QIQI WANG: All right. It's up. We'll look here and we'll start. The first real lecture on how to use numerical methods on partial differential equations. So just a reminder, of the latter lecture. We did a-- We looked at partial derivative equations in general, right? We looked at the behavior of a few simple differential equations, where we know how it behaves. We looked at [INAUDIBLE] equations, where the solution just keeps moving towards the right. We looked at a diffusion equation, where the solution [INAUDIBLE] diffuse result. It becomes that line afterwards. We also looked at Burgers' equation, which is the equation that can generate shock waves. Where the characteristics can go together they generate shock waves. When they become, or when they diverge from each other they expand, or the solution expands. And the wave will use a little bit of human kind of animation to really animate the solution of the partial differential equation. So this is a good inclusion on the innovative behavior of the differential equation. And so try to remember to us, because in this class, we are going to be using finite difference to solve them. And I'm going to ask you where the solution and body on the screen is correct or not. I mean, not all of them are going to be giving us the right answer, as you already saw in the audio section. And some of them are going to give me, in this case, answers that are correct in some parts. But not really correct in other parts. And I also ask you, like which part is correct, which part is not correct?

OK, so let's first start with a finite difference approximation. In this case with partial derivatives. So let's look at the simple case where my solution U is a function of (x, t). What are the partial derivatives? What is a partial derivative? Du, dx and du, dt. Right? Why do we call it a partial derivative? Because when we are taking derivative of one variable, we are holding the other one fixed. So in finite difference method, a good way to construct the grid to say is I think it's the first time we look at the grid. Is to actually construct the grid first that, it is aligned with one of the variables. So that, if you look at a particular direction on the grid, one of the variables are naturally fixed.

So we use in the last naturally, if you remember, we used these two axes, x and t, to visualize the solution. Right? Now let's use the same plot in one direction x, the other direction t, to visualize my mesh. I'm going to design my grid or mesh to be like this. I think in finite difference it's more common to use the word grid. And the way you go to find a volume or finite element, we will use the mesh instead. And you're going to see why that is the case. So we have a grid like this.

And you see that I'm describing both space and time separately. OK? I'm going to call this-- this x is called a zero. This is delta x, 2 delta x, 3 delta x, 4 delta x. Et cetera. And my timestamp is 0 delta t, 2 delta t, 3 delta t, et cetera. Right? So-- I have my solution right here on this grid. And for example here, my space is equal to-- I'm also going to assign my grid points. So this is what I call u of 4. And one, two, three, four, five. So remember this notation. My subscript, you know, the spatial grid, and my superscript 5 denotes which timestamp I mark. And here would be U of 4, 4. Here would be U of 3, 5, et cetera.

OK. So now I have the solution. I have the discretized solution leaving all these grid points. How do I approximate the spatial derivative of x at some grid point i, and some timestamp n? Any suggestions based on What you learned being finite in ODE? Taylor series, yes. But can somebody give me a suggestion of what should I use?

STUDENT: [INAUDIBLE].

QIQI WANG: 1 delta x further towards the left or right?

STUDENT: [INAUDIBLE].

QIQI WANG: OK, let's try to use the right. OK so I can-- nope. I can approximate the partial derivative by-- in this case, I'm fixing n, right? In partial derivatives, I fix the other variable. In this case, I fixed n, which is the same as fixing t. I took the difference in the x direction and the i direction, which is also in the x direction. So this is an approximation of the derivative in x. I can call this forward in space. OK. I can also-- let me see. I'm going to insert page. Escape it turns out I want smaller than A4. Come on. I think this is good, OK. I can also do backward in space. Backward in space, I would get Uin minus Ui-1n over delta x. So this is backward in space.

When you see it as-- forward order and backward order have different stability properties-- this is also going to have different stability properties. And I can also do central difference. So central difference would be U of i plus 1-- that is towards the right-- minus U of i-1-- towards the left-- divided by how much?

STUDENT: 2.

QIQI WANG: 2 delta x, because that is the space in-between them. So this is central in space. How about time? We also need to discretize time. We also need to discretize partial U partial t at i and n. How do I approximate that?

STUDENT: [INAUDIBLE]?

QIQI WANG: Exactly the same way as we did in ODEs. And we can approximate it as U of i n plus 1 minus U of i n, divided by delta t. And that could fall into what scheme we use in ODEs? This grid is going to forward Euler, and we are going to see later what other things we can use. So let's try to implement one of the combinations. So which one-- [INAUDIBLE] because that's really the simplest type of [INAUDIBLE] scheme, I think. And you want both or one of these?

STUDENT: [INAUDIBLE]?

QIQI WANG: Huh?

STUDENT: [INAUDIBLE].

QIQI WANG: What?

STUDENT: Central in space one.

QIQI WANG: Central in space one. All right, good. Because you guys probably can do Taylor Series in your head and figure out the central difference is actually second-order accurate. It is better than the other two. So good. You have a good pick.

OK, so I'm going to go to Matlab. The first thing I want to show you in Matlab is-- let me show you in the next class what the difference between finite difference and finite volume. But I'm just going to show really kind of how a finite different discretize a function. So when we have a function in space, we have to discretize it, right? So, because the function is really infinite-dimensional, we only have finite memory.

So if you draw a function like this, a finite difference only stores the values on the grid points. So the red circles are what it stores. What is in the middle are just forgotten by the computer. And all the derivatives have to be approximated by these discrete points. And this now seems natural to you for sure. But, as we go to find the volume, you are going to see how different discretization schemes are going to be discretizing a function.

OK so let's try to implement a central difference plus forward Euler on the advection equation. So I haven't said what equation I'm going to do yet. So I'm going to implement it on this equation. Partial U partial t plus partial U partial x is equal to zero. With a periodic boundary condition, you add 0 is equal to U at 1. So this is a periodic boundary condition. So this is the same equation as we tried to animate last Thursday. Anybody who goes out of these doors comes back from inside here. So we should be seeing-- this is the linear advection equation. And all the characteristic lines are parallel. The characteristic lines are like the colorful lines in the solution in the x and t diagram.

So all the characteristic points go towards the right, and along each characteristic line, the solution is constant. So let's try to implement the forward order central difference and see how the solution looks like numerically. So I'm going to New file to Script. So central space, forward time. So this is my central space, forward time. I'm going to set up my number of grid points. Let's just use 100 grid points in space. And because I have a domain from 0 to 1, my delta x is equal to 1 divided by n. So this is my delta x.

Let me choose the initial solution that looks somewhat like Gaussian. OK, so we have a peak, and we see how the peak evolves. So let me, first of all, set my x equal to 1:N times my dx. So I have a grid, and accessed the spacial location of the grid. And let's set my u0 my initial condition, to be exponential of minus x minus 0.5 squared, divided by 0.1 squared. And I have to do this divide. So we can take a look at how the solution looks like.

We can plot x and u0. And we are going to run the script to see how the solution looks like. So this is the initial condition. We have a Gaussian curve. The shape of the curve is exponential. 1 minus x minus 0.5. So 0.5 is the central of the Gaussian. And the reason [INAUDIBLE] standard deviation can be plus 1, [INAUDIBLE]. The width of the Gaussian or half-width of the Gaussian. Any questions on this? No? OK. So let's decide on a timestamp. What do you guys want on the timestamp? 0.1? All right.

I goes from 1 to 1/dt. So let's simulate it for one time unit, because after one time unit, the speed of the solution going towards the right is 1. That is because in my equation, I put if there should be a small c, then this c is equal to 1. So my speed of the solution going towards the right is equal to 1. So after one time unit, the solution should go through the domain once and end up exactly at the same location as the initial condition.

Now when I'm doing forward order, plus central difference, I need to first compute the dU/dx at each grid location. And dU/dx, at each grid location, is the difference between the solution towards the right minus the solution towards the left, divided by 2 delta x. And look carefully at how I'm computing the solution on the right and on the left. du dx is equal to-- what I'm going to do is I'm going to do circshift of my u0, of my u. I'm first going to say u is equal to u0. [INAUDIBLE]. I am going to do circshift of u, and I'm going to be shifting 0 on the first dimension and a -1 on the second dimension.

So does this give me? I'm going to go back to Matlab and tell you what it gives us. So if I do 1 to 10, for example, I'm going to get an array going from 1 to 10. If I do circshift of 1 to 10, I'm going to shift it-- I'll put 0 in the first dimension and 1 on the first dimension. This is what I'm going to get. I'm circularly picking these 1 to 10, towards the right. So instead of 1 here, I get [INAUDIBLE], which is why this method-- the [INAUDIBLE] method-- comes out of the [INAUDIBLE] and goes here. And the [INAUDIBLE] is that [INAUDIBLE] here. Everybody's shifted towards the right 1.

Now if I want to grab the i plus 1 person at the i place, this is the opposite of what I need to do. What I need to do is I need to put -1 here so that at the first place, I'm going to be grabbing the next solution and the next grid point. And particularly at the last grid point I'm grabbing the solution [INAUDIBLE], which is actually the other way you're supposed to do it on the left [INAUDIBLE], which is 1. So this is the way you can [INAUDIBLE] Ui plus 1. If you have periodic [INAUDIBLE]. What about Ui minus 1? This is my Ui minus 1. This is my Ui-1. Any questions on that? Someone said it back there?

STUDENT: [INAUDIBLE]?

QIQI WANG: Yeah. On this one, things are being rotated towards the left. The 1 which is here now was the end. And the 2 [INAUDIBLE] here now is here. 3 was here, here. Yes?

STUDENT: [INAUDIBLE]?

QIQI WANG: So in the ODE part, why didn't we do a circshift? The answer is in the ODE part, we didn't do a circshift, because we don't have the solution at ndx. So when we are at the end timestamp, we don't have the solution at the n plus 1 timestamp. While in the PV part, we have a grid like this, and we are given the whole initial conditions. So I've given the whole initial condition at [? p ?] equal to 0. So we have all the data at this timestamp. Now our job is to go from this timestamp to this timestamp. We want to compute from the solution here all the solutions on these grid points. And then we go forward and forward and forward.

So there is a fundamental difference between space and time. In time, you don't know what is going to happen tomorrow, right? In space, we have access to the solution at all the spatial locations. At the current timestamp, and if you stored the past timestamps, also at the past timestamps so you know all these things.

So that's the difference between space and time. Although, doing Taylor series analyses on them are the same. But in terms of coding, they are different. In terms of the order of computation, they are different.

So going back to our Matlab code, we shift towards the -1 direction that is grabbing Ui plus 1. If we shift the whole thing towards the positive 1 direction, we are grabbing ui. And if we divide that by 2dx, we are computing du dx, right?

So now we have du dx, we want to plug this into this equation. dU/dt, in this case, is equal to minus of du dx. This is what this equation is. If you shift du dx towards the right-hand side, du dt would be minus of du dx. So that's what we are going to do. du dt is equal to minus du dx We used the partial differential equation to convert a spatial derivative into a time derivative. This is the result of using the partial differential equation.

So the first step, find the difference approximation of the spatial derivative, the partial derivative of x. Use the differential equation to get the time derivative. And anybody want to suggest what is next? Are we back into the ODE world? We are back into the ODE world. We have computed the time derivative of the solution at all the grid points we had. We can use forward order. From the cutoff, we can use [INAUDIBLE] or other sexy things we learned in the ODE section.

So let's do the most elementary thing and apply forward order. u is going to equal u plus dt times du dt. This is exactly how you implement forward order in ODE. So we already plotted the solution at the initial condition. Let's also hold on so that we don't lose the plot of the initial condition and also plot my x and u at t equal to 1. And let's see how the solution will look like. And let's actually plot this in red, because the initial condition is going to be-- let's plot the initial condition black and the solution red and see how it looks like. We get a pretty big solution.

This may not be the fault of spatial derivatives, though, because we know forward order has a pretty limited stability region. But we know forward order is 0 stable. Which means what?

Which means there it is always going to shrink the timestamp. Which means it is globally accurate. And global accuracy means when you make the timestamp smaller, the solution of the ODE at least converges to the [INAUDIBLE] solution of the ODE. So let's see if it helps in this case. Let's make delta t equal to 0.001, and let's run it again. Let me close all before I do the plot.

This time, we get pretty good. We do see the solution having a pretty good match with the analytical solution. And let's also visualize how this thing progesses. For every timestamp, let me actually do the plotting. For every timestamp, let me do the visualization at clf which means clear the figure. I'm going to plot the initial condition. I'm going to hold on. And I'm going to plot the solution. So I'm clearing the figure, plotting the initial condition, plotting the solution. And then you could do draw now. So let's take a look at how-- let me actually pause for a little bit. Pause for 0.01 seconds so that we can actually see the solution moving.

So the red one is my solution as my time goes from 0 to 1. So does it look like we are capturing at least the qualitative behavior of the analytical solution? Yes. Although, it is not completely satisfactory, because [INAUDIBLE] and the solution seems to be growing a little bit.

STUDENT: [INAUDIBLE].

QIQI WANG: And also, another concerning point you have [INAUDIBLE] is that the solution initially was all positive. And you can look at my analytical graphing for the initial condition where [INAUDIBLE] quantity. It's the exponential of something. How can it not be positive? But, like [INAUDIBLE], it becomes negative at some point. So it overshoots and undershoots. I mean, we're not going to discuss this today. But we're going to be discussing the reason of this error, both on Wednesday-- Professor Wilcox is going to teach on Wednesday. And on the error of this finite difference method. How to analyze this error and convergence. And another, very related issue is stability. That is something we are also going to be discussing in the PDE section.

But this class, we are just happy we got a solution. And we can also see that the solution is going to get better as I make my grid to be finer. Instead of 100, let's do 200, and we run the same thing. It's going to get slower, but the rate of growth is going to be not as severe as in the 100 case.

[INAUDIBLE]. Don't really see the undershooting [INAUDIBLE] for the [INAUDIBLE] Question?

STUDENT: [INAUDIBLE] questions. [INAUDIBLE] Why didn't we see the [INAUDIBLE]?

QIQI WANG: Right. So we did a comparison between-- actually, let me ask you a question about it first. So the question is on backward difference, what's the central difference? So what do we need to change here to make it backward difference?

STUDENT: The ability of [INAUDIBLE]

QIQI WANG: First of all, it just needs to be delta x not 2 delta x. And instead of ui plus 1 minus ui minus 1, we have u minus ui minus 1. So we're just going to be putting u here. This is going to be backward difference. And when you run backward difference-- whoa. Yeah, we get a similar solution, but now what does the solution do? Instead of growing a little bit, it decays a little bit. And we see we have the same number of grid points. That's where the difference seems to have more undershoot than overshoot of the backward difference. And again, Professor Wilcox is going to discuss the accuracy of these two methods on Wednesday.

So this is the implementation of the central in space and backwards in space finite difference scheme. And we also discussed in the last lecture how to solve-- what is the behavior of a diffusion equation when we have diffusion instead of advection. We have a differential equation that is like this, partial U partial t is equal to k times partial squared u partial x squared.

And we can even have a mixed differential equation. We can have a mix, advection and diffusion. So how do we approximate this term? How do we approximate the second order derivative? Now the second order derivative has a very concise approximation. So the second derivative of x at a particular grid point at a timestamp can be approximated as this-- as u of i plus 1n minus 2 Ui n plus U of i-1 n. Divided by delta x squared.

Unlike the first order derivative, you have the choice of forward and backward difference. For approximating second order derivatives, this is the single most popular scheme we use. And it is very difficult to find an approximation that is as popular as this. And let me describe to you why this is the case. OK We are approximating the second order derivative. We are only using three points, the center, left, and right. And amazingly, this thing has second order accuracy, has very good accuracy.

And you can show that by doing Taylor Series on that-- it's U of i plus 1 n can be expanded as U of i n plus delta x times, in this case, the partial derivative of partial U partial x at i n, plus half of delta x squared of the second derivative of U at i n. I'm going to be expanding more terms.

It's the third of the derivative times delta x cubed divided by 6. And I'm just going to write over here O delta x to the 4th. Now this this term. I am also going to be expanding this term. Ui-1 of n is equal to Ui n. In this case, it's minus delta x because the difference between the i-1th grid point and the i-th grid point is minus delta x.

The second derivative term is plus again, because we are taking the square of minus delta x. The third derivative is minus again, because we are taking the cube of delta x. And again, we have plus delta x to the 4th. Now we can start to see which terms start to cancel out. See we have one of Ui n here plus another Ui n here. Now we have minus 2 of Uin. So this and these cancel together, cancels with this minus 2Uin. All right, this is one term.

This and this is added up. So i plus 1 i minus 1 is added up. So the first derivative term cancels. The third derivative term, instead of canceling, they add up. So they, divided by delta x squared, serves as an approximation of partial squared U partial x squared. It is a consistent approximation for partial squared U partial x. And usually, when you have three terms-- when you are playing around the coefficient of three neighbors, you can make sure three terms add up to what you want it to be. You've got rid of the constant terms. You've got rid of the first order derivative terms. You make sure that the second order derivative turns out to be what you want to approximate. But in this case, you get a bonus. The third order derivative term cancels out too, just by accident, and now, you only get a delta x to the 4th, which after dividing by delta x squared, you get a second-order accurate approximation to the second derivative.

So the error, which is the actual second order derivative minus the finite difference approximation goes like O delta x squared. And this is pure luckiness. Any questions on this? This scheme is what we are going to be using.

Now let's go back and modify our code to also do second order derivatives. So we have du dx. We have d2udx2 equal to-- let me copy this. And we have a Ui plus 1, which is circshift of U towards or the minus direction. We minus 2 times u and plus u of i minus 1. And divide it by delta x squared. So this is our second order derivative.

If we want to solve the diffusion equation over here, the du dt would instead of being minus du dx we have the positive of d2udx2. And let's put a cover here, a 0.1 cover so that it diffuses slower. And then let's look at how the numerical solution looks like. Let's change back to 100. Oops. Again, if the stability is coming up, let me make sure we have a small enough timestamp.

So we can see are the solution of this PDE agrees with the analytical behavior of the PDE, which is the solution diffuses out. It becomes less sharp. And we can even combine these two terms. We can du dx plus this, or we can do a convection diffusion equation. And we can see the solution diffuses and also, in this case, it's shifting towards the left, because I think that the right convection-diffusion equation should be minus du dx. Let me make sure that is the case.

So we have a solution that is diffusion by the diffusion terms, and also advects toward light by this advection. Now let's take just a 5 minute break before we go to the matrix form. We did find a difference [INAUDIBLE]. Before we go into the matrix form of this finite difference method-- and we see the matrix form is going to come up a lot, and it's going to be very useful in, for example, in physics scemes of finite difference methods. All right. OK. Let's continue. So the really [INAUDIBLE] like you haven't seen before.

There is, as I said, unlike discretization in time, you only have one stamp of information, right? You can only view things one step at a time. You don't really know what happens tomorrow, what happens at the next timestamp. So you have to go one step at a time.

We find a difference when we discretize things in space we actually have much more data than we have in ODE discreditization because we have the entire timestamp. At every grid point we have the solution in our head. OK. That makes it possible to write and find a difference in a matrix form.

OK. So when we write the finite difference in the matrix form, we first need to write our solution in a vector form. We know our solution-- let's say we know our solution yields at the first grid point, u at the second grid point, u at the n-th grid point. We have them all, right? So that is what we have.

Can we compute not d but partial-- partial u partial x at the first grid point, partial u partial x at the second grid point? Is that true of partial u, partial x at the last grid point? As a joint matrix-- I said joint matrix because sometimes when you have, let's say, a million grid points, how big is the matrix going to be? It's going to be a million by million matrix, right? It's going to be a million by million matrix. Think of n as a million.

And this is why-- I don't know if some of you have heard of the term spark matrix. That is why spark matrices are invented because usually people don't have enough memory to store those million by million rate. I'm going to talk about that later. But sometimes you have this joint matrix.

Now, what is this joint matrix? What are the elements in this joint matrix? If it is a central difference-- let me write down the central difference formula here. Let me write it here so that it's down here now. In a central difference, partial u, partial x at i is equal to u or i plus 1 minus Ui minus 1 divided by 2 delta x.

And let's assume again we have here [INAUDIBLE] boundary condition so that if i is equal to n, then i plus 1 would be equal to 1. If i is equal to n, if i is the last grid point, then your neighbor who was y-- which actually, the first y of the other end. So n plus y is equal to y.

What is the first line of the matrix? [INAUDIBLE] boundary condition. What happens if I want to approximate of e to the x at what, when i equals to what?

STUDENT: [INAUDIBLE]

QIQI WANG: Yeah. I have 0 here because this formula do not depend on u and i, right? Now when I go to the second one, what is the modify on u2? The modify on u2 is 1 over 2 down x, right? And is there any other than 0s?

STUDENT: [INAUDIBLE].

QIQI WANG: All the way up to the end because I have periodic boundary condition, I have this. OK. How about the second line? The second one is actually a easier way to do it. When i is equal to 2? When i is equal 2, du dx at 2 is equal to y. It's equal to u3 minus u1 divided by 2 down x. So u3 and u1, which is based on this-- so this guy is now empty. This guy is now empty, right?

We would get minus 1 over 2 down x here because u1 is being minused and u3 is being plussed. So we have 1 over 2 down x over here. We have 0 in the middle, 0, and 0. Everything else is 0. So I think under [INAUDIBLE] each line there is only going to be two non-zeros, only two non-zeros because each derivative only depends on two neighbors. That means we should only have two non-zeros.

If you have more sets over here, then they are going to be more non-zeros. Two terms means two non-zeros for each line. And for the third one, my minus is going to be operating on u2. My plus is going to be operating on u4. And for the fourth line, my minus is going to be operated on u3. My plus is going to be operated on u5, and et cetera, et cetera.

So diagonally, it's always 0 towards the very last line. The sub-diagonal is always going to be minus 2, minus 1 over 2 delta x until the last line. And the upper diagonal is always going to be 1 over 2 delta x. Am I missing something here?

STUDENT: [INAUDIBLE].

QIQI WANG: So the last one, the positive, is over here because of the periodic boundary condition. Yes?

STUDENT: [INAUDIBLE].

QIQI WANG: Exactly. Because we also need a periodic boundary condition, this point, instead of lying over here, there is no longer anything [INAUDIBLE] anymore. It is actually the first element of the left. This entry, no matter how matrix multiply with a vector? This entry is going to be multiplied [INAUDIBLE] instead of U n plus 1, which [INAUDIBLE].

Any question on this [INAUDIBLE] matrix? So all of this matrix is really a million by million. But it only compares two a million, 1, 0 entries. Most computers have enough memory to store a million non-zero entries. All right. So this is central difference. Let's just do an exercise of how, if we do a-- oh, I haven't talked about upwind yet, so backward in space difference.

STUDENT: [INAUDIBLE].

QIQI WANG: Huh?

STUDENT: [INAUDIBLE]

QIQI WANG: Towards the upwind of the [INAUDIBLE]. Good guess. Right. So if I do a backward in space, I'm going to be writing the same thing, partial u partial x at 1, partial u, partial x at 2 essential, partial u partial x at n. I think I used a small-- yeah. I used a small n-- is equal to a joint matrix of the same size times u1, u2, et cetera. So nothing has changed, except for the content of the matrix.

And a backwards space difference is partial u partial x at i is equal to Ui minus Ui minus 1 divided by delta x. All right. Somebody help me fill in the blanks. Fill the blanks of a million by million matrix. Doesn't take that much time to fill in, actually. What is the first top level entry?

STUDENT: One over delta x.

QIQI WANG: One over delta x. It is no longer 0 because the du dx at ix usually depends on Ui. In the last case, the du dx at i does not depend on Ui. So we have one other entry because we have two elements in each derivative. What is the other non-zero entry?

STUDENT: At the very end.

QIQI WANG: At the very end. That one is the same. All the others are 0's. How about the second line? We have diagonals to be always 1 over delta x because we always have 1 over delta x multiplied by Ui. That going to be du dxi. And we have 1 minues-- sorry.

We have minus of 1 over delta x always towards the left because the value at the left is always multiplied by minus 1 over delta x. All right? OK. Another x.

So OK. So how about let's try to, in MATLAB, try to construct the matrix form. When we construct the matrix form, we don't need to do it in every concept. We an do it well in advance. So after we settle initial conditions, let's actually-- before we settle our initial conditions, we know what n is, we try to set up matrix a.

OK. Let's call it a ddt. So that is the matrix that gives me-- sorry-- ddx, that gives me derivative to x. So let me see what I did. Here is actually backward in space. So let's try to reproduce backward in space.

This is the matrix I have. I already have delta x computed. Yes, I do. So then I need two matrices. One matrix, the first matrix, is this guy. I need, first of all, this matrix. And somehow if I can add that matrix to the second matrix, if I can add that matrix to this matrix plus this little element over here, then I have the entire matrix.

So let's look at the diagonals, the blue one. Can everybody tell me, is there an easy way to [INAUDIBLE] the blue matrix?

STUDENT: [INAUDIBLE].

QIQI WANG: Huh? It's identity, it's identity matrix times one over delta x. That one is actually super easy. How about the other one? How about the black one?

STUDENT: [INAUDIBLE].

QIQI WANG: Yes. I can use my x. I can use diag if I don't have this one. I'm going to see later on if I have not periodic boundary condition. Periodic boundary condition is actually this. If I don't have periodic boundary condition, [INAUDIBLE] then diag is a very good way to do it. In this case, I'm going to use the same function, the third shift I just used to centrally shift the solution itself. I can also shift the matrix. I can take this matrix, shift it to the left or towards the bottom. It's the same way. I'm going to get the back matrix.

It is equal to i of n divided by delta x. So that gives me the blue one. I'm going to minus in by dx. But I'm going to circshift the whole matrix by left 2 towards the bottom. Towards the bottom is [INAUDIBLE] in the third direction, 1 in the other direction, 0 in the [INAUDIBLE] direction. So I'm going to shift it by 1 and 0.

OK. That is my plan a difference matrix. Now how do I apply it? How to apply it-- when I computed du dx, When I compute du dx-- let me first do one thing. I want to make my everything to be a color matrix first because by default, if I do 1 to n, it's a row matrix. Where if I want to multiply the matrices that way, I need my solutions to be like a column. So initially, I want to construct this by columns. And everything should be in columns. And this is no longer true anymore. I'm going to remove this.

I'm going to just do a linear advection equation first. But my du dx is equal to a_ddx times u. That's how easy it is to apply this on a difference matrix. Also, I have the final difference matrix, I don't need to do anything at each concept other than just applying the matrix just as if it is the mathematical operator.

I just think this matrix is my actual [INAUDIBLE]. I apply on top of u, I get partial u partial x. So this a serves like a mathematical operator. I'm going to round this. I'm going to get the same solution. Of course, this is backward in space. I got a solution that moves towards the right but also the case. So this is what I have as a backwards difference.

And I can modify this. So I'm going to [INAUDIBLE] this out. This is backward difference. I'm going to do central. And in the central, the difference is I need to circshift both. I need to circshift this in the opposite direction. And instead of dividing by 1 delta x, actually, let me divide this by 2 delta x. So this is going to be my central difference, right? Agreed?

If you look at this, I have one identity matrix. shifted towards the right. And I have another one shifted towards the left. So I get two matrices subtracted. I get the central difference. Let me look at delta behavior. Looks like the same as the central difference. It does, right? I have a solution where we go to right. It doesn't really decay at all. But [INAUDIBLE] a little bit.

[INAUDIBLE] if the maximum occasionally gets greater than 1, it still shifts down a little bit. All right. Any questions on the matrix form of these two final difference operators? No? Whenever you need them, I'm basically constructing a final difference operator just like a mathematical operator, just like ddx. And when I need it, I can just apply it.

The interesting thing is that the operator I have already contains the boundary condition. The operator I have here, the periodic boundary condition is encoding into the operator [INAUDIBLE]. If I didn't have periodic boundary condition, would I still have this? No. I wouldn't be able to say, if I want to-- I wouldn't be able to say if I only graph the solution towards the left, I just need to grab the one at the right hand. I wouldn't be able to say that.

So what if we-- let's say we have a boundary condition at the left, but if it is not periodic, what if we have a boundary condition that says u at 0 equal to 1? OK. So let's see. U at x equal to 0 at net equal to 1. This is consistent with our matching question, because in this matching equation, the characteristics move towards the right.

So if you didn't have periodic boundary condition, you need to specify the solution. You need to specify the solution also here and here, right? You need to specify the solution where the characteristics originate. Because if I'm here, I have the initial condition that p is equal to 0. You also need to specify those conditions here, either boundary conditions. That is when x is equal to 0 or x is at the left end of the domain, [INAUDIBLE] equal to 0.

So this is the left boundary condition. All right? OK. In that case, how do we deal with that? How do we encode that boundary condition into the matrix, into the matrix form of finite difference. Let's look at this. Let's take a look at the matrix.

And the only thing we need to change is which one? Which row of the matrix do we need to change? If we need to change one at all. We only need to change the first row, right? Because all the other rows of the matrix are still exactly the same because they are the interior. Only the first row involves the boundary.

So let's look at the first row. The first row has partial u, partial x at r equal to 1 at n [INAUDIBLE]. Originally, it is equal to u of i minus 1, which is in this case is 0 minus u1 divided by delta x.

This is my backward-- again, this is my backward in space. This-- yeah. It's the other way around. Thank you. It is u1 minus u0. Yes.

OK. In the previous [INAUDIBLE] we said the domain is periodic. So 0 is just U n, right? So what this is is that we have a boundary condition that is given to us. You know what exactly U0 is. We know that your 0 is equal to 1. Then the derivative is u1 over delta x minus 1 over delta x.

We have a term that doesn't depend linearly on the solution. Or, if you look at the matrix form, we have multiple conditions all the way. We don't have this. But in addition to the matrix, we have something else. Obviously, we have a negative of 1 over delta x sitting over here as an additional vector to this matrix.

Let me make a new page. OK. I'm going to draw what the matrix is going to look like. When we have a boundary condition, like partial u, partial x at 1, partial u, partial x at 2, partial u, partial x at n is equal to a similar matrix at minus delta x on the sun diagonal.

I want to leave enough space for the vector ends on the right. My u1-- I'll think I'll use the big U for the first U n. This is the big U plus something else. And that something else is going to be 0 almost everywhere except for the first row, which is minus 1 over delta x. And it is going to be the 0 everywhere because I don't need to modify anything.

How id I achieve this? I plucked the boundary condition into the finite difference formula at the grid point that is close to the boundary. All right? That is a general technique of locking in the value of the boundary condition into the [INAUDIBLE] difference here.

So let's modify what we did under here. So let's copy this. I think I should rename this as backward because this is backwards. And I'm going to do probably another one and say boundary condition backwards space forward time. That's correct.

So what do I need to do? I need to change-- oh, OK. I need to change the operator. I need to change the operator to include the boundary condition. How do I do that? If you look at what the matrix is like, I still have the diagonal term to be exactly the same. I still have my identity of n divided by delta x.

So that is this part. Now, I also have my rare part, which is not exactly a circular shift of identity matrix. So here I'm going to use the function code diag. Diag, it's a function that you need to give to me. What are the elements of a diagonal and how much I shifted.

If a is equal to 0, then elements aren't exactly on the diagonal. If a is equal to minus 1 then I get exactly what I want because it's minus 1 shifted towards the opposite diagonal, which is 1 shifted towards the lower diagonal. So the elements would be once with m minus 1 elements divided by delta x-- these are the values-- and minus 1 towards the shift.

OK. So this is what the matrix is going to be like. I'm going to paste it here and take a look at the matrix. The 100 by 100 double on the diagonal [INAUDIBLE]. I get what I want, and I also don't have this-- I don't have a minus 100 over here in the periodic case.

Now, let's run it to see what we have. OK. We still see these things going over here. What happens on my left? Oh, OK. Did I forget something? I forgot the boundary condition. So I forgot to add the vector v when I applied this derivative. The end result here is I have a boundary condition of 0 here.

I'm setting the boundary condition to be 0. [INAUDIBLE]. As this thing advances towards the right, it's going to be no longer [INAUDIBLE] towards the left. So yeah. So I did forget something. But we still have such a boundary condition that is not periodic. Because you see this [INAUDIBLE] go out, I'm still going to have 0 here.

OK. So let's add to the b. My b_ddx is going to be 0n1. And my b_ddx, the first element is going to be 1 minus delta x. Minus. All right. When I apply it ddx is this plus [INAUDIBLE]. Yeah. This is still going towards the right, and it doesn't come back from the left.

So now let's look at what if we do have a non-trivial boundary condition. Why? What happens? Ah, undefine the functional variable, b_ddx. All right. I'm going to round it. All right. So we have a boundary condition of b equal to 1 over here as you can see. This is the behavior of the differential equation [INAUDIBLE] so we'll see everything that wraps towards the right.

And [INAUDIBLE] there shouldn't be a discontinuity over here, but the [INAUDIBLE] actually moves this discontinuity out, and you get this finite jump from 0 to 1 over here. But you still the boundary condition being forced correctly as things do move towards the right. All right? So the right way to put boundary conditions, one right way to put the boundary conditions is to substitute the boundary value into the differential operator, into the discrete highest differential operator and derive not only a more refined matrix form, but also a vector that added on to the matrix form.

STUDENT: [INAUDIBLE].

QIQI WANG: OK. So the question is, this form o the boundary condition looks like we are asking a [INAUDIBLE] into the differential equation. And you are absolutely correct. I think not only in this case, but in several others [INAUDIBLE] of laying the boundary condition, it's essentially adding a [INAUDIBLE], like [INAUDIBLE] boundary condition. So sometimes [INAUDIBLE] the boundary condition is like knowing how to add the fore onto the right-hand side near the boundary to-- so that your emergence solution converges to the analytical solution with that boundary condition. All right? Did I answer your question? Does that?

STUDENT: [INAUDIBLE].

QIQI WANG: So the graph I have is-- the graph I have is not with the boundary condition but due to the minor difference here. Yeah, the scheme is on the-- gives you a truncation error. It is marked on the [INAUDIBLE] derivative exactly on that truncation error. [INAUDIBLE] smooths out any discontinuities, if there's computation error, you can actually see that a truncation error looks like a diffusion here. It looks like you're diffusing your solution out instead of just advecting it. And the behavior of the central difference scheme is very different.

All right. OK. I'm just going to very quickly also give you the finite difference form, the matrix form of the second order derivative. The matrix form of the second order derivative is if you have the second order derivative of u at 1 and the second order derivative of u at n-- again, is equal to this joint matrix. And again, let's assume is periodic boundary conditions so that we don't have to worry about the boundary conditions for now.

OK. So the second derivative, partial u, partial square u, partial x at i is equal to three terms. We have a u of i plus 1 over delta x squared minus 2Ui over delta x squared plus Ui plus 1 over delta x squared, right? [INAUDIBLE] the same as the second order [INAUDIBLE].

So Ui plus 1 minus 2Ui plus Ui plus 1, everything divided by delta x squared. Now if we have periodic boundary condition, how do I fit this into the matrix?

STUDENT: [INAUDIBLE].

QIQI WANG: Sorry?

STUDENT: [INAUDIBLE].

QIQI WANG: The derivative. Oh, the derivative. Yes. Thank you. Yes. Yes. OK. All of these terms going to that matrix. So let's start with the diagonals. The diagonals are all the same. They are coefficients on this term. This term goes into the diagonals. Yeah. This term goes into the diagonals.

And let me color that by blue. This 2 over delta x squared--

STUDENT: [INAUDIBLE].

QIQI WANG: Negative. Thank you. Delta x square all the way to here minus 2 over delta x square. Now, where does this go, Ui I plus 1? Both with their upper diagonal goes over here. It's 1 over delta x squared, 1 over delta x squared' all the way to 1 over delta x square, and here, 1 over delta x squared. Because when i is equal to n, i plus 1 is actually equal to-- it's the same as ux1.

OK. And lastly, let's fill in the-- OK. Let's fill in this i minus 1. This is i minus 1. And fill in the i minus 1 term we have on the lower diagonal, it's the same as the i plus 1 term, 1 over delta x squared, 1 over delta x square, and 1 over delta x square. The first term is actually over here.

It is in order to get the second derivative for the first grid point. You have to graph all the way from the last grid point. OK. So this is the matrix form of the second order derivative. OK. So next class, we're going to look at how the finite difference schemes converge as we make the grid point more and make the spacing smaller, and also stop to look at two-dimensional problems. All right. Thank you.