1 Math Review

A vector-valued function is a function that takes in n input variables and puts those input variables into component scalar functions. Example:

$$f(t, v) = f_1(t, v)\hat{x} + f_2(t, v)\hat{y} + f_3(t, v)\hat{z}$$

There can be as many t, v as you want. This is basically just a vector field (they're basically the same thing for me at least), a (3D) plane where every point is a vector (assuming no domain restrictions). When you plug in real actual numbers into the vector function (co-domain), the values are put into the scalar functions and each scalar function computation is separate. You end up with a vector.

A scalar-valued function is something just like

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$$

where the output of the function is just a scalar.

You take the gradient of a scalar-valued function. It doesn't really make sense to take the gradient of a vector-valued function. The gradient is just all of the partial derivatives of the scalar function put into a vector. The gradient of a scalar-valued function will give you a vector-valued function. So, the gradient at a single point is just like evaluating a regular vector-valued function which I described above: you just plug the values into each component scalar function and you end up with a vector. However, in this case, because the components of the vector function are the partial derivatives of some other function, the vector that you end up with when you evaluate the gradient at a certain point is the slope of the tangent line in the x direction, y direction, and z direction. Pretty cool, huh!

2 Computational Differentiation

2.1 Numerical Differentiation

Bad, slow. Numerical differentiation is when you just

$$\frac{f(x+h) - f(x)}{h}$$

and choose a really small h. Limited by the precision of h.

2.2 Algorithmic/Automatic Differentiation

Forward, backward, and midpoint method. Basically the idea that you multiply some matrix D by a vector which represents the function you are differentiating. Each row of the vector is the value of the function at that point. A greater

N means more precision because now you're splitting up the range more and getting more and more accurate values of the function at each point.

$$\frac{df}{dx} = \begin{bmatrix} d_{00} & \cdots & d_{0N} \\ \vdots & \ddots & \vdots \\ d_{N0} & \cdots & d_{NN} \end{bmatrix} \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_N) \end{bmatrix} \text{ as } N \to \infty$$

2.3 An Extension to Multiple Dimensions

So, now we have this new structure $\mathbf{D}$ that lets us differentiate a function. Just to be super clear,

$$f' = \mathbf{D}f = \frac{df}{dx}$$

Surely, we can define $f : \mathbb{R}^n \to \mathbb{R}$. Let's use an example in $\mathbb{R}^2$ and define $f(t, v)$.

$$\mathbf{D}_t f = \frac{\partial f(t,v)}{\partial t} = f_t \text{ and } \mathbf{D}_v f = \frac{\partial f(t,v)}{\partial v} = f_v$$

Therefore,

$$\nabla f = \begin{bmatrix} \mathbf{D}_t f \\ \mathbf{D}_v f \end{bmatrix}$$

But what does this really mean? Let's be more explicit with an example where $N = 4$.

$$\mathbf{D_t} f = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} f(t,v)_0 \\ f(t,v)_1 \\ f(t,v)_2 \\ f(t,v)_3 \end{bmatrix}$$

Now I'm confused. Because numerically, you need to supply some actual t and v. Should t and v be over the same range?