[SQUEAKING]

[RUSTLING]

[CLICKING]

ALAN EDELMAN: All right. Well, welcome to the second lecture for this IAP class on matrix calculus. Steven, do you want to say anything before I start on nonlinear versus linear maps? So Philip's going to help out right away. So here's a picture of Philip, the corgi. He's right over there, in case you haven't seen him yet. And you folks could actually grab your favorite photo. You don't have to use a corgi. You could have any photo you like.

So here in Julia I'm just uploading this picture of Philip. And I've got a few different transformations of a corgi. And the reason why I'm showing this to you, it's because it's come to my attention that not everybody has a fairly good visual intuition as to the difference between a linear and a nonlinear map, or even a map. Here, so let's go ahead. Let me just ask a real quick simple question for starters. So here I'm just rotating. Right? So I'm rotating Philip. He's going to get dizzy.

OK. No, he's not going to get dizzy. No animal is harmed. So, all right. If I'm talking about maps from vectors to vectors from Rn-- it's usually Rn as in Nancy to Rm as in Mary. OK? What are m and n for this transformation? That's an easy question. Not a trick question. So what are m and n for here?

So think of this as a transformation from the original upright picture to the rotated picture. What are m and n? That's an easy one. Oh, gosh. No, I'm not even thinking that way. That's too fancy. Not that you're-- in some deep sense you're probably right, but for the purposes of this class you're way off. So let's try again. So m and n are the dimensions of the vectors. Anybody? I thought this was easy. The fact that it's not easy is telling me something.

STEVEN JOHNSON: I think the question is, is it the transformation of the image you're talking about or the transformation of the coordinates of the image that you're talking about?

ALAN EDELMAN: Does that make a difference? Right.

STEVEN JOHNSON: Is it image in and image out or coordinate in and coordinate out? Which one are you talking about?

ALAN EDELMAN: Does it matter? Aren't they inverses of each other? Answer either question that Steven just said. What do you guys think? Or is the question just not clear? Yeah, it's from R2 to R2. That's whatever. I mean, don't overthink the question. Right? right I've got this Julia. I'm going to hold up my phone.

Steven, maybe you can see. I've got this Julia sticker. Right? It's in the plane, this plane, this two dimensional plane of the phone. Right? And this rotation is a mapping from R2 to R2. I mean, if you want to focus on-- I suppose if I have the origin in the middle of the phone and I rotate, this is a linear map. Right?

STEVEN JOHNSON: Because Alan, you could also talk about the transformation that takes the image of the corgi in and gives the rotated image of the corgi out.

ALAN EDELMAN: Isn't that what I was-- that was exactly that--

STEVEN JOHNSON: But then the vector space is the number of pixels in and the number of pixels out.

ALAN EDELMAN: All right. This is going down a rat hole that I don't want to go-- I don't agree. But I think-- yeah. All I really wanted to do-- all right, I'm going to go to the blackboard. I'm going to go to the blackboard then. And let's turn on the lights on the blackboard here.

All right. So for years in linear algebra classes-- and I'm sure you've seen this-- people will write R2. Right? And then they would put down-- if it's linear algebra, you'd have a matrix A. And people would-- you would draw a vector here, like a vector x. OK? And then you would have the image plane, also R2 in this case. Right? And x goes to ax. Right? And if you have another vector y, that might go to ay. You've all seen this in a linear algebra class, right?

OK. And if anybody is familiar with the cover of Gil Strang's book-- which maybe I should just bring up while we're at it. So if you're familiar with the cover of Strang's linear algebra book, which is the textbook that's used right here at MIT and all over the world, one of the versions of the book has these houses, this one right here.

OK, so maybe you've seen this version. Right here with the-- and this is meant to indicate the transformation of a house. Right? And so you have a house and some windows and a chimney or something. I don't know what you have. And the image-- let's say-- I mean, just to be clear, if this house was a square and this was your everyday general matrix, what would the image look like over here? Just generally speaking.

If you transform a square with a matrix, what does it look like? Just what shape? It's a word you learned in high school or elementary school. It's not a hard question. It's a parallelogram of some kind, right? I don't know where it'll be, but it'll be some sort of-- the square will become a parallelogram of some sort. I don't know where it'll be. The square window will also be a parallelogram with parallel parts. That little chimney or I don't know what that is will get skewed in some way as well, right?

This is what happens when you take every point. I mean, I didn't want to get into pixels. I mean, just every point in the plane is being transformed to another point in the plane, right? This is what we would call a linear map from R2 to R2. Linear map from R2 to R2. It emphasizes about a matrix. Not the fact that a matrix is just a two by two four elements. It emphasizes the fact that a matrix is something that transforms space. OK?

And that's all I wanted to say, and I thought that this was a basic element of linear algebra. I'm a little worried that just the way I asked the question wasn't clear to people. So if I had a picture of Philip over here-- which I could never draw a good picture of Philip. Here's a Minecraft Philip. It's a terrible Minecraft picture of Philip. Right? It would get transformed to some skewed version of Philip on the other side, and that's what a linear map is.

OK. So am I saying things you don't know? You all know this, right? OK. So with that understanding, let's go back, turn the lights down low again. So with that understanding, I just wanted to show you a few linear and nonlinear maps. Not to spend too much time on this, but maybe it's worth it.

Yeah. So a rotation is a linear map from R2 to R2. Is there still any confusion about pixels or any other nonsense? I just want to think of-- to be very, very clear, I'm just thinking of every point-- this is the center. His little black nose here is the center of the plane. And you can always draw a vector. I guess, oh, I could use Zoom to draw a vector. This is great. Let's draw a vector right here. So if I'm at the origin, every point, like the tip-- well, I can't draw a straight line, but that's my fault.

But this is meant to be a vector right to the tip of his ear. And then that vector will stay where it is, but if I rotate the plane, say-- you know, let's not rotate that much. Let's rotate maybe 30 degrees. You see that his ear has moved 30 degrees. Oops, oops, oops. Everybody with me? Any questions? OK. Let's not belabor this point.

So the next thing I want you to do was-- the next thing I wanted to do-- let's just clear this-- is show you some other linear transformations. So instead of rotating is a little bit too simple, so another linear transformation is this hyperbolic rotate. So let me just switch to hyperbolic rotate. Yet again, forgive me. I just forgot my glasses today. So I'm kind of winging this a little bit. So hyperbolic rotate is a little less familiar to a lot of people.

So when you rotate-- just to say something also, I think, very obvious. Yeah. When I rotate it, you see the red circle? What happens to Philip's ears as I rotate? If I rotate, the ears follow the red circle. Right? They just move along the red circle. OK. Now I want you to now focus. I'm now doing hyperbolic rotate. I want you to focus on his eyes. You see the eyes are on a hyperbola? OK?

You'll notice that as you move, his eyes always actually-- this is what's called a hyperbolic rotation. And his eyes, even as the picture gets more and more skewed, so these squares become flatter and flatter parallelograms, you could still see-- I'll go backwards. You could still see that his eyes are always following a hyperbola as opposed to a rotation, an ordinary rotation where the red ears would follow a circle. Right?

So this is also a linear transformation in that literally if you draw a vector from his nose to any pixel and another vector from his nose to any pixel and you want to know, if you add those two vectors to create a third vector, then it will add in the image. OK? And now let me just show you a nonlinear-- some nonlinear transformations. So here's a nonlinear shear. So what this does is shears things, and you could-- it's already-- and so squares are no longer guaranteed to be parallelograms. In fact, they're not parallelograms.

It's much more of a-- here he is. Poor Philip is being folded up. And then there's this warp. I mean, we could put in any-- this is Julia. You could put in any function you want. So here's a warping, which is also nonlinear. So here's the original picture, and you can see different parts of the picture will move differently. And so just to make it clear what's going on with matrix calculus, everybody understands that this is a mapping from two vectors to two vectors, right? From R2 to R2. Right?

So if you form the Jacobian matrix, what's the size of that Jacobian matrix? A two by two Jacobian matrix. And the Jacobian matrix is different at every point in the plane, right? When you're linear, the two by two Jacobian is constant throughout the plane. It's actually-- like the picture on the board, it's just the matrix everywhere. But when you have a nonlinear map like this one, the two by two Jacobian depends on where you're starting.

And so if one could zoom-- I don't know if one can zoom. Yeah. If you were to zoom in, what it's saying is that a zoomed in picture-- I don't know if you'd be able to see it, that I'm trying to Chrome zoom in. But what a Jacobian is saying is that if you look in a small little patch, squares will become parallelograms. Circles will become ellipses. Right?

And they'll do it in a constant way. But as you move away from that small patch, it'll do it differently in different places. Is there any questions about what a map from R2 to R2 looks like? And of course, I can move my hands around and talk about maps from R3 to R3. I mean, you could easily imagine-- if we turn our heads, in effect, what we see in our eyes is a rotation of R3. Right?

That's a linear map as long as you fix on one point. But one could imagine nonlinear maps of R3. It's not something I could easily do with my hands. But you could-- right? And so that's R3 to R3. And of course, in general, we don't even have to have the same dimension going in and going out. It's Rn in, Rm out. OK? All right. That's all I wanted to say just to start things off to get people used to these maps. All right. Steven, should I turn it over to you?

STEVEN JOHNSON: Yeah. Sounds good. So I want to talk again today about this generalization of the notion of a derivative or reinterpretation or revisiting of a derivative as a linear operator. So if you have any function-- remember what we said last time. So if you have any function f of x and you change the input by a little amount, of course, there's a little change in the output.

And if the change in the input is very small, then we can approximate the change in the output by something that's linear in the input where it's f. And so this is going to be-- basically it's linear in the input plus higher order terms, but it'll be convenient just to drop these higher order terms and the notation will be this differential notation. So we think of this as a really small infinitesimal change in an arbitrary direction if x is a vector.

And so when we take f of x plus d minus f of x, we're just going to drop implicitly any term that's higher order than linear, has a dx squared or a dx to the 1.5 or anything like that. And so the definition of the derivative f prime is going to be the linear operator acting on dx, the change in the input that gives you the change in the output. And so that gives us-- it's equivalent to the 18.01 definition of a derivative, for if f is a scalar function.

A linear operator is just a number, the slope, that multiplies by the change in x to give you the change in y. But it also works for multivariable functions. So if the input x is a vector and the output is a scalar, then this is a linear operator. The derivative is a linear operator acting on a vector. dx is an arbitrary change in an arbitrary direction in this input vector that gives us the change in f, which is a scalar.

And a linear operator on a vector that gives a scalar is a row vector or a one row matrix or a co-vector or linear form. There's all sorts of fancy names for it, but it's basically-- it's a dot product with something. Right? That's the only way to get a vector and do a linear operation and get a scalar. And the thing we're taking with the dot product with we call the gradient. So the gradient, this df is the gradient dot dx, or equivalently f prime is the transpose of the gradient.

It's the row vector, the linear operator that you act that row vector on a column vector on a dx. It gives you the df. Right? This works for ordinary column vectors, and we're going to think of other vector spaces very soon. OK? And then the thing I just started at the very end of the lecture was suppose-- so this is not 18.06 revisited. This is 18.02 revisited. This is still 18.02. Oops. Black.

The other kind of function you deal with in 18.02 are functions that take vectors in and vectors out. And so for example in an integral and multi-dimensional integral, if you change coordinates, you have x and y and you get some new x and y, that's a function that takes a vector in and a vector out. May or may not be linear. So that's where the function-- the inputs, I'm going to say the inputs live in Rn and the outputs live in Rm. So maybe there's a different number of inputs and outputs.

And then when we ask, same thing. If we change the input vector by a little bit, then there's some change in the output vector which we call df. And the change in the output vector has m components. The change in the input vector has n components in some arbitrary direction, and f prime is the linear operator that goes from the change in the input, n components, to a change in the output, m components.

And the only kind-- and the way we typically describe such a linear operator is an m by n matrix. Right? So that's what you would multiply by a vector with n components to get a vector with components. And we call that matrix, that m by n matrix, the Jacobian J. And in fact, you can also think of it in terms of the entries of J.

If we ask what the entry Jij is, this notation means row i, column j. Right? This is going to be equivalent to what you learned, hopefully, in 18.02 although not every version of 1802, apparently. It's the same thing as partial fi, the derivative of the i-th output with respect to the j-th input. I always used to get these confused. Is it, which one's the rows, which one's the columns. The dfi, dxj, dfj, dxi.

And in 18.02, people just take the determinant of this matrix for integration, and then it doesn't matter. Because you transpose the matrix, it doesn't change the determinant. But as a linear operator, it really matters. The rows-- here we can even write this. Let me just do it in blue. If you think of a matrix as a linear operator, the rows are the outputs which are the df's, and the columns are the inputs, which are the dx's.

So it really matters that the rows are the different f's and the columns are different x's. Let me just do an example, since we were doing-- this is to m equals n equals 2. And so m equals n equals 2, since we were just doing transformations of the plane. So we're going from inputs in R2 and then f is taking us to outputs also in R2. So we're just mapping the plane to the plane, but it maybe it's not a linear transformation. All right?

So if we thought of this in 18.02 style, component by component, then we have-- f of x, we can think of it as two functions, f1 of x and f2 of x. And x, we can think of it as two components, x1, x2. Right? And then if you want the change in f-- let's put a vector here to keep track of what things are vectors, what things are scalars.

The change in f, my claim is going to be the Jacobian. And the Jacobian is now going to be our two by two Jacobian. And I guess maybe I should put outputs in blue here. All right? And the outputs are the different rows. So there's df1, df1, df1, df2, df2. And the inputs are the different columns, dx1, dx2, dx1, dx2.

And then if we multiply this by any change-- so you change the input by any vector in dx in any direction. So this is our dx. Right? You can see that it works out. Right? So I don't know how I would really write this out, but if we multiply rows times columns, you get df1, dx1. Let me do it, actually. So that's-- I need a little more space, maybe.

All right. What's the first row? Row times column. Let me keep my color scheme. df1, df1, df2, df2. I think this is it, right? If I do row times column, I get partial f1-- partial f1 partial x1 times sx1, which is this. Partial f2, partial x2 times dx2, which is this. And the same thing for the other row, it's just with f2. Right? Oops, and I forgot a-- I forgot a plus sign. I have to add them. All right?

And you can see. So a component by component it makes sense, right? If you want the change in f1, well, you take-- its change, the rate of-- its slope with respect to x1. You multiply by the change in x1. And you also have to add the slope of f1 with x2 and multiply by the change in x2 and so forth. So really, all the components match up, but this is a really-- this gets awkward very, very quickly to write things out component-wide like this. Right?

It's much easier to just say, this is a linear-- think of it as a whole. It's a linear operator acting on the change of x, and that, if we want, we can write it down as a Jacobian matrix in this m by n matrix. So I want to do a couple of examples. We'll do another example. I think Alan did this the other day, but let me just-- it doesn't hurt to do it again. So suppose f of x is just a matrix times x. Suppose this is a linear. Right? And so here I'm just going to say, this is a constant matrix.

ALAN EDELMAN: Like the ordinary rotation and the hyperbolic rotation would be two examples of what Steven's doing now.

STEVEN JOHNSON: Exactly. So this could be a rotation matrix or something like that if this is two by two. But it doesn't have to be two by two. It doesn't even have to be a square, right? This could be in output's in Rm and the input's in-- so the outputs are in Rm and the inputs are in Rn. And therefore the A matrix is an m by n matrix in general. It could be, for example, two by two. Right?

And again, let's just do this the slow way. But still much faster than 18.02. So the 18.02 way would be to write this out as in components. So you take ai-- this is fi is aij sum over-- let me keep my color scheme. aij xj and sum over j equals 1 to n. And then ugh. All right? I mean, you could write it out in terms of all the components and take derivatives that way. It's not so bad, but it starts to become really awkward really quickly. Right? If we just think of--

ALAN EDELMAN: --everybody to think of indices as ugh.

STEVEN JOHNSON: Yeah.

ALAN EDELMAN: Often. Not always. Sometimes they're useful, but usually they could be not used.

STEVEN JOHNSON: Yeah. And another way of thinking about it is as soon as you start writing things in terms of indices, everything is very coordinate system dependent. So whereas if you think of it just generically, a rotation is an operator that rotates things by 90 degrees. Right? You don't have to have a coordinate system for that operator. Anyway, so what's the derivative?

Well, let's just take-- pretty soon we'll have a product rule, but here you just do it the slow way again. It's not even that slow, you take f. You take your x, and you add a little change minus f of x. And what's that? We can just plug it in and drop nonlinear terms, except in this case there won't be any nonlinear terms. This will be f-- what's f of x plus delta x? It's ax plus a dx. Right?

Because this is a linear operation, so I can-- I guess I skipped a step. a times x plus x is ax plus adx and minus f of x is ax, and then these two terms cancel. And so what we get is just adx, and this has to equal f prime of x dx. So you can see that if we just did a little compare of these, f prime is just a. It's just a linear operator a. So f prime, f prime of x is just a in this case, which that's our Jacobian.

ALAN EDELMAN: And just to say the obvious, it doesn't depend on x at all. Everywhere in n-dimensional space it's the same a.

STEVEN JOHNSON: Yeah.

ALAN EDELMAN: No matter where you are in Rn.

STEVEN JOHNSON: Yeah. Because in this case, it's like, again, if you're in 18.01 and I have a linear function 3x plus f of x equals 3x, the slope is just 3 independent of x in that case. That's not true of all functions. Right? If the function is nonlinear, like 3x squared, the slope is 6x. It depends on x. But if it's a linear function, the slope doesn't depend on x. Same thing happens here, and in homework I'll ask you to do an even slightly more general version of that maybe. OK?

But this is-- again, writing this out, I think it's still a lot easier than component by component. Right? Doing this versus doing that and taking the derivative partial fi, partial xj and seeing that's aij, and then realizing that, oh, that means that the whole thing is just a. But it's still a little bit cumbersome, especially when it gets to more complicated functions like this, so it's nice to have some rules. Right?

So just like when you learned the derivatives in 18.01, you don't use the definition of a derivative at some limiting procedure for every time. You learn some product rules and power rules and sum rules and those kinds of things, so we can do the same kind of thing. Let's do an easy one. Let's do a sum rule. Suppose you have a function f of x is g of x plus h of x. And let me draw out my vectors. You can understand that everything here can be a vector. Right? The x's can be a vector and the f's and g's and h's can be vectors.

In fact, in arbitrary vector spaces we'll see very soon. Right? Then if I want df, that's just going to be-- just write it as dg plus dh. Or equivalently, this is-- this is f prime of x dx equals g prime of x dx plus h prime of x dx. So f prime equals g prime plus h prime. Right? Just the obvious, some rule. I don't know. I don't know if you want me to derive this. I mean, again, it just comes from the definition. If I take f of x plus dx minus f of x, addition is linear. It's g of x plus dx minus gx plus h of x dx minus dx.

ALAN EDELMAN: I don't think anybody in the room doubts it. You don't actually have to derive it.

STEVEN JOHNSON: Yeah. So let me do the product rule. All right? So this one starts to be a little bit. I should do this on-- well, I'll write it out first. Suppose you have-- and I'm just going to write-- do I need to write the f? Sure, I'll write f of x equals g of x times h of x. So these have to be things you can multiply, obviously. Right?

These could be two numbers or they could be two matrices of the right size. They can't be two column vectors, at least not with the ordinary. I'm just doing multiplication here. Maybe I should do element-wise multiplication in a minute, but let me just do multiplication. OK. So then df is going to be just dg times h plus g times dh.

ALAN EDELMAN: And I guess the point is, whenever gh makes sense, whether it's element-wise products or matrix multiply, if it makes sense--

[INTERPOSING VOICES]

STEVEN JOHNSON: Yeah. So if you think about what this means in terms of f prime, this f prime of x dx is going to equal g prime of x dx times h plus g of x plus h of x plus g of x times h prime of x dx. I cannot necessarily write f prime equals g prime h plus h prime g because I can't commute. I'm running out of space here. Why can't I do that?

Because I can't necessarily move the x to the side here. All right? So g prime dx h is not equal to g prime h times d in general. If these are numbers, obviously this works. But now that these could be some other kind of object matrices or something like that, this is not going to work in general. OK, so let's-- and maybe that's-- let's derive this, just to-- we already kind of derived it, but let's do it in general. This is derivation. Right?

So f is in gh. So what is df? Just use our definition. df is going to be f of x plus dx minus f of x, and f was g times h. This is g of x plus dx times h of x plus dx minus g of x h of x. And then this is g of x plus dx. We said by the definition of the derivative, that's g of x plus g prime of x, whatever that linear operator is, times dx times h of x plus dx-- h of x plus dx. Same thing.

By the definition of the derivative, that's h of x plus h prime of x minus g of x h of x. And if we just write out all those terms-- and again, these might be matrices or something. I can't change the order. But I'm assuming that if this is a multiplication operator, it could be element-wise or it could be just matrix multiplication. Anything that satisfies the distributive law. I can write this as gx h of x. That's this term times this term.

There's also this term times this term, this term times this term, and this term times this term. This term times this term. Sorry. So I get plus g prime x dx times-- which I could also just write as dg. This is the same thing as dg. This is the same thing as gh. Maybe I should have done that. It's a little shorter to write.

This is dg times h plus I have a g times dh plus I have dg times dh. But that term we're going to ignore. This is higher order. All right? That's gone. If d is arbitrarily small, that's infinitesimal. And then we still have the minus g of x, h of x term here. And that's it. Right? So we're done.

ALAN EDELMAN: And this is really nothing but a copy of what you did in the first month of calculus when you first learned it except that we are just extending it to vectors and matrices and making sure that we don't mess the order up.

STEVEN JOHNSON: Yeah.

ALAN EDELMAN: Otherwise it's exactly what you did in the early days of calculus. I'm sure you all remember.

STEVEN JOHNSON: Yeah. And let me just do a couple examples. Suppose the one we just had. f of x equals ax. Right? Then df is equal to da times x plus a times dx. But this is zero. Since a is constant, a is independent. It is independent of x. So remember, you always have to keep track of what are the inputs, right? So here df-- f is a function whose input, in this case, is x. So df really means, f of x plus dx. f of x plus dx minus f of x. So since a doesn't change with x, that's zero, so then I'm done. Right? This is, again, the same term I got before.

ALAN EDELMAN: Let me ask the class and not you, Steven. What zero is this? Right? I think that might be useful to think about. Can someone tell me exactly which zero--

STEVEN JOHNSON: Shape of zero, yeah.

ALAN EDELMAN: Shape of zero.

AUDIENCE: [INAUDIBLE]

ALAN EDELMAN: Sorry?

AUDIENCE: M dimensional column vector.

ALAN EDELMAN: Is it m dimensional column vector? No, I don't think so.

AUDIENCE: It is m.

ALAN EDELMAN: Yeah. So what is da?

AUDIENCE: A square matrix. An m cross n matrix of zero.

ALAN EDELMAN: Yeah. It's an m by n matrix of zeroes. So that infinite red zero that Steven drew is really the m by n zero matrix. Right? In fact, he could have written da equals 0dx. Might have pointed that out. But it's so easy to quickly look at that zero and lose sight of the fact that that's a m by n zero matrix. Which if you multiply by any dx that's n little perturbations, you get m0's coming back out.

STEVEN JOHNSON: Yeah. So let me do another one, one that I did before. So did x transpose ax, I believe. Right? So now if I do df I get dx transpose. This confuses people, by the way, a little bit. Well, actually let me do-- why does dx transpose equal dx transpose? Because it's just dx-- dx transpose is really x plus dx transpose minus x transpose. But that's equal to x transpose plus dx transpose minus x transpose. So that's dx transpose. Right?

ALAN EDELMAN: I wouldn't have thought of it that way, but of course it's right.

STEVEN JOHNSON: Yeah.

ALAN EDELMAN: To me it's just you just take something as a column and you make it a row and you perturb. Either way, it's the same thing. But formally you're quite right, Steven.

STEVEN JOHNSON: Yeah.

[INTERPOSING VOICES]

And so in the homework I want to get you to think of this in terms of linearity. Right? The transpose is-- well, maybe I'm kind of giving away, though. But transpose is an example of a linear operation on a vector. And it's one that's kind of annoying to write down as a matrix because it's not really multiplying x by a matrix unless you reinterpret the output as a vector and the input as a vector in some way. So anyway. So yeah, so this is-- let me just go back. The product rule-- I guess I only showed you the product rule for two terms, but it's the same thing for three terms. So there's df.

ALAN EDELMAN: You know, you could talk about the product rule for three terms, or you could think of this as two terms and use the result that's on the top of your screen.

STEVEN JOHNSON: Yeah. Exactly. I could just use--

ALAN EDELMAN: --now Right.

STEVEN JOHNSON: Yeah. But let me just do it do it for three terms. But yeah, I could use two terms and then you write it as-- let's do it with two terms. So there's the x transpose times dax. Right? And that's equal to dx transpose ax.

ALAN EDELMAN: So in effect, you're putting a parentheses around the-- it's x transpose times left parenthesis ax right parenthesis [INAUDIBLE]. Right? You see what I'm saying?

STEVEN JOHNSON: Yes.

ALAN EDELMAN: The parenthesis itself is thought of as extra-- well, even-- yeah, yeah. That's what I'm thinking of.

STEVEN JOHNSON: You're thinking of that, yes. And of course, I can put parentheses anywhere I want in these kinds of things.

ALAN EDELMAN: Of course you could, but that's what you're focusing on this way, right?

STEVEN JOHNSON: Yeah. And dax is product rule again, but I just did that, so it's adx. Right? And then we used the same trick before to-- this term here is a scalar, so I can just transpose it and get the dx on the right. And so that's equal to x transpose a plus a transpose dx. And then this was our f prime of x, which is the same thing as the gradient of f transposed. Right?

Which therefore means that the gradient of f is equal to the transpose of that, which is a plus a transpose x. So that's the product rule. Pretty straightforward. Right? We could do more complicated things. So we could do, for example-- actually, let me hold off on that for a second. Let's do the chain rule. So suppose we have f of x equals g of h of x.

So gh of x is a function, takes a vector in, gives you-- some x in, gives you some vector out, some vector space. Maybe numbers. Maybe column vectors. Maybe matrices pretty soon. And g takes whatever the output space of h is and gives you some other vector as output. All right? So for example, x could be a column vector, and then h of x takes a column vector and produces a matrix.

And then g of x takes a matrix and gives you a scalar, for example, like a determinant. OK? So they could be all different shapes. Right? And one of the things we talk about in linear algebra is it's always good to keep track of the shapes of things. Everything's not a number anymore and you need to keep your shapes lined up. OK? So then, what's the chain rule going to be?

It's going to be-- I'll just write it down first and we'll just see why. f prime of x. x is a linear operator acting on a dx. Right? And that's our-- and that's our df. We should maybe write that down. df is that. OK. And what do you do? It's going to be just like the chain rule for numbers. It's going to be g prime of h of x. That's a linear operator, and that's going to act on h prime of x, and that's a linear operator that's going to act on dx.

I guess I haven't-- we're not putting vector signs in there. Very often, I'll drop the f prime of x equals g prime of h of x, h prime of x. But when I write a product like this, this really means the composition. It means it's a product of two linear operators, of g prime and h prime. It means you do h prime first and then g prime, and this is very much not equal to h prime times g prime. Think of these, for example, as matrices, right? You can't change the order.

ALAN EDELMAN: Can I throw in something that's a foreshadowing of something that might come later? Just to set a stage. I want to ask the class, from ordinary calculus, just an application of the chain rule. Let's say you were taking just the derivative-- you all can do this. Derivative of sine of x squared. I want to ask, how many of you will first do the cosine? So it'll be cosine of x squared, and then in your mind will do the 2x?

And as opposed to, how many of you will do the 2x on the inside first and then do the cosine? So how many of you are cosine first people? I think that was everybody. Does anybody do the x squared first going into out? There are two people in the back who do it. But you see, I mean, of course, you could do it either way.

And just to say some words now that will come up later. But the first way is what eventually becomes forward mode automatic differentiation, if you've ever heard that term. And the second way is reverse mode automatic differentiation. Of course, for scalars it's all pretty simple, but when we get into this matrix calculus stuff it gets to be a whole big thing of machine learning and lots of other things-- technologies that underlie so much these days. Right?

But it really-- in its simplest, simplest form, it all boils down to, as Steven writes, you have the choice, in effect, of doing the h prime first or the g prime h first. Right? If you're a computer, you'll have to do it sequentially. Right? So one of the things will come first. I guess it doesn't have to be sequentially.

STEVEN JOHNSON: I'll show an example of that in a second. So I think it becomes more clear once you have three things.

ALAN EDELMAN: Yeah, OK. All right. I just wanted to set that thought up.

STEVEN JOHNSON: Yeah. Absolutely. It's really going to matter, left to right. When these are not scalars anymore, the left to right versus right to left is going to start to matter a lot.

ALAN EDELMAN: And also the order.

STEVEN JOHNSON: Yeah.

ALAN EDELMAN: Right?

STEVEN JOHNSON: So, yeah. I don't know if people want me to derive this again. We can just plug in the definitions and do f of x, take this plus dx minus g of h of x plus dx minus g of x is g of h of x. Sorry. And then--

[INTERPOSING VOICES]

--everything and just drop the dx's.

ALAN EDELMAN: Who wants the derivation and who wants to go straight to the application? Derivation?

STEVEN JOHNSON: Derivation.

ALAN EDELMAN: How many want to see the application? Like, five or six people.

STEVEN JOHNSON: OK. OK. So let me do an example of this just to-- suppose, to start with, that-- so the x, or inputs, live in Rn. So we have n component vectors as inputs. And then suppose that h of x-- h of x are our first function. Our inner function lives in our p. p sends vectors to vectors. Oops. What happened? OK.

And suppose our outer thing, our g of h, lives in our m. So if we do f of x. All right? Which is not going to be-- because remember, it's g of h of x. All right? What it does is it tend-- it sends things in Rn through to Rp via h. And then sends them via g to Rn. OK?

So it sends vectors to vectors. Vectors may be all the same size, maybe not. OK. And then what is our chain rule? Right? Is that f prime-- let's e blue. f prime of x. And so now this is-- we know this is a Jacobian matrix. This takes n inputs, m outputs. This is an m by n Jacobian. This is equal to g prime times h prime.

So this is going to be g prime of h of x. And g takes key inputs and m outputs. So this one is a-- is an m by p Jacobian of g. I think I might need a little bit more space. I love that I can do that. On the blackboard I always write and write and get to the edge of the blackboard, and then my writing gets tinier and tinier. And then also not equals. Times h prime.

ALAN EDELMAN: I've seen people move over to the walls when the blackboard space runs out.

STEVEN JOHNSON: Of x, and h prime of x takes an input- h has n inputs and p outputs. Right? So it's a p by n matrix. When you linearize it, it's a p by n. These h and g are not linear functions in general, but the derivatives are linearization. There are linear operators. They're just matrices in this case. And you can see that these match up. Right?

Not only is this not equal to h prime times g prime, but that wouldn't even make sense. Right? Right? If you multiply a p by n times an m by p, it doesn't even work. Right? So you really can't change the order of these things. So this is nonsense, I guess, if n is not equal to m. And even if n equals m, then that product means something but it doesn't actually-- it's not equal to the derivative of f anymore.

That would be the derivative of g of h of x if you swap the order. So for just functions that are vectors in, vectors out, the chain rule is just the ordinary product, the matrix product of Jacobian matrices of each of the terms in the order. So you always have to go keep them in the same order, otherwise it's not going to make sense. The outputs are here. Then the outputs of h are there and then the inputs are finally there.

And so not only does the order matter, but the associativity matters practically. Right? And associativity is going to matter where you put parentheses. So here I'm just multiplying two things. There's only one product. There's only one choice. But suppose we have-- suppose we have three functions. Right?

So now suppose we have f of x equals-- this is still Triton blue. F of x equals-- let's see. I'm running out of letters. gh. What letter should I use? a. Well, no. Let's use abc. Let's use abc. A of b of c of x. And suppose that-- suppose these all have different number of components. Suppose this has-- f has m components.

Suppose that x has n components, and these have-- let's see. c has p components. p, and b has q components. And then a-- let me color code it. So a is the output so that it has to be the same as m-- as f. So then what is f prime? That's a function of x. And it's just a product of the three Jacobians. So it's a prime of x. Am I putting my x's in red?

Let me just not put x's. You know that the x's are there. It's a prime times b prime times c prime. Right? And a is m by q. My m's are in blue. By q. b prime is q by p because it takes p inputs and had q outputs. And c is p by n because it has p inputs and q outputs. And so this-- I can now-- so I can't change the order. This is matrix multiplication that's not commutative, but it is associative.

I can always put parentheses. I can always multiply from left to right, or right to left. So this is doing it left to right and right to left. And these are going to have-- it seems like it's who cares, right? You do it in whatever order is convenient. It turns out, it's incredibly important. This one, from left to right, is called reverse mode, and this is called forward mode differentiation for an automatic differentiation.

ALAN EDELMAN: You see the reason for the names? At first glance, it looks like the names are backwards. It looks like reverse modes going left to right, which is forward in English and many languages, and forward is going the other way. But obviously forward mode refers to the first function is c, then b, then a, and it's respecting that same order. So it's forward. Right? Reverse mode is going a, b, c which is the reverse order in which the functions are applied.

STEVEN JOHNSON: Right. So forward mode is going from-- inside out is like from inputs to outputs, from starting with the input c and then going outwards. And reverse mode is going from outputs to inputs. So why does this matter? I claim that this matters. And the reason it matters is that the multiply matrices, how costly it is depends on the shape a lot. So let's think about the cost of matrix multiplication.

So if I do a-- what am I doing here? m by q times q times p. So if I multiply an m by q times q by p matrices. So the result of this is, of course, an m by p tricks. So how do you do it? So for each entry of the output you've probably learned row times column. For each component of the output, you do a dot product. So this is np dot products of length q. Is that clear?

All right. So for each entry, if I imagine this as a matrix, for each entry here I do a row times a column. And the rows have q entries and the columns here have q entries. So it's a dot part of length q, and there are m times p outputs. Right? And so the cost, how much does a dot product cost? Well, to take two dot products of length q-- a dot product of length q. Write it in blue.

So a dot product of length q costs q multiplications, because you have to multiply q things together. And q minus 1 additions of scalars. So basically it's proportional to q scalar operations, which is what the CPU is doing. So the whole thing is proportional to mpq scalar operations. Right?

So that's a review. If I do, for example-- if I do m by m times m by m, two matrices, this is proportional to m cubed. In computer science you would say this is theta mpq. That notation for proportional to. That big O notation. But I'll just say proportional to m cubed. If I do m times m times m by 1-- so if I do a matrix times vector times a column vector, that's only proportional to m squared.

I only do m dot products. Or if I do 1 by m times m by m. So if I do a row vector times that, that's also proportional to m squared. So why does this matter? Does the order of the chain rule matter? So suppose you have lots of inputs and only one output. So let's do--

ALAN EDELMAN: Machine learning, you--

STEVEN JOHNSON: Yeah, like in machine learning. So we have many inputs. Lots of inputs. Actually, my inputs are red, right? So lots of inputs. n is large. Right? And only one output. So m equals 1. So this would be the case, for example, in most of optimization. Right? So in large scale optimization. Optimization. Or, for example, such as machine learning.

So in that case, the output is called the loss function or the objective function. And these are the parameters. Right? The optimization parameters. So machine learning, if the neural network-- you have a big neural network. It has a billion parameters in it, and you're trying to optimize those parameters to make one number better. Right? There's something called a loss, some measure of accuracy of whatever the neural network is trying to predict, for example.

You're trying to make that measure of accuracy better. So you have one output and a billion inputs. OK? And let's also suppose-- let's suppose we have lots of intermediate values, which-- what was I calling those? q and p. All right. Suppose those maybe are comparable to n. So this also happens in a neural network.

You have a zillion inputs, and they go through a bunch of stages of the neural network that have lots and lots of values at each stage, and then finally, at the very end, all those outputs, you have a zillion outputs, like maybe it's trying to predict an image, and then that goes through one function that's saying, how accurate is that image? How well does it match if you're trying to generate images of dogs or something like that, or how good is it? Right? Or this happens in engineering optimization, right?

Which is not really data driven, but suppose you're trying to design an airplane wing that is as strong as possible. Then you have a zillion inputs which are describing the distribution of all the material over the whole shape of the airplane wing. But you only have one output at the end of the day, which is your one number which is some measure of how strong it is, how much weight can it hold, or something like that. Right?

And you take those zillion inputs, get a zillion numbers which are all the stresses everywhere on the airplane wing, and then pass it through another function that takes all those stresses and works out the strength of the wing, and that's what you're trying to maximize. All right? So then think about what the shape of the chain rule is. All right? So our f prime-- right? So this is one output by n inputs, so really, this is a gradient. All right?

And this is what you're going to typically going to want in optimization machine learning. We'll talk about this more later, that you want the-- the gradient gives you basically the direction, the uphill direction-- or the downhill direction is minus gradient. It tells you in what direction do you perturb the parameters to make your system better, like in what direction do you change all the parameters in the neural network to make it more accurate, to make the loss less, or what direction do you perturb all the parameters describing the shape of your airplane wing to make it stronger. Right?

So you really care about getting this gradient, these n numbers. OK? But what is that in terms of the chain rule? We said it's g prime times-- sorry. It's a prime times b prime times c prime. Right? So it's a prime times b prime times c prime. And we said a is m, m by q. m is one, so one by q. Let's just suppose for simplicity that these are just equal to n.

So this is 1 by n. b is n by n. c is n by n. c prime is n by n. Right? c takes n inputs, your x's, gives you n intermediate outputs, p, and then b takes those n things and gives you another n. So its Jacobian is another n by n. Then you multiply by a. And notice it really matters where you put the parentheses, because if you put the parentheses here, think what you're doing, versus if you put the parentheses in the other place. a prime. b prime. c prime.

If you do forward versus reverse mode. Right? So if you put the parentheses here, this is our reverse mode. a, b. This product here is a row vector times a matrix. Vector times matrix is cheap. That's m proportional to n operations. And then what's the output of this? It's the whole thing, the product of vector times matrix-- row vector times matrix is another row vector. And so then you multiply this row vector times a matrix. So you have the whole thing costs order n cost.

ALAN EDELMAN: Vector times matrix, Steven?

STEVEN JOHNSON: Sorry. Order n squared. Sorry. Yes. It's proportional to n squared. Right? Because it's vector times matrix. Right? Whereas if you do forward mode, if you multiply right to left, what's the cost? Well, you're doing b times c first. This is matrix times matrix. That's much harder. Matrix times matrix is n cubed operations, and then you multiply by a vector which is n squared. So the whole thing ends up being n cubed. So you really, really want to do it in this order.

ALAN EDELMAN: This is probably a good time to take a few minute break. We're a little late.

STEVEN JOHNSON: Yep. And let me just say-- we're going to take a break in a second. So this order is also called-- this is reverse mode. It has a lot of names. It's also called back propagation. And it's also called an adjoint differentiation. And it's the key to optimizing functions over zillions of parameters.

It means that I can compute all the derivatives with respect to all the parameters in quite a reasonable cost. And basically it ends up being in cost proportional to the cost of just evaluating f. Whereas if I did on the opposite direction, if I have a billion parameters, it becomes a billion times more expensive.

ALAN EDELMAN: And I do like to stress that most students-- when I did the poll earlier in this classroom, it showed itself. Most students, even for scalar calculus, do the outputs towards the inputs working their way from output. When I ask you about sine of--

STEVEN JOHNSON: Forward mode, yeah.

ALAN EDELMAN: Or sine of x squared or whatever I asked, you are all doing-- without even thinking about it, it's all second nature. You're going from outputs to inputs. And so this reverse mode, which is often more efficient for many real applications today is also exactly what you're doing instinctively without even thinking about it.

STEVEN JOHNSON: Yes. Well, it depends. When it's more complicated functions, sometimes reverse mode is like-- usually when you're doing more complicated things, I would say the forward mode is often the more obvious.

ALAN EDELMAN: That's right. I wasn't going to say that today.

STEVEN JOHNSON: Yeah. And we'll talk about it. And we'll talk more about how these things are evaluated. How many of you have heard of back propagation, by the way?

ALAN EDELMAN: Everybody but one.

STEVEN JOHNSON: Everybody but one. But how many people knew that it was just multiplying Jacobians left to right instead of right to left?

ALAN EDELMAN: Three, including one who never heard of the term back propagation.

STEVEN JOHNSON: OK. Yeah. And we'll show more examples of this. It's one of those things that once you see it explained, it's obvious and trivial. It's just left to right versus right to left, after all. But I think that until you see it explained-- so for example, if you're computing the strength of an airplane wing as a function of 10 billion parameters describing the shape of the wing, it's a really huge mechanical stimulation, like a solid mechanics simulation, to calculate the strength of that wing given its shape.

And it's really not obvious that you can get the derivative of that strength with respect to all billion parameters with basically one additional mechanics solve. And it turns out that that's another instance of this same thing. But once you see it explained-- and we'll see that kind of example more explicitly later on. I think it becomes much more straightforward. Really, it's surprisingly non-obvious when it gets to complicated functions that you can do this.

ALAN EDELMAN: But I think maybe this is just a great time to just take a five minute break.

STEVEN JOHNSON: Yeah.

ALAN EDELMAN: So I've got 12:23 on the clock behind-- in the back of this room, so how about we reconvene at 12:28?