PROFESSOR: So this is our last 10.34 lecture of the year. And we're just going to use it for review. So I'm going to give a brief recap of the course. We've done a lot, actually. I hope you've learned a lot over the course of the term. We've had lots of opportunities to practice with different numerical methods.

I've checked in from time to time on the homework submissions. I've been really impressed with the quality of the work that the class has produced as a whole. It tells me that you guys do understand what you're doing, that you can apply these methods to problems of engineering interests.

And I hope that the things you've learned over the course of the term can be applied to your research down the road. Right, that's sort of the goal here. Whether you do experiments or modeling or theory, at some point, computation is going to play a role. You should feel comfortable after that things that we've done in this course on reaching back to these tools and utilizing them to either solve simple transport models of a drug delivery problem that you're working on, or fitting data reliably, doing hypothesis testing of models versus data. You have the tools now to do that.

The homework assignments that you've completed over the course of the term sort of lay out the framework in which that can be done. And you seem very competent. We'll test that competence on the final exam, which is Monday from 9:00 to 12:00 AM. It's not in Walker, though, OK. It's in 56-154. I'll remind you of that again at the end. But it's not in Walker, so don't show up there at 9:00 AM. You'll be taking the wrong final exam. It's in 56.

There's going to be a review session with the TAs on Friday. It's going to be in this room. It's going to be from 10:00 to 12:00 AM. And the TAs asked me to ask you to review your grade on Stellar. Make sure that everything is entered accurately. So check that your assignment grades match the grades that are entered there. Check that your quiz grades match the grades that are entered there.

You're going to take your final on Monday. And we're going to grade it on Tuesday and submit your final grades, because some of us have to travel right after the final exam period. So we want to make sure everything is as accurate as possible when we enter the final data. OK? Good.

OK, so a course recap. We've done a lot this term, actually, right? You should be really pleased with yourself for having made it this far. Any one of these subjects could comprise an entire course in and of itself. And you've gotten a broad overview of all of them.

So I'll remind you. We started with linear algebra. I told you the first day that linear algebra would be the underpinning of all the numerical methods that we utilized along the way. And that was true. I didn't lie. You really wanted to understand linear algebra well if you were going to solve any problems numerically.

We moved on to things that were more interesting from an engineering context, systems of nonlinear equations and optimization. So being able to solve engineering problems, being able to do engineering design by optimizing over various uncontrolled degrees of freedom is important. And we did both constrained and unconstrained optimization.

Then we moved on to dynamics. We did ODE initial value problems and DAEs, which are the combination of nonlinear equations and initial value problems. We discussed boundary value problems and partial differential equations, which largely are a class of boundary value problems as well, at least the ones that chemical engineers are typically interested in.

And we switched a little bit. We went into probability and models and data. And then we came back to more modeling in the form of stochastic simulation. And all along, this was threaded through with lots and lots of MATLAB. So hopefully, you learned to do programming if you didn't know that before.

MATLAB's actually a very useful programming tool to know. It's not so specific. There aren't specific quirks and details you need to understand like you would with a lower level programming language like C or Fortran. It's very high level, and the concepts that you utilize there can be translated to other programming languages as well. So you may find at some point in your research you want to do a lot of data handling.

MATLAB's not great for handling data actually. Parsing large data sets is quite difficult in MATLAB. But there are tool kits that are built into programming environments like Python that

are very efficient at doing that. And Python is also underpinned by similar numerical methods that are readily accessible. Everything you learned in MATLAB can be applied there as well. So you should have a broad understanding of how to draft an algorithm, how to modularize code, how to integrate several different numerical methods together to form a solution to a more complex problem.

So we did linear algebra. This was the basis of all the problem solving we did. Principally, it was concerned with the solution of systems of linear equations, though oftentimes, that's sort of disguised. It isn't a direct system of linear equations we're trying to solve. But somehow, we need to decompose a matrix. And in the process of decomposing it, we're essentially doing solution of systems of linear equations.

One thing we're really concerned with was how linear algebra operations scale with the size of the problem that we're interested in. The bigger the system of equations gets, the slower the calculation is going to be. And how slow can that get? This will introduce fundamental limits on what we can do computationally. When we try to solve partial differential equations, and we decompose them into linear equations, and we have huge numbers of states that we'd like to be able to resolve with a partial differential equation, the problem can get so complex so fast that a computer can't approach it.

So being able to estimate the computational cost of these operations is critical for deciding whether a problem is feasible or unfeasible. You'd like to know that before you start writing down a numerical solution to the problems. We focused a lot on that.

There was fundamentals like arithmetic, solving linear equations numerically using Gaussian elimination, for example. We discussed decomposition in the form of eigenvalues and eigenvectors and, more generally, the singular value decomposition.

And we discussed briefly iterative solutions of linear equations. That was sort of our segue into iterative solutions of nonlinear equations as well. So that was things like the Jacobi method or the Gauss-Seidel method, which were ways of the really operator-splitting methods of the same sort of class as we talked about in PDEs, where we took that matrix and we split it into different parts. We split it in a way that would make it convenient to solve a simpler system of linear equations.

And depending on the properties of the matrix, that solution might be achievable or not achievable. The iterative method may not converge. But for certain classes of problems, we

could guarantee that it would converge, and it might do it very quickly.

Things we didn't cover, but that are important, I think. There's a notion called linear operator theory, which is an extension of linear algebra to infinite dimensions. You've seen some of this already in your transport class, for example, where you solve differential equations with sets of functions.

You have some infinite set of functions that you solve a differential equation with, an ordinary differential equation or a partial differential equation. And all of the underpinnings of that fall under linear operator theory, which is an extension of what you've learned already for linear algebra.

We talked about Gaussian elimination and the LU decomposition. You'll find in the literature there are lots of matrix decomposition methods. And the ones that are chosen are chosen for often problem-specific reasons. So there's things like the QR decomposition or the Cholesky decomposition or the Schur decomposition.

These are technical terms. You can look them up. And you should feel confident that, if you read about them, you should be able to understand them given the basis in linear algebra you have now. But we didn't look at those in particular.

And the state of the art in linear algebra and solving systems of linear equations lies in what I refer to as Krylov subspace methods. We talked about conjugate gradient as an iterative method for linear equations. There's a broader view of that that says Krylov subspace method. You may see that in the literature from time to time. It's a more nuanced discussion of the linear dependence of columns in a matrix or linear independence and how you can take advantage of that to rapidly generate iterative solutions to those equations.

Any questions about linear algebra? No? Things are clear. You've done a lot of it this term. Systems of nonlinear equations. So these are, of course, essential for any engineering problems of practical importance. Linear equations tend to be easy to solve overall. Most engineering problems that are interesting are nonlinear by nature. That's what makes them interesting.

And we also saw later on that solving systems of nonlinear equations was intimately associated with optimization. And so we discussed fundamentals like convergence of sequences and the rate of convergence of those sequences, stopping criteria for iterative

methods.

The best sort of nonlinear equation solvers with multiple equations and multiple unknowns are Newton-Raphson-type methods. They're the ones that are going to, if we're close enough to a solution, give us quadratic convergence, which is often more than we deserve. It's exceptionally fast, but it's not a globally convergent sort of methodology.

So we also discuss quasi-Newton-Raphson methods that can help enhance the convergence of Newton-Raphson. And then we knew that we needed good initial guesses for solving these problems. We don't know how many solutions there are. We have no idea where they're located. And so we talked about homotopy and bifurcation as ways of generating those initial guesses and ways of tracking multiple routes or multiple solutions to the system of equations.

I mentioned briefly during that lecture the concept of arclength continuation, which is a more advanced sort of homotopy method. And we didn't get to discuss that in detail because, actually, the underlying equation for arclength continuation is a DAE. OK, it's a fully implicit DAE that one has to solve in order to track the solution path as a function of the homotopy parameter. So we didn't cover this. We didn't try to solve any arclength continuation problems. So it's a very common method that's applied to solutions of nonlinear equations.

Then we moved on to optimization. So we need this for design. We use this for model fitting. And we discussed both unconstrained and constrained optimization. In the unconstrained context, we derived optimality conditions. How do we know that we found the solution to this problem, at least a locally optimal solution?

And then we discuss steepest descent methods that could get us to that solution with linear conversions. We discussed Newton-Raphson methods that can get us to that solution with quadratic convergence. We discussed dogleg methods that are heuristic by nature, but meant to mix the properties of steepest descent and Newton-Raphson to give us reliable convergence no matter how far we are from the solution.

We moved on to constrained optimization. Those problems came in two types, right, equality constraints and inequality constraints. And we discussed the method of Lagrange multipliers to handle equality constraints. We discussed interior point methods to handle inequality constraints and their mixture to handle a combination of equality and inequality constraints.

We left out a lot. Optimization is a huge field. So there are special classes of problems called

linear programs, which are solved in a particular fashion. These are problems that have a linear dependence on the design variables. There's problems called integer programs where the design variables can only take on integer values. These might be very important. You can imagine trying to design a plant and asking what integer number of heat exchangers or separators is optimal.

Associated with constrained optimization and inequality constraints, there is an equivalent set of optimality conditions that we didn't try to derive. Their called the KKT conditions. So this comes up a lot in literature. So it's sort of the fundamentals of knowing that you found the optimum to the inequality constrained problem. They're not difficult to derive. But we have a limited amount of time in the class. It's something fundamental that's sort of missing.

And then these sorts of methods of all find local optima. But oftentimes, we're interested in global optimization instead. And that's not something that we discussed at all. That's quite complicated and beyond the scope of the course. But things like genetic algorithms, that's a term that will come up a lot. That's one way of trying to seek out global optima in a landscape.

ODE-IVP, right, so now we want to move from static engineering situations to dynamics. And we were really concerned, when modeling dynamics, with both the accuracy and the stability of these methods. We were concerned about stability with other algorithms, too, for solutions of nonlinear equations.

We were concerned with whether the Newton-Raphson method would converge or not. That's a stability issue in a sense. OK, so we're concerned with stability there. But here we really focused on it in detail, in part because some of these differential equations are inherently unstable. In engineering circumstances, that happens. And we would be unwilling to accept unstable solutions in situations where the equations themselves are stable.

In the case of Newton-Raphson, we can do these sorts of quasi-Newton-Raphson methods to change the path the solution follows. And we didn't really care what path it followed. But here, the path is important. We're not free to change the path. So the method itself has to be inherently stable.

So we discussed numerical integration. We discussed two classes of method, explicit methods and implicit methods. So explicit methods, we mentioned these explicit Runge-Kutta formulas that have been derived for all sorts of accuracy levels, both local truncation error and global truncation error.

Explicit methods, unfortunately you can't guarantee that they're always stable. They lack a property called A stability. Implicit methods, on the other hand, can be made to be stable under all circumstances. So one of those is the backward Euler method. That's a nice one to use. The penalty for using these implicit methods is you likely have to solve a nonlinear equation at each time step. So there is a cost to pay associated with enhancing stability.

We discussed local and global truncation error. And we talked about stiffness and linear stability criteria. So linearizing the ordinary differential equation, assessing what the eigenvalues associated with the Jacobian of that linearization is. And using those eigenvalues to tell us, for example, what kind of time steps we need with our method in order to achieve stable integration in time.

Things we didn't cover that are sort of interesting and maybe important in engineering context, one of those is event location. You may want to know the time at which something happens. So I may want to know when does the solution hit a certain point in time. Maybe, I even want to change my dynamics when the solution hits a particular point in time. There's a broader class of methods that can do that. They're built into MATLAB too, actually. You can set up these events and have the dynamics change or have a report that an event occurred. But we didn't really discuss that at all.

And in numerical computing, parallelization is important for studying classes of big problems. It's sort of funny, but you can even do parallelization in time. So this is something that's been discussed for a long time. And for certain classes of problems, it offers computational advantages over serial integration of the dynamics an ODE-IVP.

So you can imagine how do you parallelize in time. So you can take the time window that you're interested in and cut it up into different slices. And you could try to solve in parallel the dynamics over each of those slices. But for each slice, you need an initial condition that has to be precisely prescribed. And you don't know those initial conditions. That's what the serial integration would tell you.

One way of doing parallel in time is to say, well, those initial conditions are unknown. And they match up to the terminal conditions of the previous slice. So why not wrap this serial integration process in a Newton-Raphson solver to determine each of these points?

With certain classes of problems, certain ODE-IVPs that are inherently stable, you can do that.

And you can get a computational speedup over serial integration just by doing the integration for each of the slices on different computers and bringing all of the results together. The bigger the farm of computers you have, the more slices you can utilize. That's sort of interesting, right?

Parallel in time, it's bizarre, because we usually think of time integration as being sort of causal. I have to know where I was to predict where I'm going. But you can actually divide these things up in more sophisticated ways. We talked about BVPs and PDEs. We treated them as the same class of problem broadly.

Usually, when chemical engineers are looking at these problems, we're interested in spatial variation of some sort of a conserved quantity, so momentum or energy or concentration of a species or mass. And here we are also concerned with both the accuracy and stability of method stability now for when we had also time integration of the solution, in addition to spatial variation.

And so for BVPs, we talked about shooting methods. So can I recast a boundary value problem in one dimension as an initial value problem with an unknown? That's a very common way of solving these problems. And they are sometimes stable and sometimes not.

When they're unstable, you can do what's called multiple shooting, where you divide the domain up into subdomains, and you shoot from the initial condition of each of those subdomains. It looks exactly like the parallel in time integration that I described for you before. And then you try to match up the initial conditions with the terminal conditions to get a continuous solution.

Then we discussed relaxation methods for solution of BVPs and PDEs. So collocation, Galerkin. Collocation, you'll recall, was we want to try to satisfy the governing equation at various points in the solution domain. So we might write our solution as the superposition of a set of functions. And then try to satisfy at various points that we've laid out in the domains. Maybe, there's even an optimal place to put all these points in order to minimize the error in the solution.

Galerkin, instead, we tried to make the projection of the error in our equations on different orthogonal functions that represented the solution be zero. So that was sort of a global error that we tried to minimize over different orthogonal functions. Whereas collocation is sort of a local error that we're trying to minimize, a local truncation error that we're trying to minimize.

We discussed finite difference and finite volume methods as well. So ways of discretizing in the equations. Finite difference, I would say, is the easiest one to reach for. If you had to pick a quick and dirty method to try to get a solution, finite difference can be really easy to go to. It's easy to figure out how big the errors are in your finite difference method. But it may not be the optimal approach to the problem.

For certain geometries, finite volume is really far more preferable, because it's easy to estimate the fluxes through surfaces of funny shapes where your grid shape now conforms to your geometry. Finite volume also had the advantage of being strictly conservative. So I can never gain or lose a conserved quantity in the finite volume method, at least to within the numerical error. And that isn't true of finite difference or finite element. So if I'm concerned about those things, finite volume is really the approach to use.

And you'll find one fluid mechanical solver that's freely available. You can go download and use it today, and it is widely used in research, is OpenFOAM. And that's all based on the finite volume method. So they're trying to conserve momentum, conserve mass, and conserve energy associated with the fluid. OpenFOAM is good at drawing bizarre grids of the domain and giving you very accurate integration of fluid mechanical equations. And it's good at doing that because it uses finite volume.

You discussed the method of lines, which was a way of discretizing space, but leaving the time dimension undiscretized and then applying ODE-IVP methods for solving in time. It's an incredibly useful way to approach those problems. Because it leverages expertise in two different techniques. There are specialized methods for doing time integration with some spatial discretization that are vetted for their stability properties and their accuracy.

And those are fine, but I would say they're a little antiquated. They're reliable for particular classes of problems. But for general problems, you may not have any idea whether they're going to work or not. Whereas method of lines may be a really good approach to a parabolic partial differential equation you're working with where you can just rely on the adaptive time stepping and the air control and an ODE-IVP solver to keep everything stable in time as you integrate forward.

We did application of commercial software. So you used COMSOL to solve a problem. I don't know. Would you say was COMSOL easier to use than writing your own MATLAB code? Harder to use than writing your own MATLAB code, do you think? No.

AUDIENCE: Easier.

PROFESSOR: Easier to use. But the result, I would say, wasn't as good without some careful refinement of the COMSOL solution. So for that particular problem, if you wanted a solution that looked like your MATLAB solution that was carefully resolved in space, you really had to go in and refine the grid in particular places. So it gets harder and harder the more detail you want. But it's an easy way to get an answer. You should check whether that answer is right by comparing it with other methods. You can't guarantee that it's giving you the right result either.

To give you an example, I had mentioned an unsteady advection diffusion problem to one of my grad students. And he's very good with COMSOL. I mentioned that it was a hard problem because there were boundary layers. And the boundary layers changed thickness and time, and resolving those things can be quite challenging without any physical insight. And he says, well, I think I can just do it in COMSOL.

And he tried it, and COMSOL was happy to report a solution in time and space, but the solution was nonsensical. We could visualize it and see that it didn't look like what other numerical solutions look like, and in certain limits, didn't correspond to what those solutions are known to be in those limits.

So the black box is great in that it reports an answer. But it can also be really problematic as well. So maybe, you have expertise with COMSOL. You can get a solution to converge for a difficult problem. But you should really have other methodologies that let you assess the quality of that solution. For simple problems, simple elliptic problems and simple parabolic problems, it's going to do great, because it's built to eat those for breakfast. But for complicated engineering problems, you've got to vet your solutions.

Oh, what didn't we cover? We didn't talk about hyperbolic equations. Not really. We talked a little bit about advection equations, which have a hyperbolic character associated with them. So you have like a pulse of [? solute, ?] and it's advected along in time. And you want that advection to be stable, so you want to use things like upwind differencing to ensure that stability.

But there's a broader class of hyperbolic problems that relate to things like the motion of waves in elastic solids. We don't typically encounter those in chemical engineering, but chemical engineering is a broad discipline.

And maybe you're engineering soft materials for some sort of device. And you want to look at the elastic response of that material. The PDEs that govern that are a completely different class than the ones that we typically look at. They're not parabolic. They're not elliptic. They're hyperbolic, and they are these wave-like solutions associated with them. They require their own particular solution methods in order to be stable and accurate.

And then there are all sorts of specialized methods that are dreamed up to apply to, maybe, a broad class of problems, but it's kind of hard to ensure that they're going to work. But nonetheless, you'll see a lot of this out in the literature. So there are things called multi-grid methods. They try to discretize a partial differential equation with different levels of fineness. And you use course solutions as initial guesses for finer and finer solutions in space.

For some problems, this works great. And in fact, you can show that it can work as well as is possible. So depending on the size of the problem, say you have n points at which you want to find the solution. You can find it in order n time with some of these methods depending on the particular problem you're looking at. That's good.

But oftentimes, engineering problems aren't of that sort. There's something more complicated going on. And it's hard to ensure that you get those sorts of rates of convergence. But nonetheless, they exist and something that we didn't cover you should be aware of.

We did DAEs. So now, we coupled ODE-IVPs with nonlinear algebraic equations. Most dynamic engineering problems are of this type. We showed that DAEs are really an issue with model formulation ultimately. We write down conservation equations, and we write down certain constraints on those conservation equations.

The constraints can be specifications like we want particular flow rates in certain places. Or they can be fundamental. They can be thermodynamic constraints on the coexistence of different phases of a particular material in a particular unit operation.

If we could solve those constraints for certain unknowns and substitute them in the equations, we might be able to produce ODE-IVPs. Probably we can't come up with those solutions. So instead, we've always got this coupled system of initial value problems with nonlinear equations.

So you have that sort of a circumstance. How should you think about that? One way to think

about it was to say that the algebraic equations are essentially infinitely stiff. They have to be imposed exactly at every time step. That they never relax in time with the finite time constant. And so we knew that those sorts of methods are hard to resolve with typical ODE solvers.

And we tried to assess that by computing something called the differential index. So we asked how many times do I have to take derivatives of the algebraic equations or any new algebraic constraints in order to get a system of ordinary differential equations to represent the same model. I may want to solve that system of ordinary differential equations. Or I may just want to know the differential index so that I can choose a particular method to solve this problem.

We saw the higher the index was, the more sensitive the solution was to perturbations in different input variables. So the differential index played an important role in telling us something physical about the problem and the responses that could be elicited by DAE systems.

We also talked about consistent initialization. I need to prescribe the initial conditions for all the differential and algebraic variables. Can I prescribe any of those independently? Well, the answer is actually no. Depending on the differential index, there are underlying algebraic constraints that have to be satisfied at all points in time. And if they're not, then that can break the method that's trying to integrate the DAE. So you have to prescribe the initial conditions consistently.

If they're not prescribed consistently, and you go to a piece of commercial software, it may give an error. It may tell you nothing. You may get a reasonable result or an unreasonable result. It really depends on the methodology. So it's up to you to know in advance that this is an important aspect of the problem.

We solved index-1 DAEs typically. Index-2 DAEs can be converted into index-1 DAEs and solved pretty easily. Index-3 and bigger DAEs are difficult because of their high sensitivities. So there's special software that's out there. I mentioned some of it during our DAE lecture. And that's really what you want to reach for if you have a high index DAE.

Examples, we were able to craft all sorts of examples where we tried to do funny things with causality, where you tried to change the input by affecting the output, which doesn't make a lot of sense. But that led to very high index DAEs. Mechanical systems often have very high index DAEs associated with the mechanical constraints, typically lead to index-3 DAEs. So these pop up in places that are important. And being able to look at the model and assess the index of it

is kind of essential.

We did probability. So the physical world has inherent uncertainty. It's not just uncertainty in our ability to measure things, but fundamental, physical uncertainty is built into the world around us. So measurements have uncontrolled or even uncontrollable sensitivities to this uncertainty. And we wanted to know how to handle it. So we talked about things like sources of randomness. We did fundamentals.

Maybe this overlapped with 10.40. But I think overlap is good, because humans tend to have bad intuition about probability in general. So we discuss probability densities, means, covariances, expected values, joint probabilities, conditional probabilities. We talked about common probability densities. You covered the central limit theorem. You talked about the difference between sample averages and the population mean.

One thing that's important that I don't think that gets covered much in detail, but if you're going to be working with randomness computationally, oftentimes you have to generate random numbers. And you'd like to generate good random numbers.

Turns out a computer doesn't generate proper random numbers, only pseudo random numbers. And these generators have various properties associated with them. And some of them are good. And some of them are bad. And you want to use good ones. So if you have to generate random numbers for your research, you'd like to know that they are high quality, that the sequence of numbers that's being generated doesn't repeat over the time that you're using it, that it won't overlap over many different uses of this methodology. So high quality random numbers are important.

We discussed models versus data. So we wanted to regress, estimate parameters from the data. We wanted to do things like hypothesis testing. I have a model. Is the model consistent with the data, or is it not consistent with the data? So you covered things like linear and nonlinear least squares, maximum likelihood estimates, confidence intervals found using the chi-square distribution.

We did Bayes' theorem and Bayesian parameter estimation as well, so a way of using prior information about parameter values to better estimate the probability that those parameters will take on values given the data. We didn't cover design of experiments. That's a 10.551 topic. But there are specific ways that one can design the experiment. Which data values do I take in order to make the fitting of parameters optimal?

And there are other problems besides regression that are important. But we don't typically utilize those in chemical engineering. So one of those is classification. So if I have a series of points, and I do regression, I'm in some sense asking for like what's the best curve due to this model that can be drawn through those points, right? What parameter values give me the curve that's closest to all these points, that's least squares.

Classification might ask instead, what's the curve that divides these points most evenly, or sits furthest from all these points in an equal amount instead. Which side of this line do the points sit on? Are they of type A or type B? It's kind of a related problem to regression. But it's a different sort of problem. The same sorts of methods can be applied to classification that you apply to doing regression and parameter estimation. It's an important problem.

And then we finished up with stochastic simulation. So you learned that sampling of random processes can be used for engineering calculations. And they're even inherently random physical processes that you might want to model reliably.

So we did Metropolis Monte Carlo. This was a way of basically computing high dimensional integrals. That's what it boiled down to, where we wanted to sample from different places in the domain, weighted by some probability density function.

You did kinetic Monte Carlo, which is really a sort of event-driven algorithm. You are expecting that certain events are going to occur with certain rates. They're actually nonstochastic versions of that, the same sort of event-driven algorithm. I know certain things should happen at certain points in time. And rather than integrating the dynamics of the system all along these points in time, I just advance the dynamics until the next event occurs. That event occurs. I keep a list of what the next event is going to be. And I follow those events along.

So it could be something like billiard balls on a table. I know the trajectories of the billiard balls, given their position and their momenta. And all I need to do is keep track of when they collide and where they're going to go next and what the next events are going to be. Well, kinetic Monte Carlo was the same thing, except the events are now stochastic. They occur with prescribed rates. And so I just need the sample from this distribution of rates to get the right sequence of events.

There's funny things about kinetic Monte Carlo. So applied to chemical kinetics, you can estimate these rates reasonably well. You've got to know the rates for this process to look

right. If you don't have the rates correct, or you don't have the relative magnitudes of the rates, then the simulation is kind of meaningless. The result is going to be nonsense or garbage. And so there's going to be limitations on your ability to do this based upon how accurately you can estimate these rates.

It also means that if you know that there's disparities in the rates, say one rate is very high and one rate is very low, you can make certain assumptions about the process. So things that happen very quickly you may treat as though there are equilibrated and not try to resolve them at all and look more at infrequent processes that occur, slower rates.

So the precision with which you have to know these things is important. The relative rates are what's really important in these problems. And you want to get them right in order to resolve these physical processes correctly.

Then we discussed molecular dynamics after that, which is another type of stochastic simulation methodology. Here you just integrate the equations of motion associated with whatever particles you're interested in. Kinetic Monte Carlo and molecular dynamics are sort of fundamentally far from equilibrium, but may relax towards equilibrium, depending on what constraints you put on the particular rate processes that you're modeling.

Metropolis Monte Carlo is usually sampling an underlying-- if we're applying it in the statistical mechanical sense, it's usually sampling an underlying equilibrium Boltzmann distribution. So they often get used in fundamentally different ways as well.

We didn't cover stochastic differential equations. So depending on which degrees of freedom you want to represent in a model like molecular dynamics, there may be some degrees of freedom you simply don't care about. But they have an influence on the problem.

So in my research, for example, we model nanoparticles and particulates. We're not very interested in the details of the solvents surrounding these particles. There's a huge disparity in length scales associated with nanoparticles and the molecular dimensions of the solvent that surrounds them. The solvent has an influence nonetheless. It's fluctuating all the time. Solvent molecules are colliding with these particles, and they're imparting momentum.

One way of resolving a system like that without having to look explicitly at the solvent is to try to integrate out the solvent degrees of freedom and treat those collisions of the solvent with this particle in a stochastic sense. So at various points in time, there's momentum that's

randomly transferred to particles dispersed in a solvent.

The physical manifestation of that is Brownian motion. When you look at small particles in the fluid, you see them diffuse around, and that's where that's coming from. So you can simulate processes like that without having to resolve molecular scales. So as you go up in scale, you integrate out degrees of freedom, but oftentimes, that introduces inherent stochasticity in the underlying dynamics of whatever you're trying to model. That's very interesting.

And then for Monte Carlo methods, there's advanced sampling methods. And we didn't discuss any of these in detail. But you saw some of this with the model you saw with the problem you approached in your homework where you had two populations or two peaks in the probability distribution that were widely separated. And it may be hard for a Monte Carlo method to traverse between those peaks.

And umbrella sampling as a way of biasing the probability distribution to make those paths from one peak to another more probable. So it's just a way of seeking out rare events like the transitions between these two different populations. And it's pretty important, ultimately, for exploring complex high dimensional spaces. But you have to understand something about the underlying physics in order to figure out the biases needed to enhance the sampling. So it's so quite complicated.

So I'll remind you. Your final exam, right, it's Monday. It's in 56-154 at 9:00 AM. Don't show up to Walker. It's not going to be there. It's three hours. It's going to be comprehensive. You can expect roughly one problem each thematic topic. We've written it now. I think it's nine problems long. So we're in the process of revising it for difficulty and time. It's going to be short answer format.

We sort of aim at 10 to 15 minutes per problem. I think that's the sort of frame you should be looking at. The problems are structured in such a way that that's about how long they should take. You should use that to guide your temporal budget on the exam. If it's taking too long, switch to something else. Don't let yourself get hung up focusing on one thing. Move around and come back to it later.

There will be some fundamentals. There's going to be some translation of engineering into numerics. And there's going to be some MATLAB. So it'll look like a blend of the first exam and the second exam. Yeah.

AUDIENCE: When you say it's on MATLAB, do you mean [INAUDIBLE] MATLAB?

PROFESSOR: Yeah.

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah.

AUDIENCE: No computers.

PROFESSOR: No computers. It's just write a little script. We've asked problems like that before. It'll be of the same type. Yes.

AUDIENCE: You said nine problems. [INAUDIBLE].

PROFESSOR: I think there's nine, yeah. Well, that's how many we've written now. I'm not planning on writing anymore.

AUDIENCE: Plus or minus one.

PROFESSOR: Plus or minus one, yeah.

AUDIENCE: [INAUDIBLE].

PROFESSOR: Oh sure. Listen, if we gave you nine problems of the same length as on the first two quizzes, you wouldn't get through the problem. So they're not written to be so long. But there are multiple parts. But multiple parts are usually intended to guide you to the solution we're looking for.

Usually, we're testing for a particular understanding. It also helps you out when there are multiple parts. So there's lots of partial credit available. You don't have to get the whole thing. You've just got to get two out of the three parts. You did pretty good. OK, more questions?

AUDIENCE: So when you say short answer, does that just mean that similar types of problems as what we had on the first two quizzes. They're just shorter.

PROFESSOR: Yes, that's right. Yeah.

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah, they're not multiple choice, true or false. They're short answer. You want to hit a level of

detail that shows us you understand what you're talking about, right. We don't need pages of information on these things. But we need to have enough information that we can see you know what you're doing. We're trying to assess your understanding of the material. That's all.

More questions? No more questions, OK. Well, good. It's been a real pleasure teaching you guys. I'm really pleased with the outcome of this course. This is my third time teaching it. I think this is the strongest group that I've seen come through 10.34. So you guys should be really proud of yourselves. Good, all right, good luck.