initial guess, function parameter, interpolation point, element of the matrix or right-hand side, or initial condition – is changed. Small spreadsheet programs like these suffice to solve many calculation-oriented textbook exercises.

Among the advantages spreadsheets offer are ease of programming (including relative references that are often ideally suited to numerical methods, and automatic renaming as rows and columns are introduced and removed), and immediate recalculation upon changes to any number. Among their disadvantages are clumsy variable names that are easily confused and floating-point arithmetic with sometimes dubious properties. While the latter flaw, in particular, makes spreadsheets an unlikely medium for industrial-strength implementations of numerical methods, the advantages make them an excellent tool for beginners.

Of course, numerical methods are about more than looking for numerical patterns and coincidences. Students who have engaged the material by translating mathematical notation into spreadsheet or program code and doing numerical experiments still have a lot to learn about numerical methods, many of them not at all suited to spreadsheet implementation. Examples include:

- analysis of accumulating rounding error in floating-point computation;
- proofs of convergence properties of infinite sequences and series;
- "backwards error analysis" – problem sensitivity to small changes in input parameters;
- calling functions in libraries (note that spreadsheet libraries include many useful functions, as well as "solver" and "optimization" routines);
- Gaussian elimination with partial pivoting.

For topics like these, conventional programming languages, higher-powered mathematical software, and even chalk on a blackboard can be more appropriate. Nevertheless, spreadsheet software is so convenient and readily available that I find myself turning to it to illustrate a point at least once in every class period.

Acknowledgments

Thanks to Jon Bentley, Steve Kass, and Brian Kernighan for careful reading and helpful suggestions. Thanks also to the anonymous referees, some of whom provided the following references:

A. J. Davies and D. Crann, "Numerical methods on a spreadsheet", *Proceedings of the International Conference on. Technology in Collegiate Mathematics* 12(2001), 88-92.

D. Green, "Numerical methods using a spreadsheet", *Teaching Mathematics and Applications* 13(1994), 179-185.

D. McLaren, *Spreadsheets and Numerical Analysis*, Chartwell-Bratt, 1996.

E. Neuwirth and D. Arganbright, *The Active Modeler: Mathematical Modeling with Microsoft Excel*, Brooks-Cole, 2004.

References

[1] W. Cheney and D. Kincaid, *Numerical Methods and Computing*, 4th ed., Brooks/Cole, 1999.

[2] J. D. Faires and R. Burden, *Numerical Methods*, 2nd ed., Brooks/Cole, 1998.

[3] J. H. Mathews and K. D. Fink, *Numerical Methods Using MATLAB*, 3rd ed., Prentice Hall, 1999.

[4] C. Pozrikidis, *Numerical Computation in Science and Engineering*, Oxford, 1998.

P.S.

Many computer science courses can benefit from exploration facilitated by spreadsheets. Try filling column A with all ones, filling row 1 (except for cell A1) with all zeros, putting the formula =A1+B1 into cell B2, then copying B2 over and down to a 20×20 square, thus implementing the recurrence formula for binomial coefficients (Pascal's triangle). Plotting row 21 with a linear vertical scale yields a good approximation to the normal (bell-shaped) curve. What do you think the picture will look like if you plot using a logarithmic vertical scale?