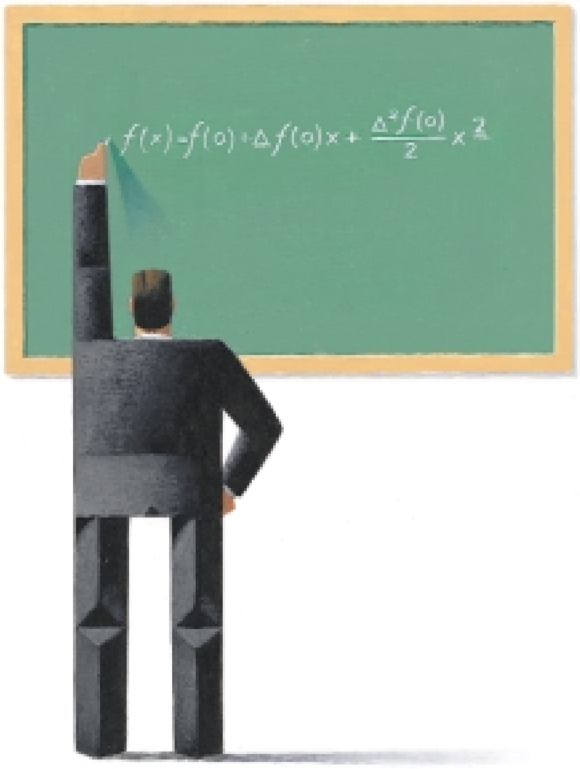
by Jeffreys Copeland and Haemer



Nature has … some sort of arithmetical-geometrical coordinate system, because nature has all kinds of models.
– R. Buckminster Fuller

Why is it that we entertain the belief that for every purpose odd numbers are the most effectual?
– Pliny the Elder

High-School Algebra, Backwards

You know how to write programs to do high-school algebra. For example, you could use *bc* to compute the sequence generated by $y=x^2$

```
$bc
for (i=0; i<10; i++)
 i^2
 0
 1
 4
 9
 16
 25
 36
 49
 64
 81
```

That wasn't hard.

But suppose you're given the problem backwards. Suppose someone gives you the sequence 0,1,4,9,16,25,36,49, 64,81, … and asks you for the equation, or for its 51st term. You can look at it and say, "That's the squares, and the 51st term is 2500."

Trivial, we admit, but now suppose someone gives you the sequence 5,4,5, 14,37,80,149,250,389,572, … . Quickly now–what equation generates this sequence? What's the 51st term?

Our first attack is to look it up in Sloane's On-Line Encyclopedia of Integer Sequences, (`http://www.research.att.com/~njas/sequences/`).

Unfortunately, it's not there.

Now what? Amazingly, there is a general method to attack this problem, with several wonderful (to us, anyway) properties:

- It's easy to do.
- It's easy to remember.
- It makes sense.
- It's not widely known.
- It's useful.
- It gives us an excuse to write some code that will fit in a single column.

The code isn't actually needed, it's just fun, so we'll start with the math.

Actually, the math is fun, too, but the math requires some familiarity with that advanced mathematical technique, high-school algebra; if that scares you off, just skip to another article. In case you're still on the fence, you'll need to know how to multiply out and simplify equations like $y=(x+1)(x-1)$.

It's Easy To Do

Let's start by going back to the squares

y:0,1,4,9,16,25,36,49,64,81, …

Gillian Haemer once told us, when she was a little girl, the differences between these numbers were just the odd numbers

Δy:1,3,5,7,9,11,13,15,17, …

and that the differences between those were always the same

$\Delta^2 y$:2,2,2,2,2,2,2,2, …

and that the differences between *those* were always zero

$\Delta^3 y$:0,0,0,0,0,0,0, …

It should be clear that we can use this process to generate even the 51st term of $y=x^2$ by taking the 50th term and adding the 50th odd number. And we could get the 50th odd number by taking the 49th odd number and adding 2. This sort of building-up process will work for many sequences. Let's try our mystery sequence:

y:5,4,5,14,37,80,149,250,389,572, …

Δy:-1,1,9,23,43,69,101,139,183, …

$\Delta^2 y$: 2,8,14,20,26,32,38,44, …

$\Delta^3 y$:6,6,6,6,6,6,6, …

$\Delta^4 y$:0,0,0,0,0,0, …

It should be clear that we could calculate the 51st term of this series the same way we sketched for x^2; it would just be tedious. With only a little math, we can cut down on the calculation and get a closed-form solution.

It's Easy To Remember

With no justification, we're going to introduce an idea and a notation.

Suppose $y=x(x-1)(x-2)\ \ldots\ (x-n+1)$ [n terms]. We could write this with factorials

$$y=\frac{x!}{(x-n)!}$$

Instead, we're going to call this function a *factorial power*, which we'll write like this: $x^{\underline{n}}=x(x-1)(x-2)\ldots(x-n+1)$. (We could, we suppose, now write $5!=5^{\underline{5}}$ but we personally still like 5! better.)

Just as with regular powers, we'll let $x^{\underline{0}}=1$.

Why this new notation? It makes our method easy to remember and understand. We're about to use factorial powers in *Gregory's Theorem*:

If $y=y_0, y_1, y_2, \ldots$ is a sequence, then the sequence is generated by the polynomial

$$y=f(x)=f(0)+\Delta f(0)x+\frac{\Delta f(0)}{2}x^{\underline{2}}+\frac{\Delta f(0)}{6}x^{\underline{3}}+\ldots=\sum\frac{\Delta^{i} f(0)}{i!}x^{\underline{i}}$$

Here, by $\Delta^{i}f(0)$ we mean "find the ith set of differences, then take the zeroeth term."

Looks complex, but perhaps oddly familiar. Remember this?

$$y=f(x)=f(0)+f'(0)x+\frac{f''(0)}{2}x^2+\frac{f'''(0)}{6}x^3+\ldots=\sum\frac{D^i f(0)}{i!}x^i$$

where

$$D^n f=\frac{d^n f}{dx^n}$$

That's just Taylor's theorem, from elementary calculus.

So, we have a mnemonic: our formula is just like the Taylor-Maclaurin series, but with differences instead of derivatives and with factorial powers instead of regular powers.

Shall we try it? Let's do $y=x^2$, just for Gillian. Gilly noticed that Δ^3 and all higher differences are 0. This leaves us with

$$y=f(x)=f(0)+\Delta f(0)x+\frac{\Delta^2 f(0)}{2}x^{\underline{2}}$$

Plugging in, we get

$$y=f(x)=0+1x+\frac{2}{2}x^{\underline{2}}=x+x^{\underline{2}}=x+x(x-1)=x^2$$

Eureka.

And it's easy to remember.

It Makes Sense

Let's digress again for a few seconds. Be patient with us.

Suppose we have a sequence generated by a factorial power: $y=x^{\underline{n}}=x(x-1)(x-2)\ldots(x-n+1)$. Can we write an equation that generates differences between pairs of successive terms, $\Delta y=f(x+1)-f(x)$? Sure. It's just high-school math:

$$\Delta y=(x+1)(x)(x-1)\ldots(x-n)-x(x-1)(x-2)\ldots(x-n+1)=$$

$$x(x-1)\ldots(x-n)[(x+1)-(x-n+1)]=$$

$$x(x-1)\ldots(x-n)[n]=nx^{\underline{(n-1)}}$$

What's this result, $\Delta x^{\underline{n}}=nx^{(\underline{n-1})}$, look like? Simple derivatives of polynomials in elementary calculus.

When working with sequences, factorial powers give you the same kind of easy manipulations that regular powers do for continuous functions. Plus, it's not hard to convince yourself that every polynomial can be expressed as the sum of factorial powers times constant coefficients:

$$y=\sum C_n x^{\underline{n}}$$

(The ambitious reader can prove from this that Δ^n for any polynomial of degree n will be 0.)

Let's figure out the coefficients, C_n. Take a polynomial,

$$y=C_0x^{\underline{0}}+C_1x^{\underline{1}}+C_2x^{\underline{2}}+\ \ldots$$

What's the first difference?

$$\Delta y=C_1x^{\underline{0}}+2C_2x^{\underline{1}}+3C_2x^{\underline{2}}+\ \ldots$$

What's the first term of that sequence? $\Delta^1 y(0)=C_1$.

Likewise, the second difference is:

$$\Delta^2 y=2C_2x^{\underline{0}}+(3\text{x}2)C_3x^{\underline{1}}+(4\text{x}3)C_4x^{\underline{2}}+\ \ldots$$

And the leading term of that sequence is $\Delta^2 y(0)=2C_2$.

Continuing on this way, we see that the first term of the n^{th} difference is

$\Delta^n y(0)=n!C_n$.

In other words,

$$C_n=\frac{\Delta^n y(0)}{n!}$$

hence Gregory's theorem. (The Taylor expansion works for an exactly analogous reason, of course.)

This, then, shows us why Gregory's theorem makes sense: it creates a sequence whose Δ's match the sequence you first started with.

It's Not Widely Known

We didn't learn Gregory's theorem in high school. Did you? It's an elementary result in what's called the "Calculus of Finite Differences." It's not hard to use; it's easy to learn, understand, and remember; it's an answer to what seems like an elementary question; yet no mathematician we've ever asked has even known about Gregory's theorem or factorial powers.

Lots of math is like that: accessible, just not fashionable.

It's not widely known.

It's Useful

We confess that we first learned about finite differences as undergraduates, from a book of Haemer's mother's that we found tucked back on an out-of-the-way bookshelf and mysteriously pierced by a nail hole: Lancelot Hogben's "An Introduction to Mathematical Genetics," New York, W. W. Norton & Company Inc. [1946].

We've never seen another copy of Hogben's book, but you can find a modern treatment of finite differences in "Concrete Mathematics," ISBN 0-201-55802-5, another in the long line of amazing books by Don Knuth–this one coauthored with Ron Graham and Oren Patashnik. The book, typeset by Knuth himself, covers math the authors believe is accessible and useful to computer programmers, but not typically covered in degree programs because it's out-of-vogue. Unlike Hogben's book, "Concrete Mathematics" is regularly in stock in our local Barnes and Noble.

We agree with Graham, Patashnik, and Knuth. We've never seen it in any other books, but since reading Hogben, we've used Gregory's theorem to attack everything from genetics problems to Sunday supplement puzzles. It's useful.

It Gives Us An Excuse ...

"Well, that's nice. Where's your code?"

Impatient, aren't you? Let's go back to our other example:

y:5,4,5,14,37,80,149,250,389,572, …

Δy:-1,1,9,23,43,69,101,139,183, …

$\Delta^2 y$:2,8,14,20,26,32,38,44, …

$\Delta^3 y$:6,6,6,6,6,6,6, …

$\Delta^4 y$:0,0,0,0,0,0, …

Fourth-degree terms and above go away, so our equation will be

$$f(x)=f(0)+\Delta f(0)x+\frac{\Delta 2f(0)}{2}x^{(2)}+\frac{\Delta 3f(0)}{6}x(3)$$

Plugging in, we get

$$f(x)=5+(-1)x+\frac{2}{2!}x^2+\frac{6}{3!}x^3$$

Oh, ugh. We can hardly wait to simplify that.

One alternative would be to use a symbolic mathematics package, like Maple or Mathematica. These aren't free, but an AltaVista search for `+'symbolic algebra' +linux` returns a wide range of other open-source offerings that could fit the bill. We haven't tried any of these, but would love to hear reviews from anyone who has.

We need something simple that will fit in our margins. A trip to CPAN, `http://www.cpan.org`, leads us to Matz Kindahl's package, `Math::Polynomial`, which lets us create "polynomial" objects we can do arithmetic on.

For example, let's multiply ($3x$+2) by ($3x$-2):

```
#!/usr/bin/perl -w
# $Id: polytest,v 1.1 2000/11/25 ... jsh Exp $

use Math::Polynomial;
use strict;

Math::Polynomial->verbose(1);

    #3*$X + 2
my $p = Math::Polynomial->new(3,+2);    #
    #3*$X - 2
my $q = Math::Polynomial->new(3,-2);    #

print "($p)*($q) = ", $p*$q, "\n";
```

Running this gives us the expected result.

```
(3*$X + 2)*(3*$X + -2) = 9*$X**2 + -4
```

We can use `Math::Polynomia` to write a program that takes a sequence as input and prints out the polynomial it comes from.

```
#!/usr/bin/perl -w
# $Id: gregory,v 1.6 2000/11/27 ... jsh Exp $

use strict;
use Math::Polynomial;

sub delta {   # finite diff of a seq
   my @delta;

   for (my $i = 1; $i < @_; $i++) {
      push @delta, ($_[$i] - $_[$i-1]);
   }
```

```
    @delta;
}

sub fact {    # factorial
    return 1 if $_[0] < 2;
    my $f = 1;
    $f *= $_ foreach (2..$_[0]);
    $f;
}

sub fact_pow {     # factorial power
    my $f = Math::Polynomial->new(1);
    foreach (1 .. $_[0]) {
        $f *= Math::Polynomial->new(1,1-$_);
    }
    $f;
}

# non-zeroes in seq?
sub non_zero { grep($_ != 0, @_) }

# grab the input sequence

my @s;    # array of finite diffs
    # s[0] is original seq
    # s[1] is 1st f.d.
    # etc.

while (<>) {
    # words from input stream
    my @l = split /\W/, $_;
    # discard non-numbers
    @l = grep /^\d+$/, @l;
    push @s, @l;
}

# calculate coefficients

my @c;    # coefficients of final equation

for (my $i=0; non_zero @s; $i++) {
    $c[$i] = $s[0]/fact $i;
    @s = delta @s;
}

# Gregory’s theorem

my $p = Math::Polynomial->new(0);
for (my $i = 0; $i < @c; $i++) {
    $p += $c[$i]*(fact_pow $i);
}

Math::Polynomial->verbose(1);

print "$p\n";
```

And here is the result from our mystery sequence.

```
$X**3 + -2*$X**2 + 5
```

If we wanted this to run more quickly, we could tune `fact()` and `fact pow()` by saving results we already know in an array. Once we know $x^{\underline{2}}$ from an earlier calculation, we could look it up in our computation of $x^{\underline{3}}$ instead of recalculating it. (This is known as *memoizing*.)

On the other hand, this would take more development time, and the program seems fast enough as it is. We chose to use `Math::Polynomial` instead of investing in an elaborate symbolic math package, or writing our own, for the same reason.

Still, the reason we spent an hour or so writing a program was because we wanted something that worked faster than a hand calculation. Okay, if we’re not optimizing for development time (zero if we’d done the algebra with pencil-and-paper), and we’re not optimizing for program performance, what are we optimizing for?

Our own amusement.

Speaking of which, we’re amused by the observation that Gregory’s theorem can be rewritten in the following way:

$$\sum \frac{\Delta^i f(0)}{i!} x^{\underline{i}} = \sum \Delta^i f(0) \binom{x}{i}$$

This makes us think there might be some nice combinatoric interpretation or application of the formula. Can any reader give us one? Until next time, happy trails.

Jeffrey Copeland (`copeland@alumni.caltech.edu`) *is currently living in the Pacific Northwest, where he spends his time writing UNIX software in a large development organization and fighting damp rot.*

Jeffrey S. Haemer (`jsh@usenix.org`) *works at Minolta-QMS Inc. in Boulder, CO, building laser printer firmware. Before he worked for QMS, he operated his own consulting firm and did a lot of other things, like everyone else in the software industry.*

Note: The software from this and past Work columns is available at `http://alumni.caltech.edu/~copeland/work` *or alternately at* `ftp://ftp.cpg.com/pub/Work`.