Gerald's Column
by Gerald Fitton

In my column this month I briefly (and inadequately) respond to the huge quantity of correspondence I've received concerning Colton Software. Do you have Eureka? If so then I'd like to hear about it from you. The rest of this month's column deals with some mathematical topics − but you don't need to be a mathematician to follow them. I discuss the nature of mathematical induction and how it relates to recursively defined functions. Then I move onto an example of a recursively defined function; I consider it to be an interesting example because its exit condition is at an infinite number of recursions.

I complete my column with a quick look at the cube root function. A small handful of you working on the extended Heinz beans can problem have pointed out to me that this function is needed when you want to find the radius of a sphere of a given volume − but it doesn't appear in the PipeDream or Fireworkz manuals!

Colton Software

I must thank all those who have written to me making reference to my remarks about Fireworkz and Fireworkz Pro in the last edition of Archive. I shall include only two quotes but these are typical of the many.

The first is "I endorse everything you say − and more!" I do thank all of you for your support and I am touched by those letters I have received which acknowledge that it must have been difficult for me to write what I did.

The second quote is longer and has wider implications. I have added the words in brackets to make the meaning clearer. It is "Certainly Pro is not fit for the purpose intended and as such they (Colton Software) are in breach of the Sale of Goods legislation. If they do not agree (to a refund of the cost of upgrading from Fireworkz to Fireworkz Pro) I shall have no hesitation in issuing a Court Writ. . . . It is about time that we (users of Fireworkz Pro and users of other software which does not live up to its specification) . . . raised our voices . . . taking the matter to Court if necessary. . . Perhaps then we will begin to get what we paid for."

In response to this second quote I have to say that in spite of the accolades and persuasive remarks I have received (such as those of my first quote) I do not wish to co-ordinate such a protest to Colton Software. You must find another champion of your rights.

Longman Logotron's Eureka

I was interested to receive a copy of a letter sent to Alan Williams by Which? magazine. It would seem that I was wrong to assume that Which? simply looked at clock speeds when comparing the PC with the Archimedes. Indeed Which? asked Acorn for the names of packages similar to the PC's Word, Excel and Aldous Pagemaker. Acorn chose Style (as a word processor), Eureka (as a spreadsheet) and Publisher (as the DTP package).

I don't wish to comment further at this stage since Alan will be writing directly to Paul about the Which? tests, however Acorn's choice of spreadsheet has caused me to give some consideration to Eureka. From time to time I've been asked to comment on the relative merits of Eureka when compared with the spreadsheet packages which I know better. I have asked Paul if he can let me have a copy of Eureka for evaluation and, if he does so, then I'll report more fully in a future article.

Inductive Logic

I think that, even before my interest in mathematics, I was attracted to study logic; I regarded it as a branch of natural philosophy! I remember the syllogism which starts with the major premise "All cows eat grass". What struck me at a very early age was that deductive logic never increases the extent of human knowledge. Indeed, in order to make the statement "All cows eat grass" with certainty, we must have studied every cow already! So we already know that they all eat grass before concluding the syllogism with the eating habits of the individual cow introduced into the minor premise.

Inductive logic is a different matter. You can get somewhere new if you start with a set of particular examples and then work towards a general principle. Mathematical induction is a most useful tool which depends on the same principles as inductive logic (it is slightly different but I won't go into the differences right now). In its most general form mathematical induction is a process where you start with something complicated which you wish to prove and you show that this complicated thing is true if something a bit simpler is true. In turn the simpler thing is reduced to something even simpler . . . and so on − but not (usually) forever. An inductive chain of reasoning (like so many things in life) is easier to start than to stop. My four year old grandson has just discovered that asking "Why?" generates such a chain. One of my early Managing Directors used to classify his technical staff as '2 Why?' or '3 Why?' men; the MD always stopped with his fourth (usually unanswered question). Indeed, as we shall see, it is by studying anomalies at the end of an inductive sequence that we gain even greater understanding!

The Uncaused Cause

I have no wish to encroach on Paul's 'God slot' neither do I wish to offend those whose beliefs are atheist or religious. Nevertheless I do feel compelled to use as one of my two simple examples of inductive logic something which does encroach into that field. Many people believe that they are being logical when they claim that the recursive nature of the classic series of questions about causes (which eventually leads to the question "Who created the creator of the universe?") negates the claim that a creator must exist. I suggest that this logic is flawed and that you must look to the basic premise of inductive logic which, in this case, is interpreted as "You must know when to stop asking for a cause!" See if you can follow my next statement: "It is the recursive nature of 'cause' which forces you to accept that (if causality exists at all) then there has to be a *first cause* (an exit condition from the causal chain) which is often referred to as *the uncaused cause*".

The Recurrence Relationship

Enough for now of religious philosophy − I shall return to my 'God slot' example later! One manifestation of inductive logic used in mathematics is the recurrence relationship. An example which I use a lot at college is the factorial function. The definition of

factorial x (written as ***x!***) is best demonstrated with an example, factorial seven.

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1.$$

Now let me write 7! as a recurrence relationship. $7! = 7 \times 6!$

In more general terms $x! = x \times (x - 1)!$

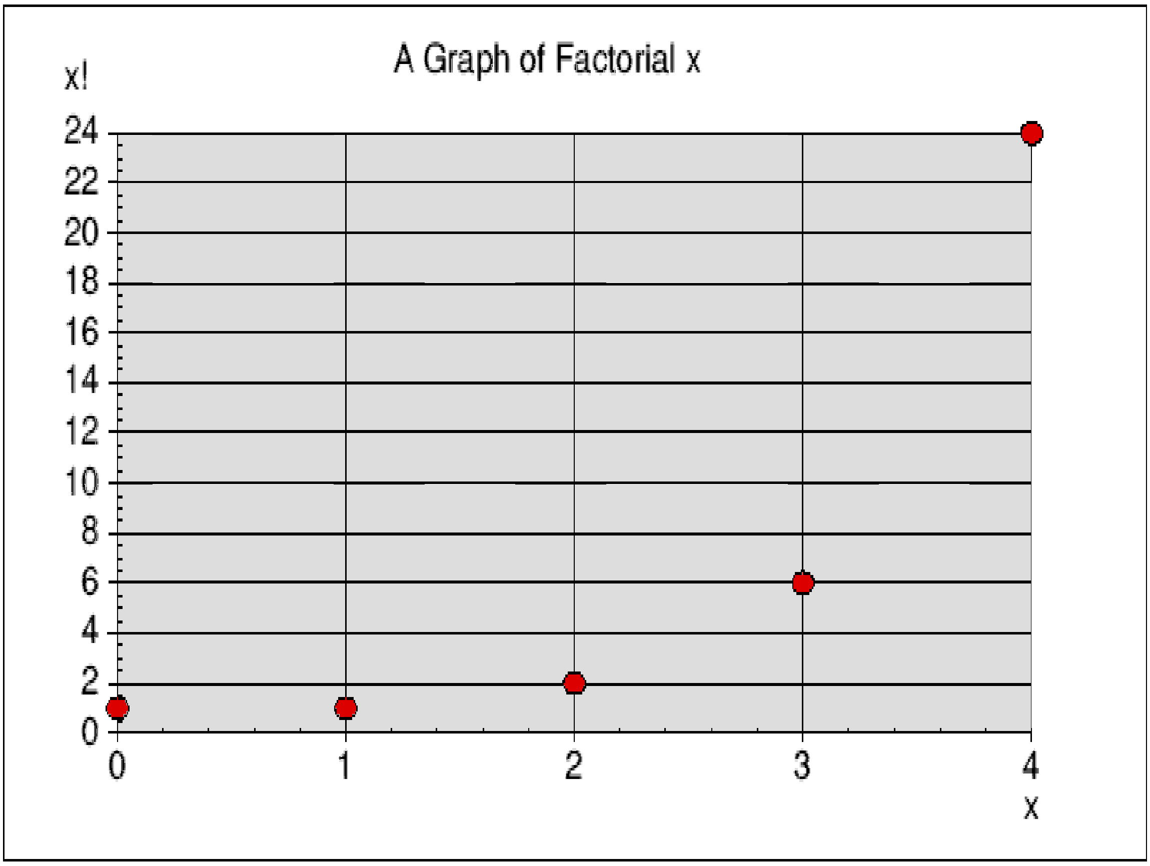
If you accept that $(x - 1)!$ is simpler than $x!$ then you will agree that I have defined the factorial function in terms of a simpler version of itself. It is knowing when to stop using the recursive definition that is all important. If you go one step too far then you'll finish up with nothing.

Have a go using the recurrence relationship (not the original definition) a couple of times to find $1!$ and you'll find that you're multiplying by zero − and anything multiplied by zero is the nothing you finish up with. Let me repeat. Knowing when and how to stop defining the function as a recurrence relationship is all important. You have to accept that the recurrence relationship $x! = x \times (x - 1)!$ is not and can not be true for every value of x (and, going back to my 'God slot' example, not every effect 'needs' a cause).

Those of you still in possession of that obsolescent tool, a calculator with a factorial function, will be able to discover for yourselves that $0! = 1$ is built into the software. This is the exception; factorial 0 is not 0 times something (and therefore nothing) but just a plain and simple (but illogical!) 1.

The Gamma Function

Study the graph of the function $y = x!$ which is shown below. In particular I want you to notice that it consists of a set of points with nothing in between. The reason there is nothing in between is because the factorial function exists only for positive integers.



In a situation like this mathematicians are always anxious to find out what goes on between the plotted points. You might think that there ought to be nothing there and, in terms of the simple factorial function, you're right. However, in the wider scheme of things (functions of a complex variable as opposed to functions such as factorial which only work with positive integers) there is a property that well behaved functions have called 'regularity'. Loosely translated a regular function through a few given points is the function giving the smoothest possible curve drawn through those points. It is relatively easy to prove that the points you've got for the factorial function (even excluding the factorial zero) are points on a regular (smooth) curve. What is rather harder (but well within the Pure Maths syllabus of most universities and a few good sixth forms) is finding the function. The function which gives the smoothest curve through all the factorial points is called the Gamma Function.

If you look carefully at this curve you might guess that it can be extended to negative numbers and that a smooth curve through the plotted points will be symmetrical about the vertical line at $x = 0.5$. Mathematicians have discovered the Gamma Function by looking at the information available (the factorial function) from a new and more enlightened perspective (regularity or smoothness). Lesser mathematicians have difficulty in accepting the inelegance of $0! = 1$ as an exception to the recursive rule $x! = x \times (x - 1)!$ It is the enlightened and broader perspective of regular functions of a complex variable which allows us to make sense of the exception to the recursive definition of the function.

So that you may follow my philosophical point allow me some repetition. If you define a function recursively then you must have an exit condition. From a narrow and limited point of view it may seem as though the exit condition is chosen arbitrarily, indeed, as we have seen for the factorial function, the exit condition may be an exception to the recursive rule which defines the function. My philosophical and mathematical point is that there is always a more enlightened and broader viewpoint (in the case of the factorial function this is the desire for smoothness) which places the exit condition as the natural exception to the recursive rule.

The Uncaused Cause

Back to my 'God slot' example. What is needed to overcome the logical inelegance of having to accept the existence if an uncaused cause is a new and wider perspective within which the uncaused cause takes its natural place in the scheme of things just as the ***0!*** takes its proper and natural place as a point on the smooth Gamma Function curve. Remember that it is the more enlightened and more elegant concept of regularity (smoothness) which allows us to extend the factorial function to include fractions and negative numbers.

To repeat myself, as soon as you begin to believe in cause and effect (causality) then, because of the very nature of causes, you have to accept the existence of an uncaused cause (so you'll know when to stop looking for causes). In the wider scheme of things 'cause' implies a time dependence and, speaking personally, I reckon that I'll have to understand a lot more about the nature of Eternity before I can see how the uncaused cause breaks the causal chain in a natural way.

Defining functions recursively has its place in the mathematical and philosophical scheme of things but (for this thesis at least) I want you to accept that, although it might be difficult to decide when and how to stop − in nearly all cases we must stop somewhere. Furthermore, before we start we must have a good idea where we want to stop and how to do it. Later, when we cease to "see through the glass darkly" we can put aside our imperfect recursive techniques. When I understand more about the nature of Eternity I shall be able to put aside my need for the childish concept of causality − until then causality has its place in my working model of the universe and (call it faith) I accept the exception and logical inelegance of the uncaused cause.

The Infinite Sequence

I must acknowledge that the inspiration for the next section of my monthly column is Colin Singleton's puzzle number 50, an infinite sequence of square roots. Although on the surface much of what I write will appear to be strikingly similar there is a difference; Colin's problem was about series of numbers, my article is about a series of functions defined recursively.

Infinite Recursion

In its most general form the problem which I shall set you is to evaluate a function which I shall call ***f(x)*** such that: ***f(x) = sqr(1+f(x + 1))***. Please note that ***f(x + 1)*** means "evaluate the function ***f*** using as its argument ***(x + 1)***"; unlike Colin's problem, it does not mean "the ***(x + 1)th*** value of ***f***". The function ***sqr()*** is the (aphrodisiac) square root function which featured in my column a couple of months ago. Those of you who have followed the earlier part of this month's column will recognise that, given a value for ***x***, then ***f(x) = sqr(1+f(x + 1))*** can be evaluated recursively.

Here's a feature of this problem which interests me; how do we determine the exit condition for this recursion? There is no natural exit point such as $x = 0$ or $x = 1$ because, at each recursion, x increases! What we have here is a recursive process which goes on forever (a bit like the Eternity to which I referred earlier)! So how do we get out of the recursive process? When do we stop? To our rescue comes a mathematical concept which it took me over 30 years to understand (as opposed to being able to do all the well known tricks with it). To our rescue comes Infinity! Now there is no way I'm going to explain Infinity to you today but I can't resist saying that it is not just a big number, nor is it a number bigger than you can think of, indeed it's not a number as you understand the concept of number and the usual rules of number manipulation don't apply to Infinity. Please allow me the licence to add that, in a similar way, Eternity is not just a long time, nor is it a length of time longer than any you can imagine; Eternity is not a length of time in the way we understand time and the usual rules of time and time dependance (causality) don't apply to those things which are truly Eternal.

The exit point for this recursive process (and many like it) is the mystical Infinity!

There are many tricks which mathematicians can do with Infinity. Amongst other things we can find out what happens when we get there; we can find out what happens beyond it − but that's another story. Also we can find out what happens on the way − and this is the clue to determining a more practical exit condition for the recursive process than the vain attempt to capture the mystic Infinity.

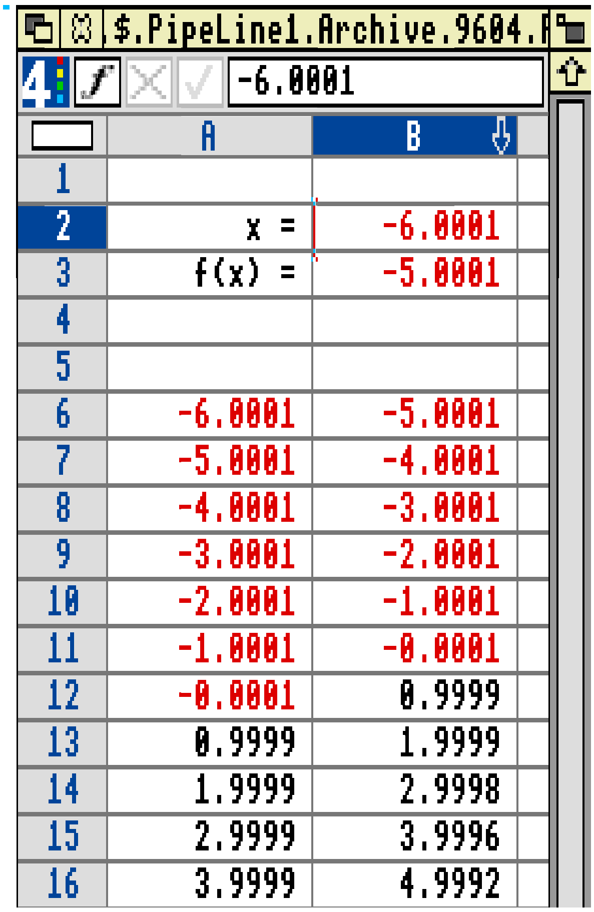
Convergence

Although $f(x + 1)$ increases linearly as we progress through the recursion, this recursion is a process which converges (and does so fairly rapidly) because the square root in the function ***sqr(1+f(x + 1))*** causes a reduction in value faster than the $f(x + 1)$ pushes it up. In other words, we can find an approximate value for the function by cutting off the recursion arbitrarily when we've had enough! What happens on the way to this Infinity is two things (a) you're never going to get there using a step by step recursive process and (b) that after a short while you're nearly there anyway! The conclusion is that the practical exit condition for an infinite recursion which is convergent is that you should proceed until you think you're near enough to the same answer as you'd get if you could repeat the process an infinity of times (which you can't) − and then stop!

I am not asking you to prove the convergence (though it's not too hard to do so); what I would like you to do is to write a custom function in PipeDream or Fireworkz or a procedure in BASIC which evaluates the function recursively. You can terminate the recursion after some reasonably large number of recursions (thirty or so) − so to determine your exit from the recursive process just count the number of recursions and stop when you feel you've done enough − get off the treadmill and do something more interesting instead!

Colin's problem and mine have the same answer. He reckons that, for $x > 0$, his infinite series evaluates to $(x + 1)$. In my version of the problem this translates to $f(x) = (x + 1)$. Both Colin and I agree on this but I add that, provided you choose your square root so that the recursion converges, then the recursive function evaluates to $(x + 1)$ even when x is not a positive integer. Like most mathematicians I am compelled to try to fill in the gaps between the points on the graph of the function. Indeed I believe that I have proved analytically that the function evaluates to $(x + 1)$ for any number, including negative numbers, fractions, irrational numbers (such as the aphrodisiac square root of 2), transcendental numbers (such as pi & e) and even complex numbers (such as the mystical and imaginary square root of minus 1). In case there's a flaw in my analysis let me call it a conjecture. My conjecture is that ***f(x) = (x + 1)*** for all sorts of numbers. Finally, let me repeat the proviso. Unless the number for which you are trying to find the square root is the number zero then there are always two square roots. You must choose your square root in such a way that the recursive process converges. I don't regard being fussy about which square root to choose as cheating. I regard it as a way around an inevitable imperfection of the recursive method of defining a function.

Using a recursive procedure which handles the so called Real numbers you should be able to discover fairly easily whether what I say is true for negative numbers and fractions. A screenshot of my non recursive numerical method for negative decimal fractions using PipeDream is shown below.


$.PipeLine1.Archive.9604.

-6.0001

	A	B
1		
2	x =	-6.0001
3	f(x) =	-5.0001
4		
5		
6	-6.0001	-5.0001
7	-5.0001	-4.0001
8	-4.0001	-3.0001
9	-3.0001	-2.0001
10	-2.0001	-1.0001
11	-1.0001	-0.0001
12	-0.0001	0.9999
13	0.9999	1.9999
14	1.9999	2.9998
15	2.9999	3.9996
16	3.9999	4.9992

I have included the file which produces this screenshot on the monthly disc. The recursive sequence is evaluated to about 30 terms starting with −6.0001 as the value for ***x***. You will see that ***f(x)*** evaluates to −5.0001 (to four decimal places), which is ***(x + 1)***. Mind you, as I said earlier, I have been a bit selective about whether to use the positive or negative square root and, in order to guarantee convergence, the definition changes (automatically) to the other square root around row 12 or 13 (you guess which) of the above sheet!

In order to study my conjecture that ***f(z) = (z + 1)*** numerically when the variable is a complex number, you will have to use the complex function feature of PipeDream or Fireworkz to evaluate ***f(z)*** for a complex number such as ***z = {5,−6}***. (I am using the usual convention which, in the example, means that the real part of ***z*** is ***5*** and the imaginary part is ***−6***.) My helpful advice is that the complex function 'square root' in PipeDream and Fireworkz is written as ***c_power(complex_no,{0.5,0})*** where ***complex_no*** is typed into a slot as an array such as ***{5,−6}*** and the second argument, ***{0.5,0}*** is the power (the square root) to which the complex number ***complex_no*** is raised. If my conjecture is right then when, ***z = {5,−6}***, the infinite sequence of square roots sums to the complex number ***{6,−6}***. Can you prove it numerically (with a recursive spreadsheet custom function or recursive BASIC procedure) rather than analytically (as I believe have)? My helpful hint is that PipeDream and Fireworkz both calculate only one of the square roots, usually the principal square root (eg it calculates the square root of +4 as +2 and never as −2) and this is not always the 'correct' root to choose!

I will have more to say on this topic of recursive functions in a future article, but, for now, if you feel like sending me your attempts then I'll be pleased to correspond with you.

The Cube Root Function

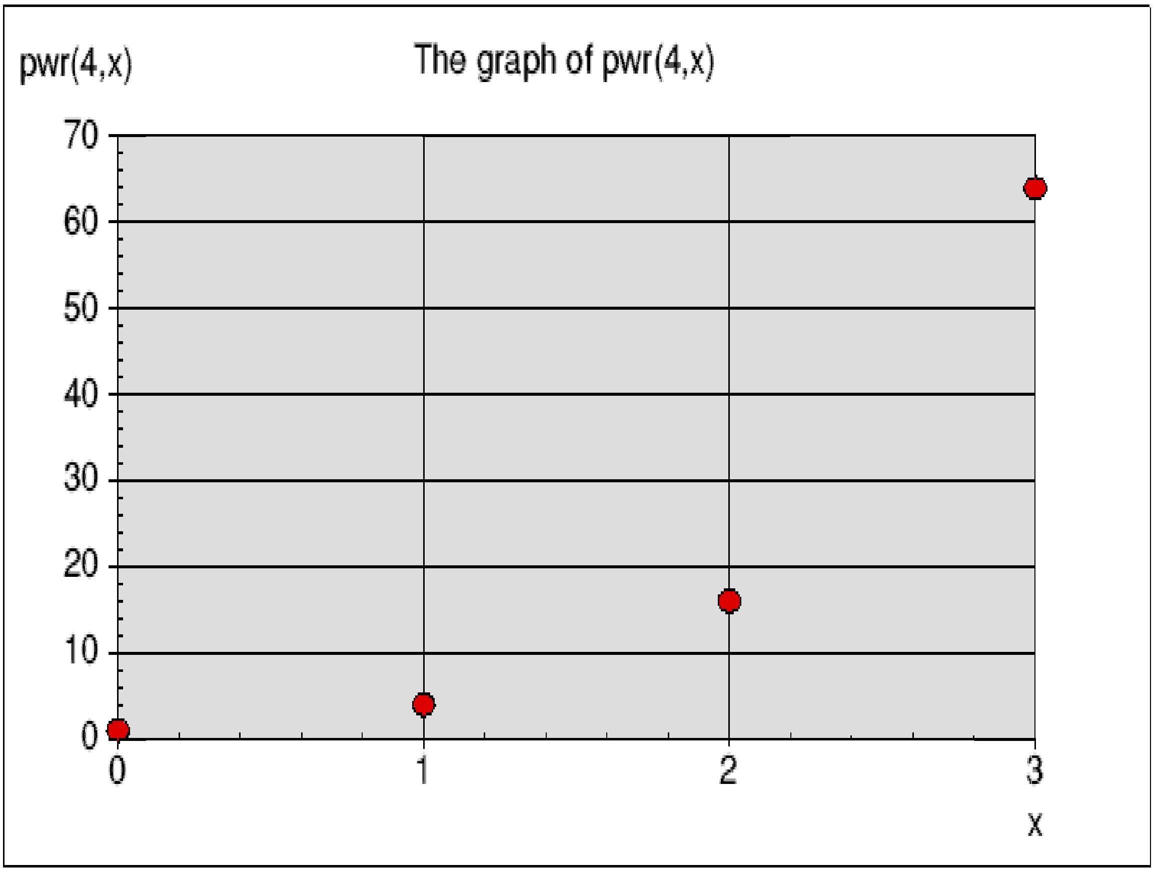
Although I haven't been employed as a full time lecturer all my life, I have been doing a bit of part time teaching for a very long time. Even when I worked at Vickers nearly forty years ago I was commandeered to do about eight hours a week teaching in the apprentice training school (augmenting what our day release apprentices were taught at the local college) and I taught A level maths at the college one evening a week. So really I ought to know that it is sometimes the simplest things which people find the most difficult. Finding the cube root of a number using either PipeDream or Fireworkz is one of those things which is so simple to those in the know that even the writers of the PipeDream and Fireworkz manuals didn't include it!

I received an interesting letter from a correspondent who shall remain nameless to protect his embarrassment. He had done much accurate and intelligent work on the Heinz beans can but, when trying to calculate the radius of a sphere of a given volume he needed to find, not the square root but the cube root of a number. Well, being a bit smarter than the usual computer user, he had a look in the PipeDream manual for the cube root function. He found the square root function, sqr(), but couldn't find a corresponding cur()! He didn't give up but decided to use logarithms to find the cube root (using logarithms this way is now a lost art) but ran into a similar problem − no antilogs! At the end of a very long tether he wrote to me. He is not alone. As a result of the Heinz beans can problem I have had a small handful of letters (one from a chartered accountant trying to get to grips with Excel) asking me how to include a cube root in a spreadsheet.

The Power Function

I guess that all of you know that '***a to the power three***' really means ***(a × a × a)*** and can be written that way in a spreadsheet slot. In the same way you can find ***(a × a × a × a)***, ***(a × a × a × a × a***), etc. Indeed, if you followed the earlier parts of this month's column you'll recognise that we can write the power function of the number ***a*** in a recursive way as ***pwr(a,x) = a × pwr(a,(x − 1))*** with an exit condition that I'll leave you to determine.

Here is a graph of the power function using ***a = 4*** for a few integer values of ***x***:



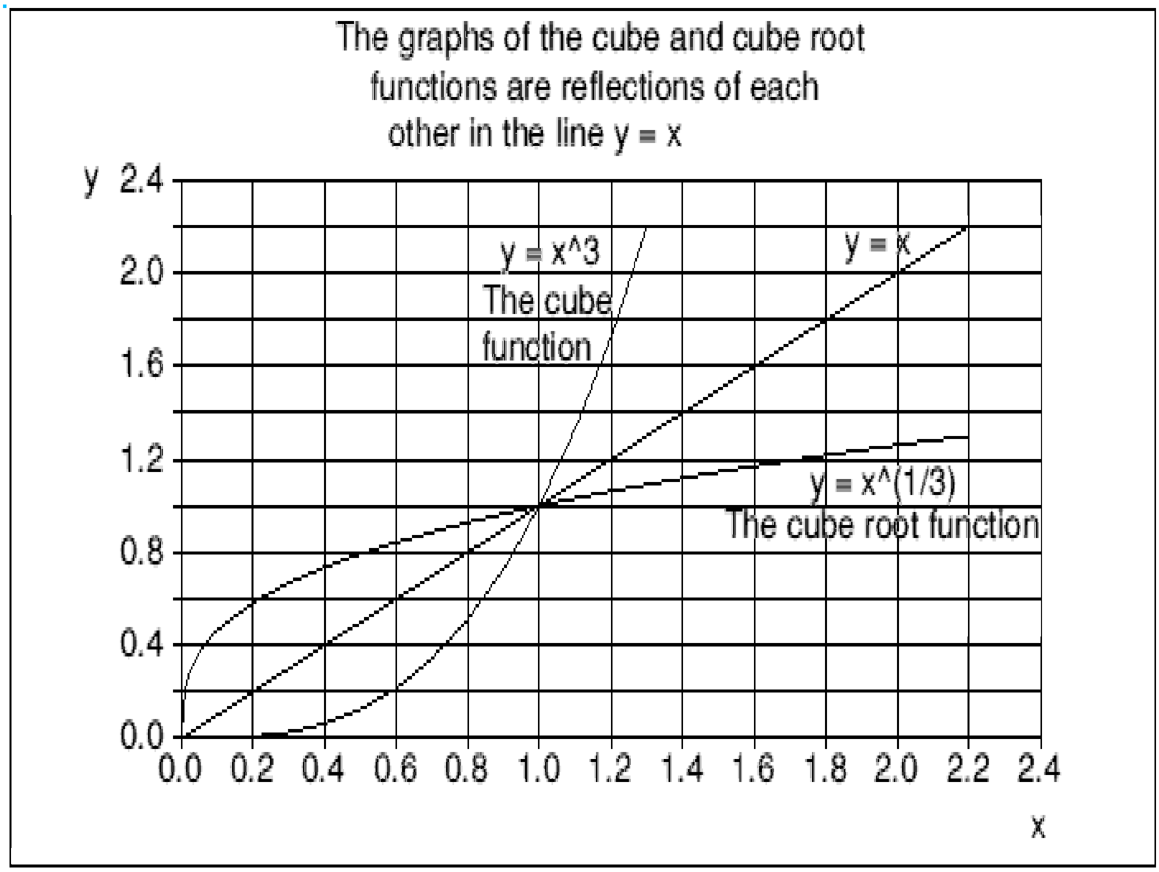
This method of multiplying together ***x*** lots of ***4*** gives values for the power function only when ***x*** is an integer. What about the values of ***x*** in between the integers? This problem is similar to the problem we had with the factorial function and the answer is the same, namely that we find a way of drawing the smoothest possible curve through the existing points; this smooth curve technique can be applied not only to fractions such as ***x = 1/3*** but also to negative values of ***x*** such as ***x = −3***.

Allow me to cut the discussion short and say that the cube root function has to be a version of the power function, in particular it is: ***cube_root(b) = pwr(b,(1/3))***. In a spreadsheet this power function is written as ***y = (a^x)*** so that if you want to find the cube root of ***64*** you would enter ***64^(1/3)*** into the slot; the answer is ***4***. To find the ***nth root of a*** you enter ***a^(1/n)*** into the spreadsheet. Using this power function technique you can find square roots as: ***sqr(a) = a^(1/2) = a^(0.5)***. Negative values of ***x*** in the power function imply division, for example, ***4^(−3) = 1/(4^3) = 1/64 = 0.015625***. An interesting result is the evaluation of ***(a^0)***; all values of ***a*** give the same answer! Try it and see what you get.

Perhaps on another occasion I'll explain how you use the fact that ***cube_root(a × a × a) = a*** together with something called ***the multiplication rule of indices*** to prove that this method is a natural extension of the power function to powers which are not positive integers.

Inverse Functions

The graph below shows that the cube and cube root functions are reflections of each other in the line ***y = x***. I have created this graph as a live graph in PipeDream. I think that this graph is an excellent example of the power of PipeDream's charting package. It demonstrates a graph which contains (a) two lines which use different ranges for their ***x*** values and (b) many pieces of text placed anywhere on the graph whilst (c) the graph remains live. On the Archive monthly disc you'll find the PipeDream spreadsheet and the live graph which it generates. At college I use Excel and I have yet to discover how to produce such a graph with that package.



The Antilogarithm Function

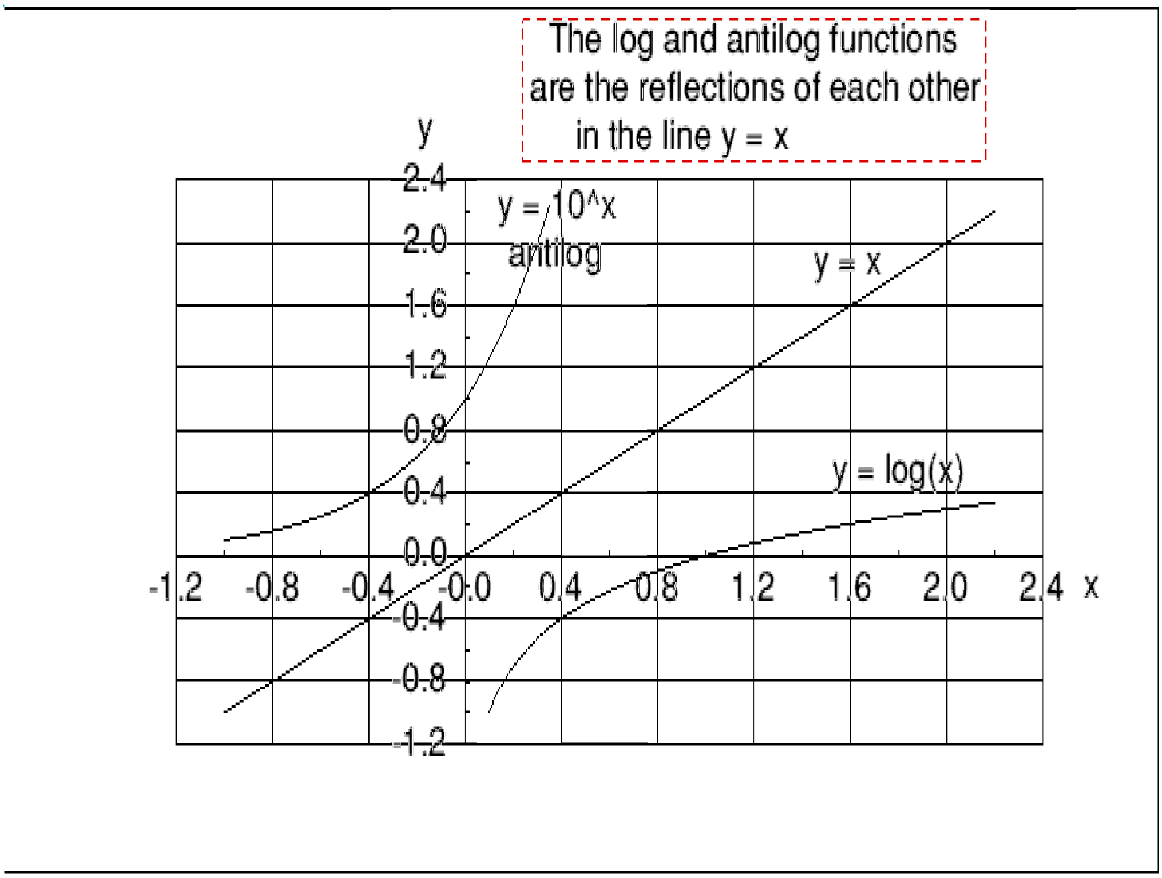
This function also perplexed my reader. The antilog function is the inverse of the logarithmic function. By this I mean that if ***y = log(x)*** then ***x = antilog(y)***. A consequence of this is that ***antilog(log(x)) = x***; applying the antilog function to a logarithm gets you back to the value from which you started. Again try that in a spreadsheet when you know how to find an antilog and you'll see that it's true.

Both PipeDream and Fireworkz include the ***log()*** function − but no antilogs! Was this a serious omission? Once again the answer is "No!" and once again the power function comes to our rescue. The logarithmic function ***log()*** is more properly called the 'log to the base 10'. The 10 comes into it when we consider the antilog. The antilog function can be expressed as ***antilog(y) = 10^y***. Colton Software (and I guess all other spreadsheet writers) haven't mentioned this anywhere in their manuals because to them it's all too obvious. Well, to at least one of my correspondents it wasn't.

So what is that other logarithmic function, ***ln()***? It's an even more tricky job to find its antilogarithm but there is a clue in the name. This logarithmic function is called 'logarithm to the base e'. I wonder if you know how to find ***e^x***. Here's the answer. If ***y = ln(x)*** then

$x = exp(y)$ and so $exp(ln(x)) = x$.

The graph below shows two functions (the logarithm and exponential functions) which are reflections of each other in the line $y = x$. Once again the spreadsheet and its live chart are on the Archive monthly disc; I have yet to find a way of generating such a graph in Excel.



Let me stop there and await your comments. If, from your correspondence, it seems that you're interested in some of the topics I've raised then I'll continue or expand on them.

To all of you

Thank you for all your letters. Please write to me (with a disc showing examples if appropriate) at the Abacus training address given at the back of Archive. If you use Eureka then I'll be specially interested to hear your comments about it.