

Lecture Content Summaries for MATH3201 (from week 4)

Travis Mitchell

Updated: 21 March, 2019

1 Week 4

1.1 Recap of the first three weeks

Started by looking at *Taylor series* approximations for a function value $f(x)$ near a known point $f(a)$:

$$f(x) \approx f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a) + \frac{1}{3!}(x-a)^3 f^{(3)}(a) + \dots + \frac{1}{n!}(x-a)^n f^{(n)}(a) \quad (1)$$

This we refer to as a Taylor approximation up to order n . This is how our expansion of $f(x)$ about $f(a)$ looks for a uni-variable function, but what about multivariate systems? An example for a two dimensional system would mean formulating our expansion of $f(x, y)$ about a point (a, b) which looks like:

$$f(x, y) = f(a, b) + (x-a)f_x(a, b) + (y-b)f_y(a, b) + \frac{1}{2!}[(x-a)^2 f_{xx}(a, b) + 2(x-a)(y-b)f_{xy}(a, b) + (y-b)^2 f_{yy}(a, b)] + \text{H.o.T} \quad (2)$$

This we can then write in a more general setting for a system of N variables by using our gradient and Hessian operators:

$$f(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x} - \mathbf{a})^T \nabla f(\mathbf{a}) + \frac{1}{2}(\mathbf{x} - \mathbf{a})^T H(\mathbf{a})(\mathbf{x} - \mathbf{a}) + \text{H.o.T} \quad (3)$$

With these expressions, we now have the base level of knowledge required to start looking at approximating derivatives. Initially, we looked at three approximations for the first derivative of a function $f(x)$ as well as an approximation for the second order derivative. The expressions for these are given in Table 1.

Table 1: Expressions and the associated order of error for first and second order derivative approximations.

Approximation	Expression	Order Error
1st order forward difference	$f'(x) \approx \frac{f(x+h) - f(x)}{h}$	$O(h)$
1st order backward difference	$f'(x) \approx \frac{f(x) - f(x-h)}{h}$	$O(h)$
1st order centred difference	$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$	$O(h^2)$
2nd order difference	$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$	$O(h^2)$

From here, we investigated possible techniques for finding higher order derivatives and the value of leading order error terms. In particular an example using the **Method of Undetermined Coefficients** was given in W2L1. Additionally, the natural extension for partial derivatives was given, e.g.

$$f_x(x, y) \approx \frac{f(x+h, y) - f(x-h, y)}{2h}. \quad (4)$$

The next topic introduced was *root finding* in the real domain, \mathcal{R}^n . Namely, this means we are wanting to solve the expression $f(x) = 0$, for which Newton's method provides a nice follow on from Taylor series expansions. In order to use Newton's method, we first make an initial guess, x_0 , and then iterate using,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \quad (5)$$

Here it is clear that the derivative of the function cannot be zero at x_n , and we also need to compute the derivative (note that this can be done with the approximations discussed if an analytical form cannot be found). When considering this method, be sure to understand its limitations and times when it can go unstable!

The method can be extended to solve algebraic systems as well by making use of the Jacobian, $\mathcal{J}(\mathbf{x})$ and the update rule,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathcal{J}^{-1}(\mathbf{x}_n) \cdot F(\mathbf{x}_n), \quad (6)$$

where $F(\mathbf{x})$ is the vector of functions, $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x}))^T$.

The issue with this method comes from the computational expense of computing the inverse of the Jacobian.

A further extension of Newton's method can be used to solve for maxima and minima of functions, namely we would be solving for $\nabla f = 0$, rather than $f(\mathbf{x}) = 0$. For this, the Newton update would become:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathcal{H}(f(\mathbf{x}_n))]^{-1} \cdot f(\mathbf{x}_n), \quad (7)$$

where \mathcal{H} is the Hessian matrix. To then classify the result, we use the eigenvalues of the Hessian matrix,

$$\text{if } \begin{cases} \lambda_i > 0 & \forall i, \Rightarrow \text{minimum} \\ \lambda_i < 0 & \forall i, \Rightarrow \text{maximum} \\ \lambda_i \neq 0 & \forall i \text{ but signs are not consistent, } \Rightarrow \text{saddle} \\ \lambda_i = 0 & \forall i, \Rightarrow \text{inconclusive.} \end{cases} \quad (8)$$

The method of steepest descent provides another means for finding local minima of a function $f(x)$. In this approach, we start from an initial guess, \mathbf{x}_0 , and move in the direction of steepest descent (as the name may suggest!). The update rule for this method is given as:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \nabla f \quad (9)$$

Here we compute $\mathbf{x}_{n+1} = \mathbf{x}_{n+1}(\lambda)$ and then substitute this into $f^* = f(\mathbf{x}_{n+1}(\lambda))$. From here, we solve λ for,

$$\frac{d}{d\lambda} f^* = 0, \quad (10)$$

and use this to solve for \mathbf{x}_{n+1} .

1.2 With that out of the way, Week 4 content - *Integration!*

The first technique for numerical integration that we introduce is the *Trapezoidal rule*, this consists of dividing the domain up into Trapezoids and then summing the area of these,

$$\int_a^b f(x) dx \approx h \left[0.5(f_0 + f_N) + \sum_{i=1}^{N-1} f_i \right]. \quad (11)$$

In order to analyse the error, we can look at a single trapezoid and use a Taylor expansion then compare this to the exact integral. From this, we obtain an error of $O(h^3)$, but be a little careful here as this is the associated error for a single trapezoid. Summing these over the entire domain gives the global error as $O(h^2)$.

To improve on the accuracy of the Trapezoidal rule, we can also use a local quadratic approximation for the function and this is called *Simpson's rule*. This means we need three points for the approximation, so locally becomes:

$$\int_{x_i}^{x_{i+2}} f(x) dx \approx \frac{h}{3} (f_i + 4f_{i+1} + f_{i+2}) + O(h^5). \quad (12)$$

Then again, summing these sub intervals over the full domain increases the order of error globally to, $O(h^4)$.

The question now comes, is there a smart way to choose the step size h ? This is where *adaptive quadrature* comes into play. This technique allows us to have the computer determine whether finer resolution is necessary at certain points in our domain. For example, with the Trapezoidal rule, we can compare the integral for a single trapezoid with that of the same trapezoid computed with 2 half intervals. The error here can effectively be compared to a tolerance to determine if sufficient resolution has been used.