Math 373: 01 — Fall 2000
MW8 SC–205
Prof. Bumby

Taylor Series and Interpolation

In order to appreciate the organization of material on interpolation, we should examine what it means to compute a function. Although your calculator appears to put all the familiar functions at your fingertips, this is only because a method has been found to approximate the values of these functions to calculator accuracy using operations like addition and multiplication that can be performed exactly. In particular, if you needed values to 20 decimal places, you would need to extend these formulas.

Likewise, you can multiply two 10 digit numbers on your calculator and get the full 20 digit product — but only if you plan for it and use some multi-precision programming. If you just press the *multiply* key, you will be shown a floating-point representation of the first ten or twelve digits of the answer and later digits will be lost forever. If further work leads to an answer where those lost digits are needed, the calculator will just make something up.

This means that expressions that are algebraically equivalent may have different computational properties. With floating point arithmetic, the largest number appearing in a calculation sets the scale. Only a fixed number of places *at that scale* are available in a calculation. When the scale is increased by multiplying numbers, a fixed *relative* accuracy is maintained while the *absolute* accuracy becomes essentially meaningless. There is also a *round-off error* caused by not being able to keep all of the digits of the product. When the scale is reduced through the subtraction of close quantities, one has the same *absolute* accuracy, but the *relative* accuracy is reduced. When converting an expression for a function into a program for computing it, one should avoid operations that temporarily introduce numbers that change the scale of the computation.

Interpolation formulas lead to expressions that may be safely evaluated, but an attempt to rewrite these expressions in a more algebraically satisfying form may be disastrous. In particular the standard form of a polynomial as a sum of multiples of powers of x is only useful for representing the polynomial near $x = 0$ (where higher degree terms get smaller and have little influence on sums accumulated starting from the constant term). When evaluated far from zero, it is likely that individual terms may be much larger than the total and there will be significant cancellation leading to loss of accuracy. By contrast, the Newton form of the interpolation formula tends to be dominated by its constant term with later terms becoming progressively smaller. This is fortunate, since the calculation of divided differences involves subtracting two close numbers (which loses relative accuracy) followed by rescaling that makes the size of the number seem reasonable. This loss of accuracy is exactly matched by the small size of the factor that will multiply this coefficient when the formula is evaluated.

It is worth looking at this more closely. When divided differences are calculated, one has

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}.$$

In use, one sets $f[x, x_0, \ldots, x_n] = 0$ for a sufficiently large n and works backwards, inverting the definition of divided difference to obtain

$$f[x, x_0, \ldots, x_{k-1}] = f[x, x_0, \ldots, x_k] + (x - x_k)\, f[x_0, \ldots, x_k].$$

When all the x_i have been processed in this way, we have our estimate for $f(x)$.

The examples in the textbook illustrate the formulas, but not this feature of the formulas. Here is an example in which the values are close enough together that you can see the loss of accuracy in building the table — and the return of accuracy in computing the interpolating polynomial. The data was obtained from a standard function in *Maple* with `Digits:=20;`.

Exercise S7. Consider the function for which *Maple* has provided the following information.

x	$f(x)$	$f'(x)$
.250000	.98443592929585270492	−.12402597732272692273
.250002	.98443568124292139555	−.12402695398658105529
.250004	.98443543318803675885	−.12402793065006343125
.250006	.98443518513119879555	−.12402890731317404769
.250008	.98443493707240750640	−.12402988397591290168
.250010	.98443468901166289214	−.12403086063827999029

Suppose that you want to compute $f(.25000513)$. First try interpolating polynomials of degree 1, 2, and 3 that use only given values of $f(x)$ and **not** any of the values of $f'(x)$. Then try an appropriate cubic Hermite interpolation. How do the differences between the different approximants compare to what you expect from the form of the error term? In building divided difference tables, you should look for ways to avoid losing accuracy at the beginning of the computation, but later steps may be done on to calculator accuracy.

(I have the 20 place value that *Maple* computed for this function, so the true error can be found after this exercise is done.)