Letters and Notes

Exploring Alternate Notations for Partial Differentials

Jonathan Bartlett

DOI: 10.33014/issn.2640-5652.1.2.bartlett.2

Recent work has shown that differentials can be made more algebraically manipulable if the notation is improved (Bartlett and Khurshudyan, 2019). Can this also be the case with the Jacobian notation for partial differentials (i.e., $\frac{\partial y}{\partial x}$)?

The typical counterexample to why partial differentials could not be algebraically manipulable comes from an example similar to the following. Take a three-variable equation, such as $y = x^2 z$. This equation has six partial derivatives. For this example, we will note that $\frac{\partial y}{\partial x} = 2xz$ and $\frac{\partial y}{\partial z} = x^2$. If partial differentials were algebraically manipulable, we should be able to find $\frac{\partial z}{\partial x}$ by simple algebraic manipulation:

$$\frac{\partial z}{\partial x} = \frac{\frac{\partial y}{\partial x}}{\frac{\partial y}{\partial z}} = \frac{2xz}{x^2} = \frac{2z}{x}$$

However, actually calculating the partial derivative of z with respect to x directly yields $-\frac{2z}{x}$.

As was the case in Bartlett and Khurshudyan (2019), the reason for the contradiction is not a failure of the power of differentials, but merely in the notation. To understand this, let us look at where partial differentials come from. Taking our original example, let us simply take the total differential.

$$\begin{aligned} y &= x^2 z \\ \mathrm{d}(y) &= \mathrm{d}(x^2 z) \\ \mathrm{d}y &= x^2\,\mathrm{d}z + 2xz\,\mathrm{d}x. \end{aligned}$$

Let's say we wanted the partial derivative $\frac{\partial y}{\partial x}$. A partial derivative between two variables means that no variation is allowed to happen in the other variables. In this case, the other variable is z. If no variation happens in z, that means that $\mathrm{d}z$ must be zero. Therefore, we set $\mathrm{d}z = 0$ and solve for the ratio of the other differentials, like this:

$$\begin{aligned} \partial y &= x^2\,(0) + 2xz\,\partial x \\ \partial y &= 2xz\,\partial x \\ \frac{\partial y}{\partial x} &= 2xz \end{aligned}$$

Now, when we solve for $\frac{\partial y}{\partial z}$, we do the *same* process, but set a *different* differential to zero (i.e., $\mathrm{d}x$). Note that both of these derivatives involve ∂y, but they come from two *different* modifications to the original equation. Therefore, the ∂y in $\frac{\partial y}{\partial z}$ is *not* the same ∂y that is in $\frac{\partial y}{\partial x}$. Indeed, the bottom differential gives us additional information about *which* ∂y is being discussed. That is why the notation $\frac{\partial y}{\partial x}$ cannot be separated—information about the numerator is contained in the denominator, and is therefore lost when the fraction is split.

To resolve this situation, we merely need a notation that allows us to be more specific about which partial differentials we are dealing with. Here we will describe two possible approaches, each with different implications. I don't think either of these are the "best" system, and I hope that this discussion sparks additional ideas which combines the advantages of each system. It is also possible that the notation chosen to represent partial differentials is based on your own goals of how to manipulate them.

The two systems will be based on subscripting our partial differentials. In **system 1** (for lack of a better name), we will subscript the partial differentials with the variable that was allowed to freely move. In the **system 2**, we will subscript the partial differentials with the variables that were forced to not move.

So, in **system 1**, instead of $\frac{\partial y}{\partial x}$, we will now modify the notation to include the variables which were allowed to freely move. Therefore, the partial derivative of y with respect to x will be $\frac{\partial_x y}{\partial_x x}$. Likewise, the partial derivative of y with respect to z will be $\frac{\partial_z y}{\partial_z z}$. This notation can be further simplified by noting that, if a variable is allowed to freely vary, that is the same as being a total differential. Therefore, $\partial_x x = \mathrm{d}x$. This simplifies our derivatives to $\frac{\partial_x y}{\mathrm{d}x}$ and $\frac{\partial_z y}{\mathrm{d}z}$.

This also means that a total derivative of a dependent variable is simply the sum of its partial differentials. $\mathrm{d}y = \partial_x y + \partial_z y$ In fact, we can actually carry pieces of the par-

tial derivative around in the subscripts, by saying $\partial_{x+z} y = \partial_x y + \partial_z y$. This can have benefit in large, multivariable equations, as the subscript can simply be added to as additional partials are added in. When *all* of the variables are included, then the partial differential is the same as the total differential.

The drawback to **system 1** is that it only fully works when y is a dependent variable, and all other variables are independent variables. If the other variables are have dependencies among them, the system breaks down. For instance, if $x = f(z)$, then $\partial_x x$ is not equal to $\mathrm{d}x$, and $\mathrm{d}y$ is not equal to the sum of its partials, depending on the particular formula those partials come from.

The other system, **system 2**, takes this into account by, instead of subscripting which variables are allowed to freely vary, subscripts which differentials were forcibly set to zero. This system makes fewer requirements of the equation itself (because it specifies what *we* are doing to the equation), but also lends itself, as far as I can currently tell, to fewer simplification mechanisms.

Therefore, the partial derivative of y with respect to x (where $\mathrm{d}z = 0$) would be specified as $\frac{\partial_z y}{\partial_z x}$. This can get unweildy in equations with a large number of variables, as you would need to subscript every variable whose differential which was set to zero.

In any case, both **system 1** and **system 2** allow for extending algebraic manipulability to partial differentials. The advantage of **system 1** is that it has rules for interchanging partial and total differentials, but the disadvantage that you must know *a priori* which variables are dependent and independent. The advantage of **system 2** is that you do not need the knowledge of which variables are dependent and independent, but building up total differentials from partial ones is more complicated.

Unfortunately, the notation for **system 1** and **system 2** are identical, so it would be confusing to use them both. Another alternative would be to use set notation. So, using **system 1**, you could say $\frac{\partial_{\mathrm{d}\in x} y}{\partial_{\mathrm{d}\in x} x}$, while **system 2** would say, $\frac{\partial_{\mathrm{d}\notin z} y}{\partial_{\mathrm{d}\notin z} x}$. Basically, this uses set notation to say which elements were allowed to be modified. There are drawbacks here, too, as it makes it look like **system 1** and **system 2** are combinable, but I am not sure that they are. Additionally, the notation itself is quite unweildy.

The primary point of this exercise is to show that (a) the problem of the algebraic manipulability of partial differentials is primarily a notational problem, and (b) to start the conversation about what notation might possibly replace the current notation, and the benefits and drawbacks of each option.

Bartlett, J L and A Zh Khurshudyan (2019). "Extending the Algebraic Manipulability of Differentials". In: *Dynamics of Continuous, Discrete and Impulsive Systems, Series A: Mathematical Analysis* 26.3, pp. 217–230.

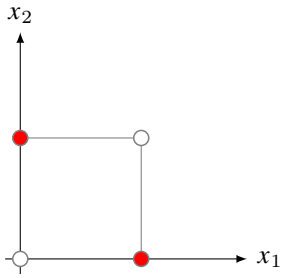
The Unlearnable Checkerboard Pattern

Eric Holloway

DOI: 10.33014/issn.2640-5652.1.2.holloway.1

In the history of machine learning, a famous upset was the disproving of the perceptron's generality by use of the XOR problem. There is no linear separator than can classify the points with 100% accuracy (Russell and Norvig, 2009, pg. 741).

Figure 1: Difficulty of Linear Separation in an XOR



In response, the multilayer perceptron was invented, which eventually became the modern day neural network.

However, the neural network, and other machine learning paradigms, still have a problem learning a tiled XOR pattern called the checkerboard, see Figure 3.

The checkerboard classifications can be generated using a logic expression. To generate the classifications from $2N$ variables, we use the logic expression $x_1 \oplus x_{N+1}$.

Here is an example with 4 variables, x_1, x_2, x_3 and x_4. The logic expression is $x_1 \oplus x_3$.