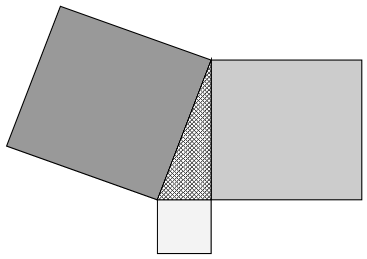
Experimental Math: one of the top 75 Byrne Seminars in the fall 2008 semester!

Searching for solutions to equations with `Maple`

I am happy to declare that the material in this discussion was inspired by ideas of Eric Rowland and will be presented by him. I thank him very much.

Geometry

Take a right triangle, *any* right triangle. Put squares on each of the sides. The area of the square on the side opposite the right triangle (that side is called the *hypotenuse*) is equal to the sums of the areas of the squares on the other two sides (called the *legs*). This is a fascinating fact, and has apparently been known to human beings for *thousands and thousands* of years (according to some sources, at least 4500 years*).



In many cultures, right angles were "constructed" and used in building and surveying with the use of specific triangles: 3–4–5 right triangles ($3^2 + 4^2 = 5^2$) and certainly 5–12–13 triangles are the triangles with shortest sides. These triangle lengths and many others certainly were known 2500 years ago**. How can we find *Pythagorean triples*, that is, three integers a, b, and c which satisfy the equation $a^2 + b^2 = c^2$?

Initial attempts

We will create some `Maple procedures` to find Pythagorean triples. Although our methods will not be elaborate, the procedures will be simpler if we use some commands that haven't been discussed yet. Here is an initial procedure mostly to show you a command you may not have seen before.

```
squares := proc(n)
local x;
    for x to n do if type(sqrt(x), integer) then print(x) end if end do
end proc;
```

You may enter this at the `Maple` screen. I have followed Mr. Rowland's suggestion and presented the procedures here with line breaks and indenting which are *supposed* to help human beings understand them. Internally `Maple` doesn't care about presentation that much. If you make typing errors or have other problems, `Maple` will gently call your attention to them. You can fix each one until what remains is a gem of computer directions. What's new here? Well, the line *local x* tells `Maple` that there's a variable in the procedure which is referred to inside the procedure and doesn't have any meaning outside it. Also, we introduced the command `type` which has many possible uses. For example, consider `type(Irving,cat)`. This command returns `true` of `Irving` *is* a cat, and returns `false` if `Irving` is *not* a cat. Here "cat" stands for one of the types which is recognized by `Maple`. There are almost 500 different types recognized by `Maple`. You could find out about them by typing the command *help(type);*. You'll see a long list. Here we want to check if `sqrt(x)`, the square root of x, is an `integer`, a type `Maple` recognizes. Some examples of the use of `type` follow.

* I'll put links to some historical discussions on the course web page for this meeting.

** Smart human beings have been around quite a while.

```
> type(1/3,integer);
                              false
> type(sqrt(2),integer);
                              false
> type(56^3+338001,integer);
                              true
```

So this should give you some idea what `type` does. This is the the result of one use of `squares`:

```
> squares(10);
                              1
                              4
                              9
```

This is satisfactory but hardly exciting. There are easier ways of printing the first 3 squares (`seq(j^2,j=1..3);` for example).

Here is an attempt to find Pythagorean triples:

```
pythag := proc(n)
local x, y;
   for x from 1 to n do for y from 1 to n do
         if type(sqrt(x^2 + y^2), integer) then
            print(x, y, sqrt(x^2 + y^2))
         end if
      end do
   end do
end proc;
```

I needed three attempts to type this correctly, matching parentheses and end's. `pythag` checks square roots of sums of squares of pairs of* integers from 1 to n, and prints out if the result is an integer. This is a simple way to get Pythagorean triples. The instruction `for x from 1 to n do for y from 1 to n do` makes the variables x and y run each from 1 to n. They `do` the instructions which follow until the corresponding `end do` is reached.
One linguistic (?) element should be mentioned. When `Maple` echoes the line

```
for x from 1 to n do for y from 1 to n do
```

you will actually see the following instead

```
for x to n do for y to n do
```

The designers of `Maple` decided that `for` "loops" which begin with 1 are so common and that `from 1` doesn't need to be shown. This can be unexpected to the inexperienced.
Here is a use of this `procedure`:

```
> pythag(10);
                              3, 4, 5
                              4, 3, 5
                              6, 8, 10
                              8, 6, 10
```

* Four of's in a row!

There seems to be some duplication. Let's improve `pythag`.

Improvements

We would like to make our procedure give more satisfactory results. At first we would like to get rid of duplications. We could consider `3, 4, 5` and `4, 3, 5` to be the same result. What we are doing now is customary. It is both irritating and very common. First, get something that works, and then tweak it to improve it (but always check that the "improvements" do not destroy the functionality!). Of course, if you can get a *better* basic idea, that's neat. But small, incremental improvements help.

```
pythagB := proc(n)
local x, y;
    for x to n do for y from x to n do
            if type(sqrt(x^2 + y^2), integer) then
                print(x, y, sqrt(x^2 + y^2))
            end if
        end do
    end do
end proc;
```

Look closely at the `do` instruction. x still runs from 1 to n but now y goes from x to n only. Therefore `4, 3, 5` should not appear, but `3, 4, 5` should. Indeed, here are results:

```
> pythagB(10);
                        3, 4, 5
                        6, 8, 10
```

and even

```
> pythagB(20);
                        3, 4, 5
                        5, 12, 13
                        6, 8, 10
                        8, 15, 17
                        9, 12, 15
                        12, 16, 20
                        15, 20, 25
```

This is very nice, since we have displayed the most familiar Pythagorean triples (the first two answers above). We could be even more critical, and observe that `3, 4, 5` and `6, 8, 10` are really sort of the same. That is, the second triple is just twice the first triple, and really the novelty is the first triple, and the second one just echoes the first. Actually, there would be lots of other such triples (9, 12, and 15, and 300, 400, and 500). Let's try to eliminate the triples with a common factor.

We can compute the *greatest common divisor* of some integers with the command `igcd` (**i**nteger **g**reatest **c**ommon **d**ivisor). Look:

```
> igcd(8,27);
                        1
> igcd(9,27);
                        9
```

```
> igcd(frog,toad);
                    igcd(frog, toad)
```

The last command was my effort to introduce humor. `Maple` grimly declares that it doesn't know `frog` and `toad` and can't compute the quantity desired. This is indicated by returning what I asked with no change at all.
Here is an *improved* version of our search:

```
pythagC := proc(n)
local x, y;
    for x to n do for y from x to n do
            if type(sqrt(x^2 + y^2), integer) and igcd(x, y) = 1 then
                print(x, y, sqrt(x^2 + y^2))
            end if
        end do
    end do
end proc;
```

This is neat because:

```
> pythagC(20);
                         3, 4, 5
                         5, 12, 13
                         8, 15, 17
```

Pythagorean triples with no common factors are called "primitive Pythagorean triples" and our procedure `pythagC` produces a list of these triples in the range specified.

Other equations?
We can experiment and alter the procedures we already have to investigate other equations. Consider the equation $a^2 + b^2 = 2c^2$. Can we find a list of (primitive!) solutions to this equation? if we want to know when a^2+b^2 is equal to 2 multiplied by a square, we consider `type(sqrt((a^2+b^2)/2,integer)`. This divides $a^2 + b^2$ by 2 and detects if square root of the result is an integer. So we can create another procedure.

```
pythag2 := proc(n)
local x, y;
    for x to n do for y from x to n do
            if type(sqrt(1/2*x^2 + 1/2*y^2), integer) and igcd(x, y) = 1
            then print(x, y, sqrt(1/2*x^2 + 1/2*y^2))
            end if
        end do
    end do
end proc;
```

In all this I am copying the spacing that `Maple` uses when it "prettyprints" the results. One crazy thing that `Maple` also does (I don't know why!) is that when the instruction *type(sqrt((x^2 + y^2)/2, integer)* is entered, then *type(sqrt(1/2*x^2 + 1/2*y^2), integer)* is printed. This is logically the same, but I don't understand why the change is made.
Here is a result:

```
> pythag2(30);
                        1, 1, 1
                        1, 7, 5
                       7, 17, 13
                       7, 23, 17
```

This is very neat. The machine works for us.

Just one more ...
How about $a^2 + b^2 = 3c^2$? A small alteration in `pythag2` will work. We're only making small changes at each step!

```
pythag3 := proc(n)
local x, y;
    for x to n do for y from x to n do
            if type(sqrt(1/3*x^2 + 1/3*y^2), integer) and igcd(x, y) = 1
            then print(x, y, sqrt(1/3*x^2 + 1/3*y^2))
            end if
        end do
    end do
end proc;
```

And here is some output:

```
> pythag3(10);
```

That's the result, **NOTHING**. In fact, here is more output:

```
> pythag3(1000);
```

The result, also **NOTHING**. Oh well, I should also mention that *pythag3(10)* took only .01 seconds (one one-hundredth of a second) on my home computer while *pythag3(1000)* took about 20 seconds. Maybe this sort of searching is not so good. Maybe there are/are not solutions to this equation. What does this *experiment* make you think? Can some argument be made?

And now something rather new ...
Leonhard Euler (1707–1783) was one of the very greatest mathematicians in European history, a very clever, hardworking person. He conjectured (guessed, darn it!) that the equation $A^4 + B^4 + C^4 = D^4$ had *no integer solutions*. Centuries went by, and no one knew if this conjecture was correct. You (and I) can certainly try examples, and I'm sure many people did, but no solutions were found. I am also sure that there was quite a lot of work done by both professionals and amateurs (which are we?) during those centuries. The equation seems so closely related to the famous Fermat equation.
In 1988, Noam Elkies published a paper entitled "On $A^4+B^4+C^4 = D^4$" in *Mathematics of Computation*. There will be a link to the paper on the web page associated with this meeting providing access to the article from any Rutgers terminal. You can understand a portion of this article. Elkies observes that

$$2682440^4 + 15365639^4 + 18796760^4 = 20615673^4$$

Elkies did not get this solution by a direct search, also called a *brute force search* or an *exhaustive search.* Such searches may not be practical without supporting evidence. A direct check of all the integers in the range up to those written above would take some time (that's an understatement!). So Elkies did some very clever mathematics to try to locate a range of numbers to search, and also to try to describe certain likely values of A, B, and C. He was successful, but there was substantial numerical and theoretical effort involved. You can check the equation above with `Maple`. I just did, and it wasn't hard, and didn't take very long (typing the numbers accurately took longer).

The postscript
Elkies wrote a short P.S. in his paper. He declared that while his (counter!)example to Euler's conjecture "still seems beyond the range of reasonable exhaustive computer search", his result inspired an effort to find smaller solutions. A person with access to fast highly parallel computer systems (many c.p.u.'s) tried to get a smaller solution. Success was achieved after about 100 hours of computation on fast machines with many processors. Roger Frye of the Thinking Machines Corporation found that

$$95800^4 + 217519^4 + 414560^4 = 422481^4$$

which is still not obvious to me, but `Maple` does agree.

My P.S. Compare our work on $a^2 + b^2 = 3c^2$. Our amateur (?) search reported no solutions, and there actually are *no solutions.* Numbers are funny, since Euler's equation does have solutions but they are not easy to discover.