PROFESSOR: So today we're going to keep talking about boundary value problems. We'll do that again on Wednesday. On Friday we'll start partial differential equations. And next week we'll work with COMSOL, which I hope you guys all have installed on your laptops. Please check that you have it installed and you can turn it on, your licensing things work and stuff like that. We'll have a COMSOL tutorial I think on Monday in class. And you'll use that for one of the homework problems not immediately but coming up.

Speaking of the homework, I received some feedback that the homeworks have been taking an inordinate amount of time. And so I've discussed this with the other people involved in teaching it and we decided to drastically cut the amount of points given for write-ups to try to encourage you not to spend so much time on that. And instead I'd really rather you spent the time and did the reading instead of spending an extra hour doing write-up. You won't get any more points for doing it. So if you want-- I mean, I love beautiful write-ups. I'm sure the graders appreciate the beautiful write-ups. It's good for your thinking to write clearly but it's not worth many, many hours of time.

And in general, the instructions at the beginning of course, are on average over the course, your 14 weeks, it should be about nine hours per week of homework. So that's maybe 13 hours per assignment, so you have 10 assignments. And so if it's getting to be 15 hours, you've spent too much time on it and just forget it. Just draw a line and say I'm done, time ran out, and that's fine. This is-- getting the last bug out of your Matlab code is not the main objective of this course.

And really, the purpose of homework is to help you learn, not that we need to know what the solution to this problem is. This is now-- I'm getting the solution from somebody else in the class too so this is-- I don't need it from you. The TAs probably already did it already and they have the solution too. So we know the solution. It's not so essential. Its purpose is to help you learn and how to figure it out. And beyond a certain point, it's not that instructive in my experience. Though sometimes the fifteenth and a half hour you suddenly have the great insight and you learn a lot, but most the time not.

And by the way the reading is in [? Beer's ?] textbook pages 258 to 311. And there's also some nice short readings by Professor [? Brautz ?] that have been posted, only a few pages long but definitely worth a look. Both on [? PPPs ?] and also in ODEs and [? DAEs. ?]

So today what we're going to talk about is relaxation methods. And we talked last time about the shooting method. So that's a good method. At least one famous numerical methods book recommends you always shoot first then relax. So try the shooting method. If it works, you're done. If it doesn't work for you, then you may have to use a relaxation method. And we'll talk a bit now about the relaxation methods.

And the general idea is you're going to write your y-- which is an approximation, it's not going to be the real solution-- and you're going to try to write that typically as an expansion to some basis functions. So let's say the nth component of y. And then you're going to vary these coefficients, the d's, to try to make your solution as good as possible.

And now we're talking about different definitions of what good is for the solution for a problem. And you have to be aware that almost always, this will not exactly solve the differential equation system. So it's always going to be wrong everywhere, typically. And you can get to decide sort of-- if you wanted to be good at some particular spot, you can do something about that. If you wanted to, on average, be good in some way, you can decide that. And it's sort of how much error you can tolerate.

And in general, you have to do a finite sum of a finite basis set. For many, many ODE, [? PPP ?] problems, there's math proofs that if-- and the limit is this goes to infinity if you add an infinite number of functions here. You can always make it work to get the true solution. But you can never afford that so you're always finite truncated basis set is what you're using.

And so a lot of the accuracy things have to do with exactly what functions you choose and exactly how many of these terms and sum you include. But typically, these problems will start from the beginning, you'll say I'm only going to include so many, how big my computer is. And so you're going to be stuck with some error, so that's just the way this.

Let's recall the problem we have. If we write it in the-- as a first order ODE system, it's dy/dt-- well, there's some function of dy/dt and y and t that's equal to zero which often can be written something like this, dy/dt is equal to an f [INAUDIBLE]. Right? And so if you plug in this approximation for y, then you'll get something here that will be some g of t that's generally not going to equal zero, but you would like it to be zero. If you put the true solution in, you would get zero. If you put your approximate solution, it is not going to be zero. So you'll get g of t and you want this equal to zero for all t.

And then in addition, you have boundary conditions. And so you have boundary conditions on the solution and you want those things to be satisfied. And again, you can write them in a form that something has to be zero. Often, you'll exactly satisfy them. So you'll choose your solution to make sure it satisfies the boundary conditions and it won't satisfy in the domain. It won't really satisfy the differential equation. That's the most common thing to do. But it could also be that it doesn't satisfy the boundary conditions either, but you want to be close to satisfy the boundary conditions.

We have to think of how are we going to judge if the solution is accurate or not, if our approximate solution is good? And from the way we wrote it, we have our parameters, d. These are numbers we can adjust. And we're going to try to adjust them to make the solution as good as possible. And now we just define what a good means.

There are several definitions of good that are widely used. And one of them is called collocation. This is like option number one. And that is you choose a set of ts, of particular time points. And for those particular type points you demand that g of the time point is equal to zero. So you're forcing the residuals-- this is called the residual, it's the error. And you're forcing the error to be zero at some particular time points.

And generally between the time points it will not be equal to zero. But you can pick your time points and depending on which ones you pick, you'll get a slightly different optimal choice of the d's. Because the d's will be adjusted to force the residual to be zero at your time points you pick. So that's one option.

Another one is called Rayleigh-Ritz. And that one is you minimize overall your d's. The integral in t zero to t final of the norm of g of t. So this means you try to make the average of the square of that deviation to be as small as possible. So it's like sort of like a least squares fit kind of thing. So that's another option.

And then a third one that people use a lot is called Galerkin's Method. And his method is you choose some functions-- some of your basis functions typically-- and you integrate them with each of the elements of the residual. And you demand that that has to be equal to zero. Yeah?

AUDIENCE: [INAUDIBLE] for [? this method, ?] what do you [INAUDIBLE]?

PROFESSOR: The d's, your coefficients. And these ones, I didn't write it out but this g depends implicitly on the d's, so you can write it that way. A lot of d's. And so you optimize the d's, you very the d's to force g to be zero at certain ts. And this one also, the g depends on d's. And so you're going to optimize the d's to force this integral equation to be satisfied. Yeah?

AUDIENCE: [INAUDIBLE] all changing d's?

PROFESSOR: Yeah. They're all changing d's. And it's just trying to-- your criterion, your error measure, how do you measure or what do you think is good? You want to make something small, some kind of error small, but you have to figure out what are you going to define your error to be. And you'll get different solutions depending on which error measure you use. Is this OK?

Now this one's pretty straightforward. I'm just going to write it down a bunch of algebraic equations that depend on my d's and then I'm going to solve them. And this looks a lot like [? an f ?] [? solve ?] problem. This is a Newton problem or something like that. So it's just some algebraic equations. Should be OK?

So that one you should be pretty well set up to do right now. Let's just think of how many equations there are. So we have-- say we have our basis set yf t dni fit. That's our basis set. And this is a sum over i equals one to say the number of basis functions. What do you want to call that? N, sound good? So we have n basis functions. And so we have how many unknowns here? Maybe n's not good because n's this one. Let's call it something else. k, OK. All right? In fact, I'll even change this to k too just to make life easier.

So there are how many dnk's are there? There's n where n is the dimension, the number of od's or the number of components in y times k. That's how many d's we got that we're going to adjust. We want to have an equal number of equations and unknowns. We have to have as many equations as that. So what equations do we got? How many equations?

So if we just have a ODE [? BBP ?] we typically have n boundary conditions. So that many boundary conditions because we need one boundary condition for each differential equation, right? One integration constant for each one. So we took an n boundary conditions. And then we have-- in collocation, we have however many capital M time points we chose. So we have M n equations-- no, that's not right. We have an equation like that for every component of g.

So it's M times n equations from the collocation. Is that all right?

So just looking at this, it looks like we have n times M plus one equations and we have n times k unknowns that we're trying to adjust. And so therefore, this says we should choose k to be equal to M plus one. So if we choose we want to say we want 100 basis functions, then we need 99 time points to do collocations at it in order to exactly determine everything. Is that OK?

How many people think this is OK? OK. He agrees. It's OK. The rest of you, no opinion. This is like the American political system. Only 5% people vote. Everybody else just listens to Donald Trump.

So this is how many equations we need. Now sometimes people will choose the basis functions so that, say, some of the boundary condition equations might be satisfied automatically. And I'll talk a little bit in a minute about cleverness in choosing basis functions. So one possible thing is you can try to cleverly choose basis functions so that no matter what values of d's you choose, you're always going to satisfy some of the boundary conditions. And so long as you don't get a full value of n because some of these boundary conditions don't help you determine the d's, any values of d's will work. And so in those cases, you need a few- - might need another time point or something. Get some more equations.

So that's collocation. And I think this should be perfectly straightforward. You just evaluate your y's. You need to have your dy/dts. You'll need them, because they appear in g as well. And they're just going to be the summation over k [? of ?] [? dnk ?] v prime k of t.

And so you choose basis functions that the analytical derivatives of. So now you know these answers. And now you can evaluate this at any time point tm. So you evaluate your time points. You evaluate these guys at your time points. You plug them all into your g expression over here, and your force is equal to zero, by varying the d's, right? No problem.

So that's collocation. That's pretty easy. And because it's so easy, it's kind of pretty widely used as one way to go. All it requires is that you have to know the derivatives of your basis functions. In particular, doesn't require any integrals. When you see the other methods, they're going to involve integrals. So we'll have to figure out how we're going evaluate those. So if you have functions you don't know how to integrate then this is definitely the way to go, to do collocation.

And if you use collocation with enough points, you're forcing the error to be zero a lot of points, then probably it won't get that big in between the points. At least you can hope that I won't get that big between the points. And you can try it with different numbers of points. And different size numbers of base functions and see what you can do. See if it converges to something, if you're lucky. All right. So that's one way to go.

And it's just f solve problem. And all that's happening is it's just forming a Dracovian Matrix inside there. Now, you still have to choose which basis functions you want. And there's a lot of basis functions you know that you know the derivatives of.

So there's some issues here about what choice is best. And we'll talk a little about that in a minute.

One thing you have to watch out for, is you don't want your basis functions to be linearly dependent because then you'll end up with indeterminate values a d. Different sets of d's will give you exactly the same y. If these phi's-- if one of these phi's is a linear combination of other phi's in your set, then you could have different values of d's that would actually correspond to exactly the same y.

Because of that, when you a Jacobian Matrix as part of it your Newton solve, the Jacobian's going to be singular, and then the whole thing's not going to work. So that's one thing to watch out for. And actually, it doesn't really matter if the functions are orthogonal in the sense of being functions orthogonal, it's really whether the vector, vk evaluated tm, if those are independent of each other or not.

I feel like we talked about this before, one time, yes? Yes. All right.

So anyway, as long as the vk evaluate of tm, if those things are vectors which are not linerally dependent on each other, this should work fine. OK? All right. Now, really [INAUDIBLE]. This one here often gets to be kind of messy. Part of it is that inside this norm of this vector, it's square, so you end up getting squares of your d's. So, the whole thing is definitely non-linear in d to start with.

And then you have to be able to evaluate all the integrals that come up. So this is really sum over n, of g and of td squared that you're trying to integrate. [INAUDIBLE] Right? So, you have a lot of these function squared. You have to add them up. You could do it, but the algebra gets a little complicated. So, this is not so commonly done unless the g has some special form that makes the algebra a little easier.

But there's no reason you can't do it in principle, but it just is a little bit of a mess because you get a lot of integrals of cross terms between the g's. And in the d's inside there. And so, not so commonly done. Though one case where it is done a lot is in actually quantum chemistry. Methods called variational methods in quantum chemistry use real [INAUDIBLE] to figure out the coefficients, and like your orbitals, and stuff like that. So, it'd be certain methods. But they've actually gone out of favor recently. So, you probably won't use it that often. But in the past, that was a big deal.

Galerkin is used the most of anything, so let's talk about that one for a minute. Now, the concept, where does this equation come from? I guess that's one thing. So maybe we can back up and say, well, where did the collocation come from? Collocation is actually like a special version of this where I choose, instead of these basis functions, I use delta functions.

So, if I integrated with delta functions, t minus tm. Another way to look at the collocation is demanding that the integral of delta function t minus tm g n a t dt is equal to zero. OK. So, now instead of using delta functions, I'm using these functions. Basis functions. So I don't know if it's particularly obvious why you'd use one or the other, but anyway, you have a choice.

And any time you do this kind of integral with some number of functions, you get some number of equations that can help you determine the d's. This particular choice of the basis functions-- one way to look at it is, you can say, well, suppose my solution g g n, suppose I could write this as a sum of some different expansion coefficients. So I can relate this plus

OK, so, there's two terms. One is the expansion of the residuals in the basis that I'm using, and one is all the rest going on to infinity of all the other basis functions in the universe. And if I think my basis set is very complete, that it kind of cover all the kinds of functions I'm ever going to deal with, both in y and in g, in the residual, then this would be a reasonable thing, and you might expect that this term-- if you'd pick big K big enough, this might be small.

So that's sort of where the idea is. So then, if I think this is small, then I want to make sure I make this part as close to accurate as possible. And this condition is basically doing that. It's saying that I want the [? error ?] to be orthogonal to the basis functions, in the sense that, for two functions, the integral of the two functions like this, is like an inner product, just like the inner product between two vectors.

So I'm saying, like, in the vector space, in the function space that I'm working in, I don't want to have any error. That's what this is saying. But I'm going to have is that there's other g terms which are the rest over here. These guys will not be orthogonal to the first part. Does that make sense?

So there's still some error left, in my g. But the part in here, I can make a good [? answer. ?] So that's where this equation comes from conceptually. All right. And we'll come back-- when we do least square [? setting, ?] we'll come back to that same idea. So this is what the equation looks like, and the disadvantage of this one is it involves some integrals. And so you have to be able to evaluate the integrals. All right.

So that's the downside of this function. So cleverly choosing using your basis functions to make it easy to evaluate the integrals is like the key thing to make this a good method. And just like we needed analytical expressions for the derivatives here, now we need analytical expressions for integrals. That we're going to get from this guy. OK. So, let's think about what basis functions we can choose that might make it easier to evaluate the integrals.

Suppose we can write g explicitly. Like, this. So, g is equal to d gx dyn/dt minus f/n. Suppose. OK? Then what integrals am I going to get? Well, dyn/dt is the sum of the derivatives of the basis functions, and y is just this sum up there. Some of the basis functions. And so, I'm going to have integrals that look like this is like I have an integral phi j summation d phi prime. No, k. t. dt. That will be the integrals from the first term. And then I'll have minus some integrals from the second term here.

So, phi j fn ft y, where y is the sum-- it's actually like a matrix. All right. Because this y is a vector. It's all the yns. And so, the whole y vector is d times the phi vector. All right. Where d's that big matrix. The elements are [? dnk. ?] And so, if you cut these integrals, this thing here is a summation of [? dnk. ?] The integral of phi j. Phi k prime. And so, if you choose your basis functions cleverly, you might be able to know all these integrals analytically. OK?

But if you don't choose them cleverly, who knows what kind of horrible mess you'll end up here. All right. Ideally, you really want to get these all analytically so you don't have to do numerical integration. As a loop inside, all the rest of the work, you're going to do in this problem. OK? And then, these ones also, now knowing something about this is really important or you may have to do the numerical integration here, because this could be really a mess.

You have a lot of phis inside some function, which could be a non-linear function of these guys. And so, in principle this could be really horrible. So you may have to do numerical quadrature for those guys. All right. OK, and so, if you do Galerkin's method, a big part of that is thinking ahead of time, oh, what basis function am I going to use? How can I make a basis function so I can evaluate the integrals easily, and then I might have a chance to do it? All right, questions so far?

AUDIENCE: Phi j

PROFESSOR: Phi j. Sorry, phi j of t. Here, I'll get rid of the t's. These are both functions of t. You're integrating over t. And these integrals from t0 to to t phi. Your domain. All right. Now in Galerkin's method, in addition to these integral equations, I still have the integrals, the equations, that the boundary conditions.

So I still have some equations that look like collocation equations that are evaluated at tm. Do I have that anywhere? Nowhere. I have an equation like this, except it's not g, it's q, it's the boundary condition equations have to be true at the boundary conditions. Right? So they're the same as before, q [? or just ?] before qn of dy/dt evaluated at tn y of tn. tn. This is equal to 0. This is sort of the general way to write a boundary condition.

And so there'll be some special ends, the boundaries, where I want to have extra conditions. I'll get some equations like this. These have to be satisfied in Galerkin's method as well and they will not be integrals. They are just that, tn. And so in addition to the integral equations, that you have to solve over here, you want these things to be zero and you also want to satisfy the boundary conditions.

So then, all these methods, a big part of it gets to be cleverness of a basis function, a choice of basis functions. And there's kind of two families of approaches. So, one family is global basis functions. So basis functions that are defined on the whole domain. And there are special ones, sines and cosines, Bessel functions, all these functions inferred from your classes, all the special functions. And a lot of them, the integrals are known for a lot of cases.

There might be special tricks that make the integrals easy to evaluate. They can satisfy the boundary conditions automatically. And some of them, for example, many of the problems we have with like the heat flow equation, so you have-- actually it won't be d. I'm probably going to get this wrong. It's, like, kappa or alpha. Which one is it? Alpha. Alpha d square of td, x squared minus, there's some source of heat that might depend on the temperature, and this has to be, say, equal to zero.

Right, does this seem what you've seen in classes before? So this is a common one. So, in this case, you might try t to be a sum of, say, sines. So [? dnk ?] sine of k something something something. In there, and the cleverness of this is that d squared phi k dx squared is going to be equal to some number times phi k. Because the second derivatives of sines are also sines. Right? So all your differentials will solve and in fact, the derivatives will be really simple. Now, the q terms could still be a horrible mess.

For example, this could be an Arrhenius thing where's the t's up in the exponent. And then the whole thing might be horrible. Anyway. But at least the differential part is, like, super easy. OK. And you might have a boundary condition say, at one end, that dt/dx at someplace like d final x final is equal to zero. Right? That might be like, you're up against a x insulator. Or something like that. So, you don't have a heat flow. So that would be a boundary condition you might have. And then, by cleverly choosing your definition of the sines, you could force that all the sines satisfy this derivative condition.

OK, so, rescale your coordinates so that ends up with pi over 2, an all sides of everything and pi over two is always zero, the derivative, so you're good. So, you can do some clever trickiness to try to make the problem easier to solve. And so, that's one whole branch of these basis function methods, is clever basis functions to match the special problem you have. OK? So that's one option. Then, the other option is the non-clever approach, where you just say, well, I've got a computer.

Who cares about being clever? Let's just brute force it. All right. And instead, you just want to write a general method, any problem you can solve. And this is more or less what COMSOL does. And you're going to do it. And so the distinction here is that this kind of thing, this sine function, has a value everywhere, all across the domain. So this is kind of like a global function. OK? And the alternative is to do local basis. Try to have basis functions that are only defined in little tiny areas.

And then, at least when I integrate the integrals, I don't have to integrate over the whole range. I don't have to have to integrate right around my little basis function. So, this global basis function-- this is sort of similar to what we're doing, interpolation? Do you remember you could use high order polynomial to interpolate, or you could alternatively do a little piecewise interpolations, say, with straight lines between your points. And, you know, it's not so clear, actually, which would be the best way to do it. Or maybe you want to do some combination. Do little parabolas between little triples of points, or something.

That might be a good interpolation procedure. It's the same thing here. Some problems, you can find a global basis set that works great and you should use it. In other problems you might do better to just break up the domain in little tiny pieces and then do simple little polynomials or something in those little domains. All right. So this is the global basis. Let's see a local basis. It's OK to delete this? So, local basis functions. A really common choice for these guys are the b-splines. And, in particular, first order of b-splines and these functions have the shape phi phi k, it looks like this. So, at zero, all the way up to ti minus 1.

Then it goes up to one at ti, and then it goes down to zero again, and goes out. OK. So this function is a b-spline. It's also called a tent function because it looks like a tent. Some people call it a hat function. I don't have a pointy head so it doesn't look like a hat to me, but they call it a hat function so I guess they must have hats like that.

And this is a very common basis set. And the nice thing about this basis function is, it's zero except in this little tiny domain around it. OK? And it's the only basis function in the whole set that has a non-zero value of ti. And so if you want the function to equal something in ti, it's going to be equal to the coefficient of this basis function, right? Because we're going to write y n of t is equal to summation [? dnk ?] phi k of t. And so, if I really care about yn evaluated at ti, the only one basis function this whole sum is going to have a non-zero value there. So that's going to be equal to dn dnk, k where this is the special k.

k prime matches up to ti. OK, so there's one base function that looks like this. There's another one over here that's the one base center on this point, and there's another one over here. So on this point, it's [INAUDIBLE]. And I have as many basis functions as I have points. So this is like a way I discretize the problem, but I've kept my solution as continuous function because my y-- that's the sum of these guys-- has a value everywhere. It's a continuous function. The way this is written it looks like it might not be differentiable at all the points because it has all these kinks, but there's a clever trick you can do to deal with the kinks. So, actually, it's not a problem.

AUDIENCE: So, for these basis functions, how would you define if you had the boundaries? Like t0?

PROFESSOR: So, you'll have a basis function at the very end. Suppose this is t0 here. You have one like that. OK, so it's just a half of a tent.

Can you get a [? sparse ?] d-matrix? Yes. So locality is really good because it makes the Jacobian matrix sparse, the overall problem. So, when I compute the integrals of this thing, for example, the integral of phi i of t, phi i minus 1 of t dt, this turns out to be equal to 1 or 2 times ti minus ti [INAUDIBLE]. No. I take that back. Just one half. Just half. So it is like a brilliant thing. Is that right?

It does have a ti [INAUDIBLE], not a [INAUDIBLE]. I'll have to double check. Possibly including delta t. I can't remember. All right. But the integrals are very analytical, and only certain ones are non-zero. So only one that the two is when the two is differ by one unit, do they have a non-zero integral? All the rest of them are non-zeros.

So, when I write down these equations, I mean, many, many, many, many zeros, so it looks horrible when I write Galerkin's method, I get so many integrals, but actually a zillion of them are zero, and then there's a bunch of special tricks I can do that make it even better than that. OK? So then, the Jacobian sparse.

So, that'll save you a lot of time in linear algebra, which allows you use a lot of points. So, you can use a very large basis set because you end up with sparse Jacobians, I mean, it's not going to fill your memory storing all the elements in the Jacobian, even if the number of points is very large. And also, there's vast numerical solution methods for those. So that's the idea of this.

I guess should we try to carry one of these out? You guys are [INAUDIBLE] trying to do a lot of algebra on the board? Do you think I can do the algebra on the board? That's the real question. All right. Should I do it for collocation, or you guys confident you can do collocation? You're all right with collocation? You're fine with collocation. Great. OK. So we'll just go right into Galerkin.

So, Galerkin. That way, we use a local basis. So, most of the equations we have to solve are this type. Phi j t times the residual function of summation [? dnk ?] phi k prime. Summation [? dnk ?] of phi k and t. Is equal to zero. All right, we have a lot of equations like that. We're trying to find the d's that are going to force all these integrals to be zero. OK? So we have a lot of different j's we're going to try. We want this to be true for all the n's. And then we're going to adjust these [? dnk's ?] to try to force this to be zero. All right? That's the main problem. And then, on top of this, there's some equations with boundary conditions.

So, now we have to look at what the form is. If the form is, oh, I erased it, if I can make this explicit in the derivatives, which I can do very often, for example, this could be dyn/dt minus [? fm. ?] So then, dyn/dt is just this. And then I'll have another term, [? fm, ?] which will depend on that [INAUDIBLE]. And so the integral phi j, and I'll just remind you that phi j is not equal to zero. If tj minus 1 is less than tj plus 1.

All right. That's the only places where my local basis function is non-zero. That's where the tent is. And all the rest of it's zero. So when I have this integral, originally have it t0 to t-final, but actually I could replace this with tj minus 1 to tj. And it's just the same. Because of all the integral outside that domain is zero. Because this function is zero. All right? So at least I have a small little domain to do the integral over.

AUDIENCE: [INAUDIBLE] tj?

PROFESSOR: tj plus 1. Thank you. Yes. Yes. Plus 1. Yes. Is that all right? OK. So, I have this integral, and then I have this times the derivative term, so it's dnk dk prime minus the same integral, j minus 1. J plus 1. Phi j sorry [? fm ?] [INAUDIBLE] summation of dn. k phi. k [INAUDIBLE] t. OK. Now, this derivative of phi k, well, I just told you what it was. It's just like the [INAUDIBLE] set functions.

So, this integral, you'll know analytically. Like, it's either 0 or it's 1/2. Maybe minus 1/2. Whatever, but it's nothing complicated, and you'll just know it right off the bat. So this is all that can be known. And sparse. It's only going to be when k is equal to j minus 1 or j plus 1 that this will be non-zero, and all the rest of it would be zero.

OK? So, that's mostly zeros. This one over here, in principle, I have quite a huge sum here, k equals 1 to k, where I have all my basis functions, which is all my points where I put my little tents down. So I have a domain and I've parked a lot of tents all on the domain, that's my basis functions. And I'm trying to figure out, sort of, how high the tent poles are. On all those tents. And my tunnel functions, the sum of all the heights of those tents. OK?

But this guy is only non-zero in this little domain, and these guys, most of them are zero in that domain, because they're mostly tents that are far away from where my special tent phi j is. OK, so I'm going to draw a picture. Here's the domain from t0 to t-final. Over here is my tent corresponding to phi j, which is centered around tj.

Hence, this function. And then, these guys are all the other tents. There's a tent here, there's a tent like this, there's another tent like this, another tent like this. All these tents. Those are all the basis functions starting from k equals 1 and going up. All of these guys are zero in this domain because they all have zero tail.

So, almost all of these, the f's doesn't really pick up anything from any of those guys in the domain I care about. The only domain I care about is this domain right here. And this whole sum is all zero except for a few special k's when k is equal to j minus 1, j, or j plus 1. So I only have to worry about three terms.

The inside contributing to the y in the region of t that I care about. All right? So, when I want to compute the Jacobian with respect to d, the only terms here where the Jacobian's going to be non-zero are for d I have a Jacobian, which is the derivative of this whole thing, ddd nk. Right? And that's only non-zero if k is equal to j minus 1 j, or j plus 1.

You guys see that? How many people do not see this? How many people are lying? This is a really important concept for the locality, so this is the advantage of a local basis, that the Jacobian will turn out to be really sparse because it's only non-zero in this special case. And you might have 1,000 points, 1,000 of these little tent, 1,000 basis functions in the sum. But only three of them are non-zero. For four each n.

So, actually, it's 3 times n. [INAUDIBLE] non-zero. Is that all right? Because it just depends on the k. Three of the ks are non-zero. There's 1,000 of these guys [INAUDIBLE]. Yep. So that's the big trick of locality. And then, also, because these integration ranges are so small, because you choose your points really finely spaced, then you might get away with simple polynomial expansions of f, for example. Around the points, or [INAUDIBLE] expansions, there are all kinds of little tricks.

Or you could even do quadrature and determine some points, do a [INAUDIBLE] quadrature to evaluate the integral. But you only need a few points, because you know it's a little tiny range of dt. And you would hope that you've chosen so many fees, so many time points, that your function doesn't change much from one time point to the next. So, more or less, first order is constant. Your function's constant with respect to t in this little tiny domain, and then maybe has a little slope.

And then, if you're really being fancy, you might be able to put a parabola on it. But it's not going to change that much. If it's changing a lot, that's telling you you don't have enough time points and you should go back and put some more basis functions in. And then you can get a good [INAUDIBLE]. Because what we're, really doing here is we're using this basis set. If I add up these guys, what I'm really doing is piecewise linear interpolation between all the points.

So I'm approximating my y versus time. It's going to look like this. All right. It's piecewise linear, because that's the only thing you can make from adding up a bunch of straight line segments. A bunch of tents is a bunch of straight lines. And so this is what the approximate function is. Well, you can see, this is really bad if these functions are too different from each other. But if they're all like this-- and you think, well, maybe it's not so bad to use piecewise linear. Yeah.

AUDIENCE: [INAUDIBLE] confused at where we're using [INAUDIBLE].

PROFESSOR: Inside [? f ?] solve, what is trying to solve for the d's, it's solving each of this giant set of the equations, a whole lot of equations like this, they all come in to a gigantic f. That's the f of d that need to make equal to to zero. And so, I'm [? burying ?] the d's, I need the Jacobian of that gigantic f. Now, the problem, because it's gigantic, I need a lot of points, a lot of closely spaced points to make my function look smooth.

That means the number of base functions is really huge. So the Jacobian principle is huge number squared. I have 1,000 points, say, discretizing my domain. And then it's 1,000 by 1,000 Jacobian. It might be really hard to solve it, or cost a lot of CPU time, or use a lot of memory. But, fortunately, almost all the elements are zero. So, it's like, has a sparsity of 0.3% or something is the occupancy. So, it's almost all completely sparse, and therefore you can solve it even though it's gigantic. Yes.

AUDIENCE: [INAUDIBLE] basis function [INAUDIBLE] the Jacobian [INAUDIBLE] should be [INAUDIBLE]

PROFESSOR: We're trying to find d, so we're really trying to solve a problem that's f of d is equal to zero. That's our fundamental problem we're trying to solve. We're doing Galerkin. We have something equal to zero, and it depends on d. And we're trying to find the d's. So this is really the function we're trying to solve.

In order to evaluate the elements in this, we have to compute a whole bunch of integrals with the Galerkin method. So that's the complexity. But the basis functions, we know what they are. We've pre-specified them. So, the whole question is just what the d's are. And the d's get multiplied by a whole lot of integrals.

Is this all right? I think I may have misunderstood the question. So you have f of d is

[INAUDIBLE], so therefore we probably want to know j with respect to d.

OK. Time ran out before clarity was achieved. We'll try to achieve clarity on Wednesday morning.