**Mathematical Methods in Earth Sciences**

Lecture 2 - Mar/14/2018

Series Expansions

Say we have a function which is complicated and we would like to simplify. Often, we can rewrite this equation in an approximate form where ,, are constants. This is incredibly useful when is small (**why**?). We call this approach a *series expansion* of the particular function we are interested in.

**Example:** you have already been told that cos when is small.

What we would like is a general way of determining the series expansion of a function . There are a couple of ways of doing so. One way is graphical, and just depends on the definition of a gradient.

Say we know the value of a function at : . We want to know the value of this function at some other location . The gradient between these two points is just

As long as is not too far away, then the gradient between these two points will be similar to , the value of the derivative at = 0 (**why**?). So we can write

Simple rearrangement then allows us to solve for :

– (1)

In other words, if we know the value of the function and its first derivative at = 0, we can extrapolate to get the value of the function at other values of . This is very useful!

**Example.** Let’s say . Then we can use equation (1) to write

which is a good approximation as long as *x* is small (and in radians).

The accuracy of our extrapolation depends on how big is. To get a more accurate expression than equation (1), let’s just assume that the general expression is a polynomial:

(2)

where ,, are the unknown constants that we want to find.

If we set = 0 then we obtain .

If we now differentiate our polynomial, we get

Again setting = 0, we obtain . Substituting back into equation (2), we get

. Ignoring the higher-order terms (,), we have now recovered equation (1).

But we can keep going. Repeating the procedure again and again, we get , , and so on. So we can determine all our unknown constants ,, in terms of the value of the function and its derivatives at = 0.

Substituting these values back into our original polynomial, we end up with a series expansion for the function , known as *Maclaurin’s Series*:

Here denotes the *r*-th derivative of *f* and *r*! = *r*(*r*−1)(*r*−2) · · · (2)(1) is called ”r-factorial”. Note that 0! = 1, and that *r*! is not restricted to either positive or integer values of *r* (though the expression for *r*! given here only works for positive, integer values). Maclaurin’s series is a specific type of a more general class of series expansions, called *Taylor series*.

****

**So what**? Series expansions allow us to find simple approximations for complicated expressions.

**Example** Let . Differentiating, we have =1 for all *n*. So using Maclaurin’s series we have

So if is small (such that ) then we can approximate (try it on a calculator!).

What about ?

**Example** Let . If we differentiate once, we get

.

Setting , we have .

Generalizing, we have

and

Substituting all this back into the Maclaurin series, we get

This is called the *Binomial expansion* and is very useful. It works for positive, negative and fractional *n*. A particularly useful application is to remember that

when *x* is sufficiently small.

For instance, if you had to evaluate (1.1)10, you would rewrite this as (1 + 0.1)10 and then approximate it using the Binomial expansion as 1 + (10 × 0.1) = 2, which is about 25% too small.

**How would you improve your estimate?**

**Example** We can similarly expand . After a bit of algebra, we get

**Example** The relativistic kinetic energy of a particle *KE* is

What does this reduce to when ?

**Example** Plank radiation formula is

(Energy per unit volume per unit frequency)

What does this reduce to when ?



**Example** What does

reduce to as ?

**Hint** The main difficulty in carrying out a simplification of this kind is in recasting the expression into one in which one quantity is small. e.g. to find an approximate expression for , rewrite this as and then for you can approximate the answer using the first few terms of the binomial expansion (.