Functions and Parameterizations as Objects to Think With

Roger Kraft
Purdue University Calumet

Abstract

At Purdue University Calumet we require all mathematics majors to take a sophomore level course on using and programming Maple. The course has three goals, to help students become adept at using Maple as a problem solving tool, to introduce students to concepts from computer programming, and to act as a ``bridge course" that uses Maple's CAS features to help students improve their understanding of important basic mathematical concepts. This paper describes two modules from the course, one on functions and the other on parametric curves and surfaces, and uses the modules to give a sense of the content and methods of the course. A further goal of this paper is to encourage people to think about the value of implementing such a course as part of their school's curriculum.

1. Introduction

Over time, computer algebra systems are only going to become more important for doing and teaching mathematics. Because of this, I believe that the time has come when every mathematics major should take a course specifically about using and programming a computer algebra system. This paper describes two modules from such a course about Maple. One module is about Maple's arrow notation for defining functions. This module shows how learning about a computer algebra system can lead students to think deeply about basic, but often poorly understood, concepts like functions, variables, parameters, and evaluation. The other module described here is about parametric curves and surfaces.

Let me begin by giving a few reasons why I think that every mathematics major should take a course in using and programming a computer algebra system like Maple.

First, as computer algebra systems become more powerful and easier to use they are becoming more useful for mathematics at every level, from industrial applications and academic research to undergraduate and secondary teaching. No matter what profession a mathematics major goes into, computer algebra systems are likely to be a part of their future.

For mathematics majors going into industry, experience with using and programming Maple will be a valuable skill that may in the near future come to be expected; it would make sense that the mathematician on a multidisciplinary team should be the most knowledgeable about using a CAS. Mathematics majors should learn to use and program Maple much as engineering majors learn traditional programming languages like C, C++, or Java; a computer algebra system should be a standard problem solving tool to be used by mathematics majors just as a programming language is a tool that engineering students are expected to be able to use in their profession.

For mathematics education majors, learning to use Maple will prepare them for the inevitable day when computer algebra systems are as ubiquitous in the secondary school curriculum as graphing calculators are now [2]. Teaching high school mathematics using a computer algebra system will not be easy [1]. Future teachers should to be taught in college how to use a sophisticated computer algebra system like Maple. Then they will be prepared to work with the computer algebra systems that are beginning to appear now in calculators.

A second reason for requiring a course in Maple is that it can help spread the use of computer algebra throughout the mathematics curriculum. If a mathematics program requires every student to take a course in Maple in, say, the sophomore year, then Maple becomes a tool, like calculus and linear algebra, that can be used freely in all subsequent courses. This would remove a lot of the friction that is inherent in trying to integrate computer algebra into advanced courses. Many schools currently use Maple in courses like calculus, linear algebra, and differential equations. But when these courses use Maple, they invariably state that the course "is not a course in Maple" or the course "is not about programming in Maple". Since these courses rightfully have their focus on specific mathematical content, faculty teaching these courses want to minimize the time spent presenting details about Maple. The use of Maple in one of these courses can be a great benefit for students, but using Maple in this way may not leave students with any real understanding or broad knowledge of the computer algebra system itself and may not prepare them for using Maple in other courses. On the other hand, a specific course just on Maple, with well thought out materials and activities, can be a resource that any number of advanced math courses can efficiently and confidently rely on.

A third reason for a course specifically about Maple is that learning to use a computer algebra system can help students raise their level of mathematical maturity. The nature of a computer algebra system lends itself to helping students overcome conceptual difficulties they commonly have with basic mathematical ideas. For example, when using a computer algebra system, everything must be stated precisely using the language's syntax which must be more careful and precise than standard mathematical notation. We can use specific aspects of Maple's syntax to draw students' attention to ideas and concepts that are basic to all of mathematics but that students are often unaware of or misunderstand.

Here are two simple examples of the interplay between syntax and concepts. The common mathematical notation for defining a function, $\mathrm{f}(x) = x^2 - 1$, blurs the distinction between defining and naming a function. The Maple notation, **`f := x->x^2-1`**, clearly makes this distinction and opens the way to understanding the idea of anonymous functions, like **`x->x^2-1`**, which will help students come to grips with the idea of spaces of (mostly anonymous) functions. For the second example, suppose we define a function $\mathrm{f}(x) = x^2 - 1$ and then somewhat latter write the expression $\mathrm{f}(x) = x$. How are we to interpret this second expression? Is it a redefinition of the function f, or is it a shorthand for the (fixed point) equation $x^2 - 1 = x$? In Maple, the distinction between a definition and an equation is clear; redefining the function would use the assignment operator (**`:=`**) but an equation would use the equality operator (**`=`**). In a standard mathematics text, the distinction between definitions and equations needs to be made by the context that the expressions appear in. But most undergraduate students are not even aware of this distinction and do not realize the importance of context. Having to make this distinction in Maple will help students become aware of it when they read mathematics texts.

Many important mathematical ideas and concepts (such as variables, parameters, functions, equations, evaluation, simplification, naming, symbolic vs. numeric results) have ways of being expressed within a computer algebra system. Learning to express these concepts using the language of a computer algebra system will help students to better understand the concepts. Another way to put this is that a course in using and programming Maple can be a type of "bridge course" helping mathematics majors to deepen their mathematical maturity and prepare themselves for more advanced pure and applied mathematics courses.

Note: A few of the sections from this paper are available in the Maple worksheet version of this document. They were removed from this version to save space.

2. Functions as objects to think with

This section describes a module on functions taken from our Maple course. This module emphasizes the third goal of our course, helping students bridge the conceptual gap between upper and lower level mathematics courses. This part of the course helps students build up their understanding of functions by having them work on a series of problems about Maple's arrow operator. By asking students to understand and work with ideas like anonymous, recursive, and higher order functions, the exercises stress the distinctions between defining, naming, and evaluating functions. This module is an example of how Maple's syntax and its abilities as a CAS provide opportunities to get students to think carefully about "elementary", but subtle, concepts such as variables, parameters, naming, equations, functions, evaluation, and simplification.

2.1. Maple Functions

There are two ways to represent mathematical functions in Maple, as Maple expressions and as Maple functions. Each way has it advantages and disadvantages. The differences between these two representations can be subtle and non-obvious. A full explanation of the differences between expressions and Maple functions requires an understanding of evaluation rules, data structures, procedures, local, global, and lexical variables, parameter passing, remember tables, and a few other ideas. All of the examples and exercises below make use of Maple functions since Maple functions are a bit more accurate than expressions at representing mathematical functions and, more importantly, it is the notation for Maple functions that is useful for getting students to think carefully about mathematical functions in general.

A Maple function is defined using arrow notation. Here are several examples of Maple functions.

```
> x -> x^2;
> x -> a*x^2 + b*x + c;
> (x,y) -> x^2 + y^2;
> z -> sin(z) + exp(z^2);
```

To the left of an arrow are variables that represent the input to a function. To the right of an arrow is an expression that represents the rule of the function. Notice that Maple functions are not really mathematical functions since there is no mention of a domain or a codomain. But Maple functions are clearly meant to define a rule showing how an output is computed from an input.

In the examples of Maple functions given above, we did not give any of them a name. Here is a Maple function named `f`.

```
> f := x -> x^2 - 1;
```

Notice that this command does two distinct things. It defines a Maple function and it assigns the function a name. When we define a Maple function using the arrow notation and give it a name at the same time using the assignment operator, we get a Maple command that can look quite strange to students, but they need to get very used to this notation.

Let us compare Maple's notation with the standard mathematical notation for defining and naming a function. The notation

$$\mathrm{f}(x) = x^2 - 1$$

defines a mathematical function named f that is equivalent to the Maple function named `f` defined by the Maple command

`f := x -> x^2 - 1`.

The Maple notation clearly separates the defining of the function (the arrow operation) from the naming of the function (the assignment operation) but the mathematical notation combines these two operations into one use of the equal sign. The mathematical notation is more compact but the mathematical notation has the disadvantage of being ambiguous. For example, suppose that the formula above is followed by the formula.

$$\mathrm{f}(x) = 3\,x^2 - x + 5$$

How should we interpret this second formula? Is it a redefinition of the function f, or is it a short hand formula for the following equation?

$$x^2 - 1 = 3\,x^2 - x + 5$$

There is really no way to tell from the second formula itself which interpretation is correct. The cause of this ambiguity is that in standard mathematical notation, the equal sign has (at least) two distinct uses, as either part of an assignment statement or as part of an equation.

Here are a few examples using the arrow operator that are purposely meant to be confusing for students. Examples like these are used in class discussions about Maple functions.

Here is an expression named `f` and a Maple function named `g` defined using `f`.

```
> f := x^2;
> g := x -> 2 * x^3 * f;
```

Here are two functions, `f` and `g`, where `g` is defined using `f`.

```
> f := x -> x^2;
> g := 2 * (x -> x^3) * f;
```

Here is another way to define `g` using the function `f`.

```
> g := x -> 2 * x^3 * f(x);
```
The last three definitions of **g** all define the same function. How does each of them make use of expressions, and the arrow notation?

Here is a Maple function named **g**.
```
> g := x -> (1 + exp(x))/x^3;
```
Here is another way to define the same function **g**.
```
> g := (1 + exp)/(x -> x^3);
```

Here is a Maple function named **f**.
```
> f := x -> (1 + x^2)/x^3;
```
Here is another way to define the same function **f**.
```
> f := (1 + (x -> x^2))/(x -> x^3);
```
Here is a third definition of the same function **f**.
```
> f := (1 + (z -> z^2))/(y -> y^3);
```

Here is a Maple function named **h**.
```
> h := 2*sin - 9/exp;
```
Notice that **h** really is a function, not an expression (so Maple functions do not always need to be defined using the arrow notation). Here is another way to define the same function **h**.
```
> h := x -> 2*sin(x) - 9/exp(x);
```

Here is an example of how Maple can force students to think carefully about what a variable is and what role a variable plays in the definition of a function. The following two lines define **f1** and **f2** to each be a name for an expression in one variable.
```
> f1 := x^2 + 1;
> f2 := y^2 + 1;
```
$$f1 := x^2 + 1$$
$$f2 := y^2 + 1$$
Add the two expressions and get an expression in the two variables **x** and **y**.
```
> f3 := f1 + f2;
```
$$f3 := x^2 + 2 + y^2$$
Define two Maple functions equivalent to the above expressions. Each of the two functions **g1** and **g2** is a function of one variable.
```
> g1 := x -> x^2+1;
> g2 := y -> y^2+1;
```
$$g1 := x \rightarrow x^2 + 1$$
$$g2 := y \rightarrow y^2 + 1$$
Add the two Maple functions. What do we get?
```
> g3 := g1 + g2;   # What kind of function is g3?
```
$$g3 := g1 + g2$$

Is `g3` a function of two variables like `f3` is an expression in two variables? Or is it a function of one variable? The following command shows the formula for `g3`.

```
> g3(x);
```

$$2\,x^2+2$$

So while `f1+f2` is an expression in two variables, `g1+g2` is a function of one variable. This is a very subtle and, for most students, very confusing aspect of using Maple. Helping students to build up their understanding of Maple to where they can understand this example will also help students to build up their general understanding of variables and functions.

2.2. Exercises

Here are some exercises from our Maple course that make use of the arrow operator. The goal of these exercises is both to get students used to Maple's arrow notation and to help students build their understanding of mathematical functions in general [3]. These exercises are absolutely crucial to this part of the course. Students will not develope an understanding of Maple functions without doing a number of them. Therefore it is important that these exercises be well designed so that they accomplish their goal. I hope that this collection of exercises will motivate some readers to try these problems with their students and to add new and better exercises to this list.

Exercise 1: Give a simplified definition for each of the following Maple functions.

```
> f := ((x,y)->x^2) + ((x,y)->y^2);
> g := ((x,y)->x^2) + ((y,x)->y^2);
```

Exercise 2: Explain what is different about the following two Maple functions.

```
> (u,v) -> 3*u+v^2;
> (s,t) -> 3*t+s^2;
```

Exercise 3: Notice that a Maple command like the following one defines a Maple function (in this case the squaring function) and then applies that function to an input (in this case `s`) and the result is an expression.

```
> (x->x^2)(s);
```

Carefully explain exactly why the following commands simplifiy to what they do.

```
> ((x,y)->x^2+y^2)(x,x)-((x,y)->x)(x^2,y^2);
> (((x,y)->x*y)(x,x))+(((u,v)->u+v)(3,y));
```

Exercise 4: Change a *single* character in the following command so that the Maple function defined is equivalent to the zero function.

```
> (x,y)->(((x,y)->x^2+y)(x,x)-((x,y)->x+x)(x^2,x));
```

Exercise 5: For each of the following Maple functions, give a simplified definition, comparable to what you would see in a calculus book. (Hint: Use the parentheses to help you figure out the structure of these definitions.)

```
> (sin @ exp / cos) + (z->z*ln(z)) - (x->x^2)(3);
> ((x,y)->x*exp(y))/((y,x)->x*exp(y))+((u,v)->u-v);
> (u->(((x,y)->exp(x)-y^2)(u,u))) + (u->u);
> ((x,y)->x^2)+((u,v)->v^2)-((s,t)->sin(s)*((cos*exp)(t)));
> (((x->x)-(x->x^2))*(1+exp)^(-1));
> sin+(x->x^2)^3-(y->y*sin(y))*sin;
> z->(((x,y)->x^3-exp(x*y))(z,0));
```

Exercise 6: Here is the definition of a Maple function `h`.

```
> h := exp*(y->exp(2*y))+(x->cos(2*x));
```

Give three other Maple commands that will each define a Maple function equivalent to the Maple function `h`. One of your commands should redefine `h` in as simple a way as possible. Find a way to show that your commands really do define Maple functions equivalent to `h`.

Exercise 7: Explain what the following command is doing.

```
> (D(z->3*z^2-2*x+z))(3);
```

Translate the above command into a single, equivalent Maple command that uses an anonymous expression instead of an anonymous function.

Exercise 8: Explain in detail how Maple evaluates the following expression. Use additional Maple commands to demonstrate the intermediate steps involved in evaluating this expression.

```
> D[2]((x,y)->(x^2+y^3)/(1+2*x*y))(0,x);
```

Exercise 9: Here are two ways of defining the same Maple function `g`.

```
> g := (x -> x) * sin;
> g := x -> x * sin(x);
```

Now here is a little puzzle. Consider the following command.

```
> g := x -> x * sin;
```

Does this command make sense?

Exercise 10: Explain in detail what the following two commands do and how they do it. How does each command make use of anonymous functions and/or expressions?

```
> plot( ((x,y)->x^3-y^3)(w,-1), w = -1..1 );
> plot( w->(((x,y)->x^3-y^3)(w,-1)), -1..1 );
```

Exercise 11: Explain what is wrong with the following `plot` command and then fix it by changing only one character in the command. (There are two ways to solve this problem.)

```
> plot( (z->z^2+1)(y), z=-3..3 );
```

Exercise 12: For each of the functions given below, describe what kind function it is, what kind of graph is most appropriate for the function, and use an appropriate Maple command to draw a graph for the function.

- (a) $(x \rightarrow \ln(x)) + (y \rightarrow \sin(y))$

- (b) $f(x, y) = (\ln(x) + \sin(y), \ln(x) + \sin(y), \ln(x) + \sin(y))$
- (c) $\ln + \sin$
- (d) $x \to (\ln(x), \sin(x))$
- (e) $\ln(x) + \sin(y)$
- (f) $(x, y) \to (\ln(x), \sin(y))$
- (g) $u \to (\ln(u), \sin(u), 0)$
- (h) $x \to \ln(y) + \sin(x)$

Exercise 13: The two formulas

$$z^2 + 3$$
$$f(x) = x^2 + 3$$

both represent the same mathematical function, the function that has a single input and adds 3 to the square of the input. Do the following two formulas represent the same mathematical function? Explain your answer.

$$s^2 + 3r + 1$$
$$f(x, y) = x^2 + 3y + 1$$

Exercise 14: How many different Maple functions of three variables can you define using the following expression on the right hand side of the arrow operator.

```
> a+exp(s)*cos(mu);
```

Exercise 15: The following four commands define two different functions. Which commands define the same function? Explain why.

```
> (x,y) -> cos(2*x)+y^2;
> (y,x) -> cos(2*x)+y^2;
> (y,x) -> cos(2*y)+x^2;
> (u,v) -> cos(2*v)+u^2;
```

Exercise 16: The following function definition is not correct and will produce an error message from Maple. Explain why Maple cannot allow a function definition like this one. Based on this example, what rule do you think is true about the definitions of Maple functions? (Hint: Using this definition, what would happen if you tried to evaluate the function `f` with some input?)

```
> f := (x,x)->x^2+x^3;
```

Exercise 17: Let $f(x, y) = 3x + 5y$ and let $h(z) = \sqrt{z + 1}$. Compute the composition h(f(x, y)) using both expressions and Maple functions. Explain why the composition f(h(z)) does not make sense mathematically. Compute the compositions f(h(x),y), f(x,h(y)), and f(h(x),h(y)). First do these compositions using expressions. Can you do these compositions using Maple functions and the `@` operator?

Exercise 18: In standard mathematical notation we use juxtaposition to denote multiplication, as in $2\,x$ to mean 2 multiplied with x, or $y\,(x+1)$ to mean y times the quantity $x+1$, or $(u+v)\,(x-y)$ to mean the quantity $u+v$ times the quantity $x-y$. Maple notation does not allow this simple and common symbolism for multiplication. Explain how Maple interprets the following two commands.

```
> y(x+1);
> (u+v)(x-y);
```

Exercise 19: Translate the following sequence of formulas into a reasonable sequence of Maple commands that mimic the meaning of the formulas as closely as possible.

$$\mathrm{f}(x) = c\,x^2 - 2\,x + 1$$

$$\mathrm{f}(x) = 0 \qquad (1)$$

$$\text{Solve (1) with } c = 3$$

2.3. Recursive and higher order functions

3. Parameterizations as objects to think with

The module on parametric curves and surfaces demonstrates the kinds of Maple problem solving skills that we want students to develop. The module is a series of examples and exercises that ask students to play with the familiar parameterizations of the circle, sphere, and torus. First, students are asked to replace the trigonometric functions in the parameterization of the circle with piecewise linear functions that parameterize a square and a triangle. Then they are asked to find periodic extensions of their new component functions in order to parameterize triangular and square "spirals". Switching to surfaces, the students are asked to use the components from the square and triangle parameterizations to replace some of the trig functions in the parameterization of the sphere in order to parameterize a closed cylinder, a cube, a closed cone, a pyramid, and a tetrahedron. This requires that students really understand the role played by every trigonometric function in the parameterization of the sphere and it leads to a deeper understanding of how these parameterizations work. The students are given similar challenges with the torus parameterization. All the parameterizations are then animated in order to better visualize how each parameterization depends on each of its variables. Finally we create homotopies between the cosine function and the piecewise linear functions that parameterize the square and triangle. This leads to a variety of animations like a sphere morphing into a cube or a tetrahedron.

3.1. Parametric curves

We want a series of examples and exercises that help students to understand parametric curves. What we present here is a series of examples using parametric curves, but for our Maple course many of these examples would be converted into exercises and/or projects.

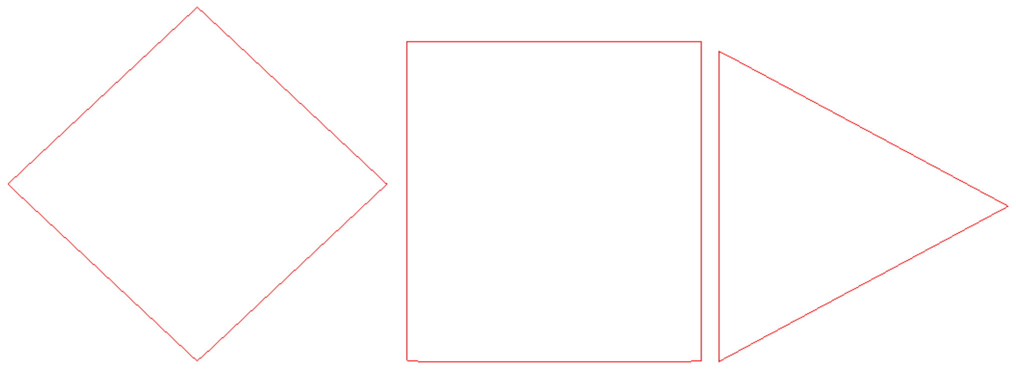
3.1.1. Parameterizing squares and triangles

Consider the standard parameterization of the circle, which is usually not well understood by calculus students.

```
> [cos(theta), sin(theta), theta=0..2*Pi];
> plot( %, scaling=constrained, axes=none );
```

Very few students are likely to "discover" this parameterization based on the idea of what a parameterization does.

Let us replace the circle with some simpler geometric figures for which it is more intuitive what the parameterizing functions are. Each of these three figures can be parameterized using piecewise linear component functions.



Let us parameterize each figure in one unit of time. For the two squares, each side will take one fourth of a unit of time to parameterize. For the triangle, each side will take one third of a unit of time. As in the circle parameterization, we begin each parameterization at time $t = 0$ at the point (1,0) and parameterize each figure going counterclockwise in positive time.

For the first square, the horizontal component of the parameterization needs to decrease from 1 to 0 in the first 1/4 unit of time and then it needs to decrease from 0 to -1 in the next 1/4 unit of time (which is the same as decreasing from 1 to -1 in the first 1/2 unit of time). Similarly for the second 1/2 unit of time.

```
> sq1_horz := t -> piecewise( t <= 1/2, 1-4*t,
>                             t <= 1,   4*t-3 );
```

The vertical component needs to increase from 0 to 1 in the first 1/4 unit of time, decrease from 1 to -1 in the next 1/2 unit of time, and then increase from -1 to 0 in the last 1/4 unit of time.

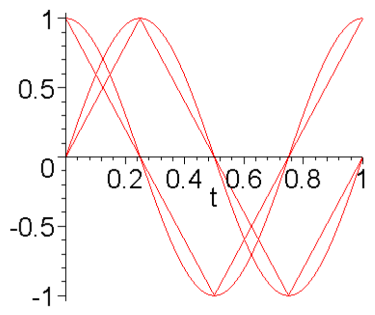
```
> sq1_vert := t -> piecewise( t <= 1/4, 4*t,
>                             t <= 3/4, 2-4*t,
>                             t <= 1,   4*t-4 );
```

Here are the component functions.

```
> plot( [sq1_horz(t), sq1_vert(t)], t=0..1, color=red );
```

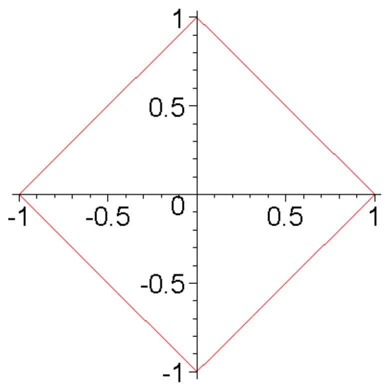
Notice that these component functions are vaguely cosine and sine like.

```
> plot( [sq1_horz(t), sq1_vert(t), cos(2*Pi*t), sin(2*Pi*t)],
  t=0..1, color=red );
```



Here is the parameterized square.

```
> plot( [sq1_horz(t), sq1_vert(t), t=0..1] );
```



Here is an animation of how the horizontal and vertical components work together to create the parametric square.

```
> x := t -> sq1_horz(t):  # horizontal component function
> y := t -> sq1_vert(t):  # vertical component function
> curves := array( 1..2, 1..2 ):
> curves[1,1] := plots[animatecurve]( [ x(t), t, t=0..1 ],
>                  xtickmarks=[-1,1], ytickmarks=[0,1/4,1/2,3/4,1],
>                  title="horizontal component", frames=50 ):
> curves[2,2] := plots[animatecurve]( y(t), t=0..1,
>                  xtickmarks=[0,1/4,1/2,3/4,1], ytickmarks=[-1,1],
>                  title="vertical component", frames=50 ):
> curves[2,1] := plots[animatecurve]( [ x(t), y(t), t=0..1 ],
>                  xtickmarks=[-1,1], ytickmarks=[-1,1], frames=50
  ):
> curves[1,2] := plot([[0,0]],axes=none): # create an empty graph
> plots[display]( curves );
> x,y := 'x','y':  # unassign x and y
```

For the next square, since we are beginning the parameterization when time $t = 0$ at the point (1,0), the horizontal component stays constant with value 1 for the first 1/8 unit of time, then it decreases from 1 to -1 in the next 1/4 unit of time, stays constantly equal to -1 for a 1/4 unit of time, then increases to 0 in 1/8 of a unit of time.

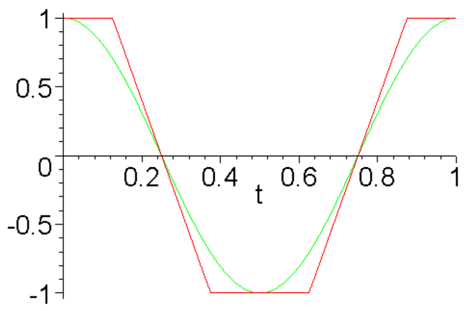
```
> sq2_horz := t -> piecewise( t <= 1/8,  1,
>                             t <= 3/8, -8*t+2,
>                             t <= 5/8, -1,
>                             t <= 7/8,  8*t-6,
>                             t <= 1,    1 );
```

Similarly for the vertical component.

```
> sq2_vert := t -> piecewise( t <= 1/8,  8*t,
>                             t <= 3/8,  1,
>                             t <= 5/8, -8*t+4,
>                             t <= 7/8, -1,
>                             t <= 1,    8*t-8 );
```

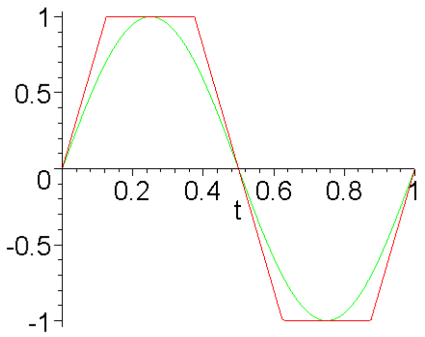
Compare the component functions with cosine and sine. First the horizontal component.

```
> plot( [sq2_horz(t), cos(2*Pi*t)], t=0..1 );
```



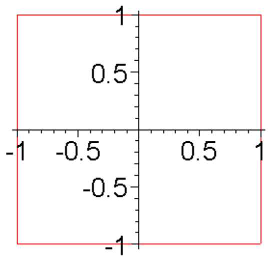
And the vertical component.

```
> plot( [sq2_vert(t), sin(2*Pi*t)], t=0..1 );
```



Here is the parameterized square.

```
> plot( [sq2_horz(t), sq2_vert(t), t=0..1] );
```



+ Animate the parameterization

Now let us turn to the triangle. Here is the horizontal component for a parameterization of the triangle.

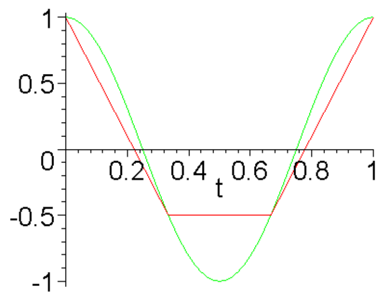
```
> tri_horz := t -> piecewise( t < 1/3, (-9/2)*t+1,
>                             t < 2/3, cos(2*Pi/3), # a constant
>                             t <= 1,  (9/2)*t-7/2 );
```

Here is the vertical component of the parameterization.

```
> tri_vert := t -> piecewise( t < 1/3,  3^(3/2)/2*t,
>                             t < 2/3, -3^(3/2)*(t-1/2),
>                             t <= 1,   3^(3/2)/2*(t-1) );
```

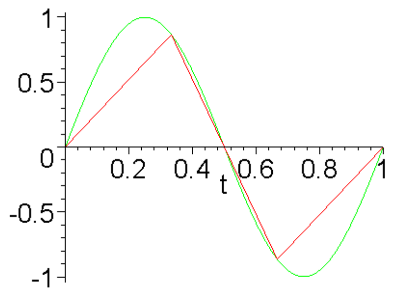
Compare the component functions with cosine and sine. Notice that these component functions are not as much like the trig functions as with the other two figures (this will be important later on). First the horizontal component.

```
> plot( [tri_horz(t), cos(2*Pi*t)], t=0..1 );
```



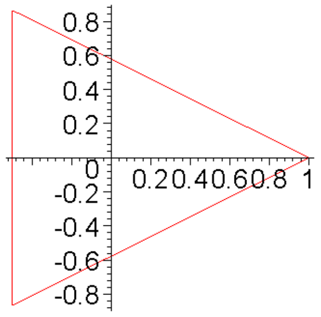
And the vertical component.

```
> plot( [tri_vert(t), sin(2*Pi*t)], t=0..1 );
```



Here is the parameterized triangle.

```
> plot( [ tri_horz(t), tri_vert(t), t=0..1 ] );
```



Annimate the parameterization

Now let us consider a few ways of modifying these examples. Since the component functions of the circle parameterization are periodic, we can easily parameterize spirals and helixes.

```
> [theta*cos(theta), theta*sin(theta), theta=0..6*Pi];
> plot( %, scaling=constrained, axes=none );
```

A helix in 3-dimensional space.

```
> [cos(theta), sin(theta), theta];
> plots[spacecurve]( %, theta=0..6*Pi );
```

Can we generate "spirals" from our other parameterizations? To do so we need to define periodic extensions of our component functions.

For each horizontal component we need to create an even periodic extension with period 1. Here is one way to define an even periodic extension for the horizontal component of the first square parameterization. The **frac** function gives us periodicity with period 1 and the **abs** function makes the periodic function even.

```
> sq1_horz := t -> piecewise( frac(abs(t)) <= 1/2, 1-4*frac(abs(t)),
>                             frac(abs(t)) <= 1,   4*frac(abs(t))-3
  );
```

Compare the periodic extension to cosine.

```
> plot( [sq1_horz(t), cos(2*Pi*t)], t=-1..2 );
```

Each vertical component needs an odd periodic extension (students should to reflect on why this is true). As before, the **frac** function gives us periodicity with period 1, and the **abs** and **signum** functions combine to make the periodic function odd.

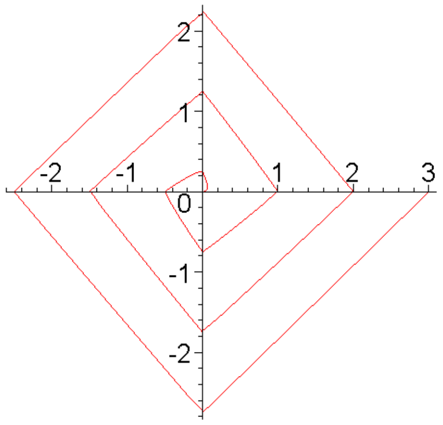
```
> sq1_vert := t -> piecewise( frac(abs(t)) <= 1/4,
>                                signum(t)*(4*frac(abs(t))),
>                             frac(abs(t)) <= 3/4,
>                                signum(t)*(2-4*frac(abs(t))),
>                             frac(abs(t)) <= 1,
>                                signum(t)*(4*frac(abs(t))-4) );
```

Compare the periodic extension to sine.

```
> plot( [sq1_vert(t), sin(2*Pi*t)], t=-1..2 );
```

Parameterize a square "spiral".

```
> plot( [t*sq1_horz(t), t*sq1_vert(t), t=0..3] );
```



A square "helix".

```
> plots[spacecurve]( [sq1_horz(t), sq1_vert(t), t], t=0..6,
  numpoints=200 );
```

Another version of the helix.

```
> plots[tubeplot]([sq1_horz(t), sq1_vert(t), t], t=0..6,
  radius=0.1);
```

Here is a way to define the odd periodic extension of the vertical component function using the `floor` function (instead of using `frac`, `abs`, and `signum`).

```
> sq1_vert := t -> piecewise( t-floor(t) <= 1/4, 4*(t-floor(t)),
>                             t-floor(t) <= 3/4, 2-4*(t-floor(t)),
>                             t-floor(t) <= 1,   4*(t-floor(t))-4 );
```

Parameterize the square spiral using this new vertical component fuction.

```
> plot( [t*sq1_horz(t), t*sq1_vert(t), t=0..3] );
```

The other square parameterization and the triangle parameterization can both be made into periodic parameterizations in exactly the same way. However, let us make the other square parameterization periodic in a different way. This other way introduces the idea of a recursively defined function. But before doing that, let us make an observation. The sine and cosine functions are just translations of each other. Since our horizontal and vertical component functions look quite a bit like sine and cosine, they also should be translations of each other. Here is a proof of this. We can parameterize the square using just our horizontal component function (the one that looks like cosine). The vertical component function is now written as a translation of the horizontal component function.

```
> plot( [sq1_horz(t), sq1_horz(t-1/4), t=0..1] );
> plot( [t*sq1_horz(t), t*sq1_horz(t-1/4), t=0..3] );
```

This observation means that we only need to define a periodic extension for the horizontal component of the second square. Now let us make another observation. The figure of a square is very symmetric. If we can parameterize the part of the square in the first quadrant, then we should be able to use that to parameterize the rest of the square. Here is an example of this idea using the cosine function. One quarter period of the cosine function defines all of the rest of the cosine function.

```
> plot( cos(x), x=0..Pi/2 );
```

Here is how to define a "new" cosine function between 0 and 2 π using only the first quarter period of the builtin cosine function between 0 and $\pi/2$. Notice that this definition is recursive (this helps demonstrate how closely related are symmetry and recursion).

```
> my_cos := t -> piecewise(
>                  t>=0 and t<Pi/2,    cos(t),          # first quarter
>                  t>=Pi/2 and t<Pi, -my_cos(Pi-t),   # second quarter
>                  t>=Pi and t<2*Pi,   my_cos(2*Pi-t) # second half
>                 );
```

The next graph shows that `my_cos` defines the correct function for inputs between 0 and 2 π.

```
> plot( my_cos, 0..2*Pi );
```

Now redefine `my_cos` so that it is periodic for all real numbers. We need to define `my_cos` for $t < 0$

and for $2\pi < t$.

```
> my_cos := t -> piecewise( t<0,       my_cos(t+2*Pi),
>                           t<=Pi/2,   cos(t),
>                           t<=Pi,    -my_cos(Pi-t),
>                           t<=2*Pi,   my_cos(2*Pi-t),
>                           t>2*Pi,    my_cos(t-2*Pi) );
> plot( my_cos, -2*Pi..4*Pi );
```

Let us use the same idea to create an even periodic extension of the horizontal component from the second square. The first line of the following **piecewise** command uses recursion to extend the function periodically over the negative numbers. The next two lines define the first quater period of the function. The next line uses recursion to extend the definition to the second quarter period. The fourth line uses recursion to extend the definition to the second half of the unit interval. And the last line uses recursion to extend the definition periodically over the rest of the positive numbers.

```
> sq2_horz := t -> piecewise(
>                  t<0,     sq2_horz(t+1),    # extend to neg
  numbers
>                  t<=1/8,  1,                # define the first
>                  t<=1/4, -8*t+2,            # quarter period
>                  t<=1/2, -sq2_horz(1/2-t),  # second quarter
  period
>                  t<=1,    sq2_horz(1-t),    # second half period
>                  t>1,     sq2_horz(t-1)     # extend to numbers >
  1
>                 );
```

Here is the periodic extension.

```
> plot( sq2_horz, -1..3 );
```

Now we can use this component function and its translate to parameterize a square or a spiral.

```
> plot( [sq2_horz, sq2_horz@(t->t-1/4), 0..1] );
> plot( [(t->t)*sq2_horz, (t->t)*sq2_horz@(t->t-1/4), 0..3] );
```

Here is a parameterization of a regular octagon. Notice in this example how powerful the **piecewise** command can be. We are doing real programming here. We are doing type checking on the argument to our function and our function is recursively defined in a name independent way.

```
> oct := t -> piecewise( not type(t, numeric), 'procname(args)',
>                        t < 0,    procname(t+1),
>                        t > 1,    procname(t-1),
>                        t < 1/16, 1,
>                        t < 3/16, (-16+8*sqrt(2))*t-1/sqrt(2)+2,
>                        t <= 1/4,
  (-16*(sqrt(2)-1))*t+4*(sqrt(2)-1),
>                        t <= 1/2, -procname(1/2-t),
>                        t <= 1,   procname(1-t)
>                       );
> plot( oct, 0..2 );
```

Now draw a regular octagon and an "octagonal spiral".

```
> plot( [oct(t), oct(t-1/4), t=0..1], scaling=constrained );
> plot( [t*oct(t), t*oct(t-1/4), t=0..3], scaling=constrained );
```

3.1.2. Parameterizing a triangle with a single function

3.1.3. More examples

3.2. Parametric surfaces

We want a series of examples and exercises that help students to understand parametric surfaces. What we present here is a series of examples using parametric surfaces that build on the examples of parametric curves from the last section. For our Maple course, many of these examples would be converted into exercises and/or projects.

3.2.1. Plugging into the parameterization of a sphere

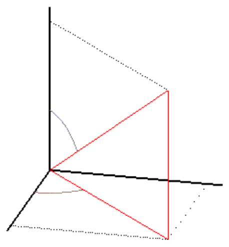
Consider the standard parameterization of the unit sphere. This is a common example of a parametric surface in most calculus books and it is usually not well understood by calculus students.

```
> [sin(phi)*cos(theta), sin(phi)*sin(theta), cos(phi)];
> plot3d( %, theta=0..2*Pi, phi=0..Pi, scaling=constrained,
  axes=none );
```

This parameterization is usually derived from the fact that the equation in spherical coordinates for the unit sphere is $\rho = 1$ and the transformation from spherical coordinates to rectangular coordinates is given by the three equations

$$x = \rho \sin(\phi)\cos(\theta), \qquad y = \rho \sin(\phi)\sin(\theta), \qquad \text{and} \qquad z = \rho \cos(\phi).$$

So remembering the parameterization for a sphere requires remembering the transformation from spherical to rectangular coordinates, and this transformation is usually explained with a picture something like this.



This is neither an easy way to remember the parameterization for a sphere nor is it not a good way to understand the parameterization of a sphere, and it does little to help students understand parametric surfaces in general.

Let us look at the parameterization of a sphere from a different point of view. Horizontal cross sections of a sphere are circles. So we can think of a sphere as a stack of circles whose radii vary with their height above the *xy*-plane. Now look again at the parameterization of the unit sphere.

$$x = \sin(\phi)\cos(\theta), \qquad y = \sin(\phi)\sin(\theta), \qquad z = \cos(\phi).$$

Notice that for a fixed value of ϕ, the x and y coordinates parameterize a horizontal circle of radius $\sin(\phi)$ that has height $\cos(\phi)$ above the *xy*-plane. So we see right away that this parameterization does describe the sphere as a stack of horizontal circles. The height above the *xy*-plane of a circle is given by the $\cos(\phi)$ term which starts at 1 and decreases to -1 as ϕ goes from 0 to π. The radius term $\sin(\phi)$ starts out at 0 when ϕ is 0 and the radius then grows to 1 and then shrinks back to 0 as ϕ goes from 0 to π. The way the radii of these circles change with their height is also described by the profile of the sphere in a vertical plane. If we fix $\theta = 0$, then $y = 0$ and we see that the profile of the sphere in the vertical *xz*-plane is described by $x = \sin(\phi)$ and $z = \cos(\phi)$, which is clearly a circular profile (similarly for fixed $\theta = \frac{\pi}{2}$ and the *yz*-plane).

In summary, we can say that the $\cos(\theta)$ and $\sin(\theta)$ terms in the paramterization of the sphere define the horizontal cross sections of the surface, and the $\sin(\phi)$ and $\cos(\phi)$ terms define the vertical profile of the surface. Now let us see how we can use this interpretation of the parameterization to create parameterizations of some new surfaces.

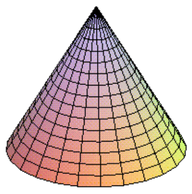
First, let us change the vertical profile of the surface to be a triangle. We can use our parameterization of a triangle from the last section. Here are the horizontal and vertical components for the parameterization of a triangle.

```
> tri_horz := t -> piecewise( t < 1/3, (-9/2)*t+1,
>                             t < 2/3, cos(2*Pi/3),
>                             t <= 1,  (9/2)*t-7/2 ):
> tri_vert := t -> piecewise( t < 1/3,  3^(3/2)/2*t,
>                             t < 2/3, -3^(3/2)*(t-1/2),
>                             t <= 1,   3^(3/2)/2*(t-1) ):
```

Let us "plug in" these component function in place of the $\sin(\phi)$, $\cos(\phi)$ terms from the paramterization of the sphere. The surface will have circular horizontal cross sections and a triangular profile. (Notice that the range for ϕ is changed to 0 to 1/2, instead of 0 to π).

```
> '[tri_vert(phi)*cos(theta), tri_vert(phi)*sin(theta),
  tri_horz(phi)]';
> plot3d( %, theta=0..2*Pi, phi=0..1/2, scaling=constrained,
  axes=none );
```

$$[\mathrm{tri\_vert}(\phi)\cos(\theta), \mathrm{tri\_vert}(\phi)\sin(\theta), \mathrm{tri\_horz}(\phi)]$$



Question: How would you remove the bottom from this surface?

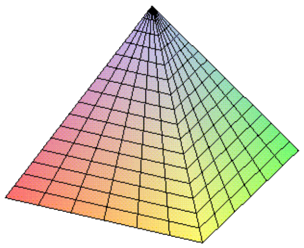
Now let us replace the circular cross sections with square cross sections. We need to replace the cos(θ), sin(θ) terms with the components of our parameterization of a square from the last section.

```
> sq2_horz := t -> piecewise( frac(abs(t)) < 1/8, 1,
>                             frac(abs(t)) < 3/8, -8*frac(abs(t))+2,
>                             frac(abs(t)) < 5/8, -1,
>                             frac(abs(t)) < 7/8, 8*frac(abs(t))-6,
>                             frac(abs(t)) < 1,   1 ):
```

Here is a triangular profile with square cross sections.

```
> '[tri_vert(phi)*sq2_horz(theta),
  tri_vert(phi)*sq2_horz(theta-1/4), tri_horz(phi)]';
> plot3d( %, theta=0..1, phi=0..1/2, scaling=constrained, axes=none
  );
```

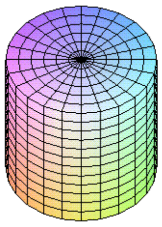
$$\left[\mathrm{tri\_vert}(\phi)\,\mathrm{sq2\_horz}(\theta),\ \mathrm{tri\_vert}(\phi)\,\mathrm{sq2\_horz}\left(\theta-\frac{1}{4}\right),\ \mathrm{tri\_horz}(\phi)\right]$$



Here is a square profile with circular cross sections.

```
> '[sq2_horz(phi-1/4)*cos(theta), sq2_horz(phi-1/4)*sin(theta),
  sq2_horz(phi)]';
> plot3d( %, theta=0..2*Pi, phi=0..1/2, scaling=constrained,
  axes=none );
```

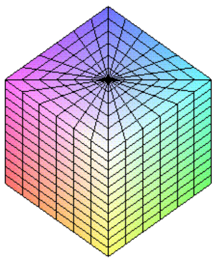
$$\left[\mathrm{sq2\_horz}\left(\phi-\frac{1}{4}\right)\cos(\theta),\ \mathrm{sq2\_horz}\left(\phi-\frac{1}{4}\right)\sin(\theta),\ \mathrm{sq2\_horz}(\phi)\right]$$



A square profile with square cross sections.

```
> '[sq2_horz(phi-1/4)*sq2_horz(theta),
>   sq2_horz(phi-1/4)*sq2_horz(theta-1/4),
>   sq2_horz(phi)]';
> plot3d(%, theta=0..1, phi=0..1/2, scaling=constrained, axes=none);
```

$$\left[\mathrm{sq2\_horz}\left(\phi-\frac{1}{4}\right)\mathrm{sq2\_horz}(\theta),\ \mathrm{sq2\_horz}\left(\phi-\frac{1}{4}\right)\mathrm{sq2\_horz}\left(\theta-\frac{1}{4}\right),\ \mathrm{sq2\_horz}(\phi)\right]$$

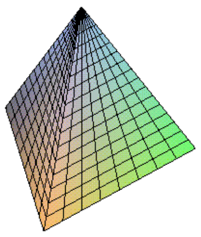


Exercise: Draw the cube with its top and bottom faces removed. Can you remove any other face?

Triangular profile with triangular cross sections.

```
> '[tri_vert(phi)*tri_horz(theta), tri_vert(phi)*tri_vert(theta),
  tri_horz(phi)]';
> plot3d(%, theta=0..1, phi=0..1/2, scaling=constrained, axes=none);
```

$$[\mathrm{tri\_vert}(\phi)\ \mathrm{tri\_horz}(\theta),\ \mathrm{tri\_vert}(\phi)\ \mathrm{tri\_vert}(\theta),\ \mathrm{tri\_horz}(\phi)]$$



An odd mix of component functions from three different parametric curves (circle, square, and triangle).

```
> '[sin(2*Pi*phi)*cos(2*Pi*theta),
>   sin(2*Pi*phi)*sq1_vert(theta), # vert component from 1st square
>   tri_horz(phi)]';
> plot3d( %, theta=0..1, phi=0..1/2, scaling=constrained, axes=none
  );
```

The following section contains some examples of the kinds of exercises that can be based on the above examples. In our Maple course, many of the above example are themselves exercises or projects that students work on.

3.2.2. Exercises

Here are some examples of exercises that can be given to students who are familiar with the above material.

Exercise 1: Part (a) Here are four different parameterizations of the sphere of radius one centered at the origin. For each parameterization try to change just a single number from one range so that the parameterization draws only the upper hemisphere. If you cannot do this by changing just one number in one range, explain why.

```
> [ sin(phi)*cos(theta), sin(phi)*sin(theta), cos(phi) ];
> plot3d( %, theta=0..2*Pi, phi=0..Pi, title="Sphere" );
```

```
> [ cos(phi)*cos(theta), cos(phi)*sin(theta), sin(phi) ];
> plot3d( %, theta=0..Pi, phi=0..2*Pi, title="Sphere" );
```

```
> [ sin(phi)*cos(theta), sin(phi)*sin(theta), cos(phi) ];
> plot3d( %, theta=0..Pi, phi=0..2*Pi, title="Sphere" );
```

```
> [ sqrt(1-z^2)*cos(theta), sqrt(1-z^2)*sin(theta), z ];
> plot3d( %, theta=0..2*Pi, z=-1..1, title="Sphere" );
```

Part (b) Repeat part (a) but this time try to draw only the right hemisphere.

Part (c) Repeat part (a) but this time try to draw 3/4 of the sphere (say, the upper hemisphere and the left or right half of the lower hemisphere).

Exercise 2: Explain why the following parameterization produces the shape that it does.

```
> [ cos(phi)*cos(theta), cos(phi)*sin(theta), cos(phi) ];
> plot3d( %, theta=0..2*Pi, phi=0..Pi,
>         title="Modified sphere parameterization" );
```

Exercise 3: The following surface has a square profile when you look down one axis and a circular profile when you look down another axis. Derive these profiles from the parameterization.

```
> [ sin(phi)*cos(theta), sin(theta), cos(phi) ];
> plot3d( %, theta=0..2*Pi, phi=0..Pi, scaling=constrained );
```

Here is a slight variation on this surface.

```
> [ sin(phi)*cos(theta)/2, sin(theta), cos(phi) ];
> plot3d( %, theta=0..2*Pi, phi=0..Pi, scaling=constrained,
>            orientation=[35,65], title="Pillow" );
```

Exercise 4: The following surface has a circular profile when you look straight down one coordinate axis, it has a cone shaped profile when you look straight down another coordinate axis, and it has a square profile when you look straight down the third coordinate axis. Use the parameterization to explain these profiles.

```
> [ cos(phi)*cos(theta), sin(phi)*sin(theta), cos(phi) ];
> plot3d( %, theta=0..2*Pi, phi=0..Pi,
>         title="Modified sphere parameterization" );
```

Exercise 5: Here is a parameterization of a closed frustum of a cone that has radius 1 at the top and radius 2 at the bottom and height 1 (the height is the vertical distance from the top surface to the bottom surface). Modify this parameterization so that it uses three parameters, `a`, `b`, and `c`, such that `a` and `b` determine the radius of the top and bottom of the frustum respectively and `c` is the height.

```
> f := x -> piecewise( x<1/4, 4*x, x<3/4, 2*x+1/2, x<=1, -8*x+8 );
> g := x -> piecewise( x<1/4, 1, x<3/4, 3/2-2*x, x<=1, 0 );
> [ f(phi)*cos(theta), f(phi)*sin(theta), g(phi) ]:
> plot3d( %, theta=0..2*Pi, phi=0..1, title="Frustum of a Cone" );
```

+ 3.2.3. Plugging into the parameterization of a torus

+ 3.2.4. More animations

+ 3.2.5. Morphing and homtopies

4. References

[1] Drijvers, Paul, *Students encountering obstacles using a CAS*, International Journal of Computers for Mathematical Learning, **5** (2000), 189-209.

[2] Fey, James, *Computer Algebra Systems in Secondary School Mathematics Education*, National Council of Teachers of Mathematics, 2003.

[3] Harel, Guershon, and Dubinsky, Ed, editors, *The Concept of Function: Aspects of Epistemology and Pedagogy*, Mathematical Association of America, 1992.

[4] Wells, Charles, *Communicating Mathematics: Useful Ideas from Computer Science*, American Mathematical Monthly, **102** (1995), no. 5, 397-408.