WILLIAM GREEN: So welcome to our special class with half the people gone for tonight's hearing. You guys are the chosen few, so I'm pleased you're here.

AUDIENCE: [INTERPOSING VOICES]

WILLIAM GREEN: So I'm just going to say a little bit about models versus data and then I'm going to go back to undervalue problems and talk a little bit about how you handle really big ones. Yeah?

AUDIENCE: Are these going to be [INAUDIBLE].

WILLIAM GREEN: Yeah. Yes, sir.

AUDIENCE: All right.

WILLIAM GREEN: And in fact, these are even going to be on the video, because the guy just this fixed the video, so it will be recorded. All right. Yeah, I'm sorry. You said the video was broken last time so that recording didn't work. All right. So when we're talking about models versus data a really critical question is whether the model is really consistent with the data or not. And how do we know this is really true?

So we talked about how we can figure out some sort of chi-squared maximum that we would tolerate. So the maximum that we tolerate, we call it Q, capital Q in a lecture before. And so if we can find a chi-squared of theta that's less than this, then it's at least plausible that our model is true. Right? Or, alternatively, if all chi-squared thetas for any value of thetas are greater than chi-squared max, then we would say the model is false, or something else is really wrong with our experiment. Right? That's what we say when we're saying we have a maximum chi-squared that we would tolerate.

So a critical question with us is whether we really found the very best fit value of the parameters. In some simple problems you might have linear dependence on the parameters, and then there's only one best fit usually, as Long as you have a non-singular set of equations and everything we find there's no trouble. However, more commonly you have equations which are singular or close to singular, and also you normally have non-linear dependence between the model predictions and the parameters. So, for example, I do kinetics. We always care about the ray coefficients, the K's and they always come in inside a differential equation. Right? You always have like d concentrations dt is equal to some terms, and some of them are, you know, k concentration I times concentration j. whatever one it is, ij. Right? You have a lot of like terms like this by molecular reaction terms.

And these K's, some of them I don't know. And so these would be one of my thetas. So this would be, you know, theta 17 would be one of these K's that I want to determine. And when I have a differential equation like this, when I integrate it, the K is always going to come in non-linearly in the solution. Right? So when I integrate this solution, even in the simplest case you get things like c of t is equal to ae to the negative kt. That might be a really super simple case if you have a linear kinetics. The K is still coming in non-linearly to the predictions. Right? And if you have anything complicated you can't even write it down in a closed form expression, and you have to integrate with oed45 or ode15s and so the relationship between K and the concentrations that you measure can be really complicated. So it's always non-linear. In my world all the models are non-linear.

And I think there's many other situations like this. So if you try to measure a diffusivity, it might be possible to set up an experiment where you'd expect a linear dependency to something you'd measure in a diffusivity, but most likely not. So most likely it will come in non-linearly somehow and you'd have some model for it, and you'd have a non-linear dependence between that number and the numbers you would measure. Right? And almost all the properties are like that.

So because you have a non-linear dependence, then you have to worry about how many minima do you have in your surface? So chi-squared of theta is a non-linear function of a lot of variables. And typically it won't just have one minimum, you'll have a whole lot of local minima. Now one of those is the global minimum. And that's the one you want, that's the very best possible fit. But you normally won't know how many minima it has and so no matter how many minima you find you don't know if you're done, and you don't know if you really found the best fit. OK? So just because all the ones you found so far bigger than chi-squared max you might not be ready to write that article in Nature to claim that the laws of physics are incorrect because you're not sure you really found the very best fit. OK? So that's one of the many problems of trying to figure out if a model of the data are really consistent.

Now that particular one, you can overcome if you can find a guaranteed way to find the global optimum, to find the very best possible that there is. And some people spend their life working on this, and one of those people is professor Barton in this department. And so he teaches a class on how to do this, and there are actually methods to do it. So if you want to see the application of this to kinetic models there's a paper by me and professor Barton. And the lead author is one of his former students, Adam Singer and J. Phys. Chem. and it just shows how you can use global optimization to guarantee you have the very best possible via the parameters, and then they'll show you what happens.

So this is a case where the experimental data was measured by one of my students, James Taylor, and that's the thing with the error bars, and he did a lot of hard work to make sure he had pretty good error bars and he really knew what they were. And so he had those points. So he measured a lot of points. This measuring a lot of points is a key thing. So it means his degrees of freedom is very large. Because he uses many, many different time points along the way there and he has error bars on all of them. And so because the degrees of freedom are very large it actually makes it a very good constraint on the model. It tested very thoroughly. Where if you only measured one point, then that's not a very good constraint.

Now what James had done, and we published a paper in 2004 where he did it, where he said, you know, when I do my fits this is what I get. I get these kind of purple and green things, and therefore they're all outside the error bars, and therefore I publish in the Journal of Physical Chemistry, I claim this model is wrong based on my experimental data. OK? So we published that in 2004, it was accepted by the referees. It's in the literature, you can read it. Don't believe it though, because Adam singer went back and did global optimisation using the same model and his red line goes right through all the points really well. Do you see that? You can see the red line there.

See this red curve? That's the true best fit. And so all the fits that James had found were all local minima in the chi-squared squared surface. And when he actually found the best fit with Adam's fancy method, then it turns out that actually the model matched the data perfectly. So that's definitely true. So what we published the first time was completely wrong, because we said the model disproved the data. But in fact the model proved the data. I mean the data proved the model, or was completely consistent with the model in every way in great detail. So I was a little embarrassed.

So then James, also in the same paper, had found-- and they measured another condition in a slightly different case, and actually was looking at a different solvent. And in this case he found this green curve as his best fit, and it looked pretty good by eye, though you can see it sort of skittering on the tops of the error bar bands. And so this is one where he said, you know, we do the statistical test, there's like a 5% chance that you would measure data like he measured if the green model was the truth. And so we said some wishy-washy words in the paper, basically because we weren't really if it was consistent or not consistent with the data.

But in that case Adam used his fancy global optimization techniques and found that James' fit was actually not that far from the global optimum, and he got the red curve there, marked by the red arrow, and that one looks pretty good, it actually goes through the data. But it doesn't really do it right, and found that the confidence intervals are only 16%. So it's only a 16% chance, one chance out of six, that James would have measured the data he measured if that model was true. And now we're sure that we have the global best fit, so now we're, like, thinking well, you know, maybe we should say that the model is not true for this case. Because it's an 84% chance it's not true anyway.

So actually we got it wrong both ways. So in the first case we said that our data disproved the model. And in the second case, we said that our model was pretty good with the data, but actually they were both wrong. So one was wrong that actually-- the data did prove the models correct for the first case and it actually looks a little fishy for the second case. So anyway this is just a concrete example to make you be a little bit more worried about what you do when you do these fitting things.

Now there's a variety of ways to try to do the least course fitting to try to find optima. There was a guy named Carr from University of Minnesota. And he published this [INAUDIBLE] way to just try a whole lot of different initial guesses, and that usually works to find the global minimum. But if you really want to have a guaranteed global minimum that you're absolutely sure is definitely is the global minimum, then you really need to use global optimization methods and you really should talk to professor Barton. And for these kinetic models he actually has a distributed code called GDOC that will just find the global optimization as long as you don't have too many kinetic parameters.

I think if you go to about 6-- around 6 adjustable parameters it can handle. If you have more than that then usually typically not. Because the problem with global optimisation methods is that they scale exponentially with the number of adjustable parameters, or the number of things you're trying to optimize. Now on the other hand, if you have a model that you need to use more than six adjustable parameters to fit your data, then you might want to try to think of a different experiment. Because you know the famous saying is like, you know, give me N parameters I can fit an elephant.

So if you have more than six parameters and you're trying to fit some data set, probably if you do it right you'll find a fit, but it may not really prove anything. So from that point of view professor Barton's code, GDOC, is pretty convenient for doing these non-linear optimizations. So that's a code for non-linear optimization with differential equations embedded. And they can be stiff and they can be large, it doesn't matter. What really matters it's just the number of adjustable parameters, that's the cost.

All right now there was a really good question, that you asked I think, in class last time, maybe. Maybe the time before. And it was what happens-- so let's look at different case you get. So one case you get, you know, if chi-squared square, theta best, your best fit parameters is a lot bigger, think chi-squared max, then you're ready to say the model disproves it. So that one you're OK. If chi-squared of your best fit is less than, or significantly less than chi-squared max-- it has be much less than it, really less, not equal. --then you're really happy to say that the models are consistent. The model and data are consistent to each other.

And, in this case, what you would normally do next is determine the confidence intervals on the thetas that you adjusted. Right? And if you read the chapter in Numerical Recipes, for example, some of the notes online, it could tell you how to do this. Right? And so that was OK? It's OK.

So the case that's really problematic is you find chi-squared theta best, and it's less than chi-squared max, so you think the model is consistent with data, but it's pretty darn close to theta-squared max. And so then you have to figure out, well, the weird aspect of this is I now have-- here's my famous plot of theta one and theta two. So I have two parameters I'm adjusting. If they want theta two, here's my best fit. My best fit.

If chi-squared max-- if the curve of constant chi-squared max is something like this, this is chi-squared of theta equals chi-squared max. If it's like that then I have some big region of a lot of different theta values that give pretty good fits, and I'm happy to draw confidence intervals, like draw lines here and here, for example, and say that's the confidence interval on theta one, and I'm all right. OK? And it's sloped here so I might want to tell people about the co-variance between theta one and theta two. All right? But I know what to do, it's fine. Now suppose this is the case that's like this. Where chi-squared theta best is significantly less than chi-squared max. Now suppose instead chi-squared best was very close chi-squared max. So then here is chi-squared max. This is chi-squared of theta equals chi-squared max, this is the good case. Here's the bad case.

So in the bad case, there is some region of thetas that the model is consistent with the data, but it's very tiny. There's a little tiny range of theta values that would make the models consistent with the data. Now what's weird about this is that my best fit, it's not very good. The chi-squared is almost up to the maximum chi-squared I would tolerate. So it means my deviations between my model and data are pretty bad. Like this plot actually. OK? And so I'm, like, you know, it could-- it's possible. I was a little bit unlucky and my data set is pretty far off but still within the possible range. And so maybe the model data are consistent. But then when I try to compute the confidence intervals on my parameters, there's only a really small range of parameters that are going to give good fits here. Because the best possible fit is this red line. There's no choice of the parameters that's going to go through all the circles with this model.

So now I'm like, what's going on here? So if I went ahead and do the same way I did before, I draw my confidence intervals like this and say, OK. Say, wow, I've done an amazingly great job of determining theta one. I know theta one very, very, very precisely. And the same for theta two. Here's theta two, it's got to be in that range and I've really done a great job. So I'm awesome. But I'm only awesome because my original best fit wasn't very good. So this might make you feel a little sick in your stomach if you're getting ready to publish in nature. Right? And then if the reviewers are on their game, you might get something really scathing back from the reviewers.

So there's kind of two ways to play it at this point. One way is, you are absolutely sure that your equations are correct. You are Sir Isaac Newton, you have just written the law of gravitation. You are measuring the orbits of the moons of Jupiter, and you are sure that your equations are right. And so, you know, my problem is my antiquated telescopes of 1600 are not that great, and so I think I have maybe-- maybe I misestimated my error bars on my measurements. Maybe I have some aberration or something, the angles are off, right? My twisted telescope looks as if the star moves or something, right? OK? So that's one possible way to go, say I really believe the model, and so I believe this is the truth, and I really believe my chi-squared max to be what it is, and I double checked the error bars and I think it's true. So I think have done a great job and I should publish it in Nature because I have determined these parameters with 17 more decimal places than anybody ever could before. OK?

So that's one way to look at it. The other way to look at it is what's recommended in the Numerical Recipes book. And they say don't use-- instead of writing the equation to be chi-squared of theta has to be less than chi-squared max, they say when you're computing the confidence intervals, instead write chi-squared of theta has to be less than chi-squared best fit plus chi-squared max. So they say don't use a stinky little circle here, draw a nice big one around there. And what is this saying? I guess what this is saying is I don't believe that my poor agreement between the model and data justifies me declaring very narrow parameter values. So

I just use the kinds of parameter value confidence intervals that it would have actually computed a priori when I do my experimental design. I think that's really how well I can determine stuff. And I think something is wrong that my chi-squared best fir is so bad. Maybe I didn't do my global optimization right and so I have the wrong parameters. Maybe I mis-estimated my error bars and so my chi-squared are all off. Maybe-- I don't know what else happened. Something happened, and I don't believe I'm that good, so therefore I'm going to quote much broader confidence intervals, which means much less precision in my determination of parameters. And that's what the Numerial Recipe guys recommend you should do. But you know, it's up to you. And I see it both ways.

So the people who won the Nobel Prize for the dark energy, do you guys know about this? So if you ever saw the data there. So that's one where they said-- they were using Einstein's theory of relativity, General relativity. They say we believe Einstein, we believe that equations true. And we're not going to, you know, the fact that the data only matches when you choose a certain value of the cosmological constant, we're just going to go with it. And we're reporting that's the value of the cosmological constant. Now some other people might say that's kind of goofy theory. Einstein changed his mind six times about whether to put the cosmological constant in the equation at all, so I'm not so confident and I would report something different. But they went with their gut. They said we believe Einstein. They went with it. The Nobel Committee agreed, and they got the Nobel prize. So you know, you're taking your chances. The Numerical Recipe guys are more like engineers, they say, oh, we don't believe models that much anyway. Let's just use more conservative confidence intervals. Any questions about this? OK. All right.

So now let's change topics. And we'll change to [INAUDIBLE] splitting. So if you remember back when we were talking about boundary value problems, we're often trying to solve problems that are like this. All right. So this is the conservation equation for a species in a reacting flow situation. So you have some convection, you have some diffusion, you have some reactions. Now in my life I have a lot of models. I have about 200 species in them. So this K is 200 different variables. And then you have these equations, and on top of them you have [INAUDIBLE] equations for that momentum conservation and continuity. And so this is actually a pretty bad set of equations.

And typically the chemistry is stiff. Because I have chemistry time scales it'll be, like, in picoseconds, sub-nanosecond time scales. And I'm caring about simulating-- well in my case maybe I have an advanced new engine burning some fuel. And so we're running over a simulation of a piston cycle, maybe 10 milliseconds, but the chemistry is sub-nanosecond, so then you multiply it out. So that's like I'm going to 10 to minus two seconds down to 10 to minus 10 seconds. That's eight orders of magnitude difference in time scales. So I'm in a bad way trying to solve this equation. And I have 200 of these guys coupled to each other. And then I have to think about what my mesh looks like. And I'm trying to simulate inside a piston, actually I have a moving mesh because the pistons compressing, so the equations are kind of complicated even with what the mesh is doing.

And then I need a bunch of mesh points there because the-- do you guys hear the Kolmogorov scale? You guys learned this yet? It's a scale like the minimum eddy size if you have a turbulent flow. And that's pretty darn small also. So it might be sub-micron. And so I have a natural, lens scale that's like microns maybe. So 10 to minus six meters, but the piston diameter is couple centimeters, so again I have like four orders of magnitude between the mesh size I need to resolve the physics and the size of the thing. So I need about 10,000 points across the cylinder, but it's a 3-D problem, so I need 10 to the fourth, times 10 to the fourth, times ten to the fourth, so I need 10 to the 12th mesh points. And then I have 200 variables, say variables at each mesh point, so I have 10 to the 14th stayed variables.

So how much memory you have in your computer? How much ram? You have probably like 16 gigabytes. So you have ten to the ninth- tenth to the ninth bytes, is that right? No-- what's giga? Ninth? OK, so you have ten to the tenth. Sorry. Ten to the tenth byes, but I have 10 to the 14th stayed variables. So I have a problem. So then you need to figure out how you solve it.

So there's two schools of thought of how to solve these equations. The school thought which has actually been the most successful so far, but is not readily available, is to get the government to build you the biggest possible computer in the world with as much memory as possible. And then you can actually keep track of your state vector, all 10 to the 14th elements, and you reserve this whole computer for yourself for, your know, a month. And just run it. And you figure out how to parallelize all the calculations, and you just use an explicit solver. And so that way it's just straightforward. All you're doing is you're computing, you know y of T plus delta T is equal to something, and it's just an explicit formula ode45, fancy version of that. OK? And it's just algebra, and you just add them up and it's no problem. But this is a limited feasibility thing. This only became possible about three years ago when they built a computer that actually had enough memory to store the state factor. It's also-- if the equations are too stiff it still doesn't work, because the step size and time step you need to use is so small-- so say it's 100 picoseconds and you're trying to get it out to a millisecond, so you need 10 to the eighth something time steps, which means you need to double precision numbers all the way through. But if the government builds you a 64-bit 128-bit machine then you can do it. But anyway this is like a very special kind of approach.

So now most people, including me I don't have access to that kind of computer power. Actually if you want to read about that there's a woman who does that, who is Jacqueline Chen, and you might want to look on the internet and see about her. And somehow she became buddies with the people in charge of the super computers, and so they're happy to give her a month of, you know, the best computer in the world every year, and she runs some really awesome calculation that's exactly right. Solves everything, has all the state variables ten to the 14th, and that's fantastic. And currently that's the benchmark for how people test approximate methods for how to solve this. Because she actually has the full solution numerically converged. But this again is a limited applicability. So a lot of people are working on approximation methods to try to solve this equation if I don't have access to such awesome computer resources.

And the general idea of them-- well the two ideas. So one idea is somehow get rid of the turbulence. OK? And so what they do is they use a mesh coarser than what you what you would need to resolve the Kolmogorov scale, and then they use models for what happens for itty bitty tiny eddies. And those are called sub-grade scale models, and this whole approach is called large eddy simulation. And the idea is you keep track of the large eddies, the big recirculation zones that's modelled explicitly, but you don't keep track of all the itty bitty tiny swirling off little eddies that lasts for a little second, a little bit of a period of time before they dissipate because of diffusion. OK? So that's the concept.

And there's, I don't know, half of all of the mechanical engineers who work in large eddy simulations are trying to figure out how to do this. Maybe Jim as well, I don't know. No. OK, but at least he knows some of them probably. So that's one whole branch. It's to try to get rid of the turbulence. And basically that's trying to reduce the density of mesh I need. So trying to get rid of this-- use the spatial mesh. But the other issue is the time mesh. And the time is even worse, right? Because I said I have ten to the eighth time points and I only need ten to the fourth spatial points in the dimension. So in some ways, if I can do something with the time, that really can buy me a lot. I have eight orders of magnitude I'd like to try to reduce to three orders of magnitude if I could, that would be really nice.

That idea is to say, well let's separate the fast timescales from the slow time scales. And in fact all the fast time scales are mostly all chemistry, particularly after I changed the eddy simulation where I get rid of the really small spatial scales, then the time scale of diffusion over this big mesh is kind of slow. And so I can go to a time scale that's completely controlled by chemistry. So we already told you, how do you solve stiff differential equations? Use a stiff solver, right? And there are special methods, there are implicit methods, and there's specialized solvers. And what they do is they allow you to take larger time steps than you would be allowed to use with explicit methods where you would run into numerical instabilities. If you try to use the explicit method with a big time stamp, it's much bigger than the real physical time scale, you'll get some crazy oscillations. But if we use the stiff solvers you can get away with time steps that are much, much larger than the real physical scale. So that allows you to go from 100 picoseconds up to maybe 100 nanoseconds. Get rid of three orders of magnitude of time step by using a stiff solver. So that's pretty good, three orders magnitude reduction? Well go for it.

It's not only reducing the CPU time potentially, but it's even more importantly reducing the number of points you have to save, because when you run this calculation what Jacqueline Chen ends up with is a list, y at every x, xn, yn, el, times e, something like this. Alpha, I don't know. That's the output of her program. OK, so how big is this object? This is 200 times 10 to the fourth, times 10 to the fourth, times 10 to the fourth, times 10 to the eighth. Right? That's how many numbers she's going to get out of her simulation. So, actually, her biggest problem is not actually solving the simulation. As long as she can get the one month of CPU time she's good. She's got to solve. Her Problem is how to analyze her data. Because now she has this much data that she has to figure out to say something rational about it.

And so she has like a whole army of post-docs whose whole job is trying to invent new day interrogation methods. And also this data set is so large it cannot be stored on a hard disk. In fact it cannot be stored in the biggest farm of hard disks. So she has to, as the calculation is running, send the data to different farms of hard disks around the world and filling them completely up, and then sending them to the next one and filling them up. So she has fragments of this data set stored all over the place. And then if she wants to make a graph she has to have a special graphs software that looks inside this data farm, get some numbers, look inside this data farm and puts it all together in some way. So this is a pretty hard project. OK?

Now if I can use a stiff solver I don't need to use so many time stamps. So I can get this down, say 10 to the fifth at least I got rid of three orders of magnitude in my data. Right? So now I just need one data farm maybe, instead of, you know, using all the ones at Amazon and Google own or something. OK? So this is a pretty big advantage to do it this way. Now the disadvantage, as you know, is solving implicit differential equations is way more expensive, right? Because you have to solve an implicit non-linear algebraic equation at every time step. And in this case it can be a pretty complicated equation. Because if I have this many state variables, so 10 to the fourth, times 10 to the fourth, times 10 to the fourth, so 10 to 12th times 200, so 10 to the 14th at each time step, that's my current state. And I'm trying to solve a non-linear algebraic equation, 10 to the 14th unknowns, you need a pretty good solver. I mean, I know backslash is good and F [? solved ?] and stuff, but I really don't think it's going to work. So that's a really serious problem.

So the next trick is to say, OK, well this stiffness is only happening because of the chemistry. So I don't-- I could solve all of the finite volumes separately. So the idea is to split the problem up. I'm going to solve the chemistry using a stiff solver only at one point a volume at a time. And then I'm going to do the transport some other way. So what I'm going to do is I'm going to rewrite this equation as My unknown is dy to t, is equal to some kind of transport term and some kind of reactant term. And this term is local. All this in this equation, this is really same dy at position x depends on this at position x. Whereas if you compute these guys I need to do finite differences or something from adjoining finite volumes. Right? I have to do fluxes from the adjoining volumes. So these guys are sensitive not just to my local value of y at this finite volume, but also to the neighboring values of y.

But if I don't worry about them, if I break it up I have this local thing and this non-local thing. This one is local, it's stiff. This one is non-local, but it's not so stiff. And often it's linear or nearly linear. Because the transport terms are mostly linear. And so I can use-- I have specialized solvers that people have worked on for many years for each of these things. So for the stiff thing I have things like ode15s, I have DAEPACK and [? DASPAK ?] and [? DASL ?] and sundials and all those programs that people have written and spent a lot of effort to solve this problem. And similarly there's a whole bunch of people who have spent their whole life trying to solve the convection diffusion problems, the non-reacting flow problem. And so I can use specialized solvers that they have.

So I want to split this up. And what I really would like to solve is I'd prefer to solve dy/dt equals r and dy/dt equals t completely separately if I could. Because I can use specialized methods that are the best for each. So not to figure out, well this is not really the real equation because it has both of them added together. How can I handle that? So this operator splitting recipe is solve dy/dt equals to t for a while, and at the end of this process I'll have a [? yat ?] final.

And then I'll solve this equation with y equals y0. That's the equation system of solving, of transport. I do some transport for a while, I get some values for the y, then I take these guys and I solve the reaction equation starting from y equals y final. How to write this? One or two 0. OK? And I solve this one for a while and I'll get a different, you know, a somewhat different [? yrt ?] final. All right? And the simplest way of putting things is just keep going back and forth. Boom, boom, boom, boom, boom, boom, boom, boom. And in the limit where I make this t final minus t0 to be very small, then maybe I might expect that they might converge to the same solution as the couple of questions. All right? So that's the concept.

Now, if you do it this way it doesn't work very well. You need to make this delta t incredibly small to get any accuracy. However you can do a slightly more clever version, which is I saw dy/dt equal t starting from yt0, so y0, and I solve this to me y at t plus delta-t over 2. Then I solve dy/dt is equal to r, starting from y of t0 equal to, call this y star. y star and this I integrate out to y of t plus the full delta t. I'll call that y double star. And then I solve this one again, and I solve starting from y of t0 equal to y double star, and I integrate that out for another delta t over 2. And this one I called my y final. And that's the one I use.

So I take a half of a transport step, sort of like I'm starting to do the transport, then I suddenly--

oh, hold on. I got you my reaction. I do my reaction, then I do the other half of my transport after that. And if you can do the analysis of this, it turns out this thing converges to the true solution of the couple equations with second order accuracy. So you might make this delta t, if I cut it in half the precision increased by factor of four. If I cut if by a factor of 10 it increased by a factor of 100. So this is looking more promising as a way to do it. This is the most popular way to do it.

So this recipe here for operator splitting is called Strang splitting. Named after Gill Strang, he was a professor in applied mathematics department here. Former head of CIAM, the international body for applied mathematics. And also author of a fantastically great linear algebra textbook which maybe some of you guys might have seen some videos of his. So he invented this 1968, I was five years old. He didn't know that his invention was of any practical use, being a mathematician. So he just published that paper. Now in the meantime, this paper is like a citation classic. It's cited like, I don't know, 20,000 times. He had no idea. He never looks at citation list, he's a mathematician.

So I was working on this problem and-- let's see-- let's back up. So this is how he usually solves it. Right? We usually solve the problems, the-- splitting, sorry. Just be able to see it. And we like to solve it this way because I can use specialized solvers. I use a transport solver, I use a stiff solver here. This one I can parallelize because I can do a different ode solution at each finite volume. So if I have a big computer, a big parallel computer, say I have 10,000 nodes, I can solve 10,000 finite volumes simultaneously in this step, so the CPU time is reduced a lot because it parallelizes easily. So this is really popular. This is the main way that people do things. And then in the ideal case you do it with a certain guess of delta t, then you try delta t divided by five. And if you get the same result then you publish it. OK? And you've converged your solution.

However, what I found doing this is I was trying to find flames. I was trying to calculate the positions of flames. And what I found was that with the delta t's that I would try, sometimes I would get a qualitatively different solution depending on the delta t I chose. And so in particular what I found was that I was solving burner flames. So I have, like, a burner here, and it had a flame on top of it like this. And there's two importantly different cases. One is where the flame is floating above the burner. And one is where the flame kind of goes all the way and touches the metal.

And you've probably seen both those cases, like a bunsen burner or something like that.

Sometimes if you get the flow rates high enough, the whole thing lift off and be floating there. And sometimes if you slow the flow rates down they say that they will anchor to the metal. Some piece the flame will actually touch the metal. Usually it's like if you have a-- a lot of times you have a case like this. You have like a metal piece, metal piece, here's your mixture coming in, out here is the air, and a lot of times the flame will actually touch this metal piece here. You know, the flame sort of looks like that. Have you ever seen a flame like this? Anyways, this is a common thing, anchored-flame. If you just have a slow speed burner that's what you get. If you go high speed on the burner you can make the whole thing lift up and sort of dance around if you look at the flame. All right?

You could probably see the same if you look at a piece of wood burning. If it's burning well the flame will be lifted off the wood, you'll see a gap between the flame and the wood. If it's not burning that well you can see bits of the flame actually touch the pieces of wood. OK? So that's a very important thing for a commission guy. So was really-- to me this is a big deal. So when I solved it with one choice of delta t I get the solution. When I solved with another choice of delta t I get the solution, with a nice big gap. Now they both can't be right. And so I was a little bit alarmed because it's like, it's really different, right? So I thought, you know, I didn't really care that much about the dynamics of my problem. I just wanted to make sure that the steady state flame I ended up with was the right one.

So if you're interested in steady state problems this is a really important thing. You want to make sure that your solution really converges to the true steady state. So what I demonstrated primarily is the Strang splitting method is not reliable to converge steady state. Now I believe I made the delta t tiny enough, probably I would converge the [INAUDIBLE] steady state. But I don't know how tiny I have to go, and I can't really judge my convergence criteria very well, because at some point of delta t is it suddenly jumps from one solution to another solution that's qualitatively and completely different. So the convergence is not smooth and continuous, it's like it's converging, converging, converging, jumps to some other solution. And I don't know where the jump happens. And I don't know if I keep going further, will it jump to something else? So this makes me very loathe to publish the paper. Right? So I don't know. So I was agitated about that.

So I start working on trying to fix this splitting. The problem was that Strang is really smart. So his splitting method is really good. It's stable, it's pretty accurate, it's really hard to beat it actually. And so I published the papers and they were that good because they really didn't really improve it that much. But then I had a really smart post-doc named Ray Speth, now works in the AeroAstro department. And he came up with a different way to look at this. And he said, you know, if I had a problem that's like this, I can always add some constant to this and subtract the same constant from this, and I'm really getting the same problem. And so now the thing is, can I cleverly choose this constant so that, for example, my steady state solution is going to be good? So how to figure out what constants to choose? And so the methods where you add these constants are called balancing methods. So it's trying to balance the operator splitting scheme.

And what Ray's method-- pretty simple balancing method. So this is called simple balancing. Let's see if I have a slide that shows it. There we go. Here we go. Simple balancing. You choose a c to be the average of your right hand side at the previous time step. OK, now idea here is that as you're getting close to a steady state, the previous time step is not going be that much different than the current time step. And so this average number is going to stay more or less study as you as you get close. And so what you're really doing is adding the, sort of, the average of these two guys to each of them from the previous step, and that has the consequence of really changing the solution. So let me just show you.

So the left hand side is what Strang splitting does. The blue chips are the transport, and the green steps are the reaction. And what's happening is I did a half a transport step, then I do a reaction. The reaction sort of jumps me off the trajectory, then brings me back to the trajectory, then overshoots because I do a whole delta t step of reaction. And then I have to do a half step of transport back. And so I get from the black-- one black dot is the first point, and the other black dot is what I call y final. And that's showing what happens. And it just repeats like that over and over again. And it makes sense. If you're close to a steady state, then if I'd just take one of these terms, without the balancing theta transport, it's going to head away from steady state, because at steady state the transport and the reaction are balancing against each other. Typically you have something transporting in and is being consumed by the reaction. Or something is being created by reaction and is being transported away. And they have opposite signs. If I only put one in that has a certain sign, then I'm going to move away from the steady state. And then I need the other one to bring it back.

And the reason why Strang splitting has trouble is it's doing these huge excursions away from the steady state at each time step, with the hope that it ends up back at the steady state. Right? That's the way it's supposed to work out. And to second order it does, that's because it's pretty clever about how it works. But it does it's crazy excursions. Now if you have a non-linear model, the further you do excursions from the real trajectory the bigger the area you get. So if you do that balance method, you can see what happens is that when you get to steady state all the steps are just moving you along the steady state trajectory. They're basically maintaining the steady state, and that's because I added-- it's like transport plus half of transport plus r or something like that. And what is it? There's probably a minus sign here. And so I'm getting rid of half of my transport and I'm adding in half of this.

So I changed-- before I had dy/dt is equal to t, but now I have dy/dt is equal to 1/2 of t. I feel I need a sign to stick here. So half of t minus f of r, something like that. And the-- this can't be right. I must have a sign. Maybe this is not the average it's a half. It should be minus. It's like the-- these things should be added, and these guys are you cancelled out, so basically [INAUDIBLE] is going to be zero, which it should be as a steady state. So I'm sorry, that previous slide had the sign wrong.

This is what happens as the solution. If you do Strang splitting on the left you get more accuracy as you cut delta t, but it's only-- it's not getting any better. It doesn't matter if it's near steady state or away from steady state, it treats everything equally, the Strang splitting. In the balance splittings, as you get to steady state, all of a sudden the errors drive exponentially. You see this litte logarithm of the error? Logarithmic plot. So you get tremendously great convergence to the real steady state once you get anywhere close to the steady state.

Now the cost of this is that because the formula, which has a sign mistake I think, it depends on yn, it makes the whole thing more numerically unstable. Because it's like an explicit thing. We're using what happened at a previous time to correct our equation for future times. And so you are more likely to get numerical instabilities. So then Ray invented an implicit version of that, and also a higher order method, and that's called re-balance splitting. The equation is way too complicated for me to write down, since I can't even write down the minus signs correctly in this one. But you're happy to read the paper, it's in CIAM. And he has the equations for that.

And when you put the second order one in, that's implicit, then the thing is just as stable as Strang splitting, but it also has this great property that as you get close to steady state it exponentially converges to the solution, which is sort of like how Newton's method is really so good. It's sort of like that. You get a lot more decimal places really fast once you get close to steady state. All right, that's enough. Enjoy Thanksgiving. I posted a bunch of readings and things about metropolis Monte Carlo and stochastic stuff that we're talking about next week, and also a little bit more about models versus data. So do the reading when you get a chance. Say hi to your families, say hi to your friends. Have a nice time.