Come up. Yes I am. Either really or not. Okay. So, um, shall we start? Yeah. How is everyone? Good. Busy with assessments? Yeah. Yeah. You have a lot of courses. Well, hopefully this one won't add that much in terms of workload, but, yeah, we'll see. It's still a couple or three weeks away. Right. The assessment deadline. So you still have some time to work on on this one outside of the other deadlines. Um, any any questions from the previous week. Um any complaints, anything that you are unable to to understand? Yeah. Are you able to grasp the concepts so far? Like you are comfortable with them? Yeah. Okay. Go a. Little slower. Slower. Sometimes I'm trying to take notes. Yeah. Okay. Okay. Yeah, yeah, I'll do that. Any, any. In any case, today what I will try to do is sort of recap a little bit what I spoke about. And so if there's something else that you need to ask me to bridge the gap, just feel free. Yeah. So today will be an introduction to multi-stage stochastic programming. Um, the concepts can be a bit, a bit complex to, to understand, but what I will do is I will try to build it in the same way that I built the two stage model, which is we start with some kind of naive approaches. We try things that are that we know don't work, and then we understand why they do not work. And then we sort of try to start building the concepts together. Okay. So, um. So firstly, I will recap a little bit what, uh, what we've dealt with so far. So we started with deterministic modelling. So I think everybody is quite familiar with with deterministic optimisation. Right. And we gave this example that involved uh the generation expansion planning problem. Can you put that? Yeah. Yeah. So we gave an example that that involved the generation expansion planning problem. Okay. So that was we are trying to minimise some cost cost of installation and cost of generation. And then we have some constraints on the number of generators to set up the capacity and the demand satisfaction. So the costs involved were the summation I in I summation t in t. Of the cost it of setup times x I t just forgetting the discount rate and all this just to not overcomplicate stuff right now. Um. Plus some generation costs. And times the amount that you generate. Okay. So that was the objective function. It was a minimisation of cost objective. And we had some constraints. The first constraint was about availability. So we said that the number of. Uh. The number of setups must be less than or equal to what's available. For all I in AI. So for every generator of type I. Throughout the time horizon, we set up summation of shit over T, and that should be less than what we have in our portfolio. Okay. And so that was about the generator availability. And then we spoke about the capacity constraints. So the amount that we generate should be less than or equal to the capacity at time t times the cumulative number of generators set up until time t. Right. So x I tore down less than or equal to t and that was the output capacity. And then we said that we needed to satisfy the demand. So it was a quite straightforward problem in the sense that everything was given. So you had a portfolio of generators, you deploy them, you generate power from them, and you satisfy the demand. And to satisfy the demand, the the amount of power generated from all generators. Should be equal to the demand at time t. So that was a demand satisfaction. Right. So that was our deterministic model. And we spoke about the various things that could go wrong with the deterministic model. What if the costs were different? What if the capacities were different? What if the demand were different. So when they are different and we showed some examples where we show that when they are different they actually make the solutions quite different. Right. And we also spoke about what needs to be proactive, what needs to be reactive. So the different types of decisions involved. So setup you can be proactive with setups. But we cannot be reactive with setups because setups take a long time to to complete. So you cannot wait for the amount of wind to happen. And then say now I start building a wind generator. This will happen like five years later. Okay, so there are decisions that need to be proactive and decisions that need to be reactive. And because of that, we built this concept of a two stage stochastic programming model. So that was when we started adding uncertainty. We created the two stage stochastic program okay. And this came into a first stage where we are saying. Um summation of I summation of d c I'd s exit, so that will be generated installation fixed plus. Then we react to the fixed setup. So we set up. First we take the fixed setup as an input and then we react. This is what we call recourse. So we take some recourse actions. In such a way that we, we minimise the average cost of the recourse action, so that recourse action takes in the first stage decision and the uncertainty in the planning. Right. And then we had the, the first stage constraints obviously remains the same x I t over t less than or equal to f I for all I. And then we had a series of second stage, uh, second stage constraints that belong to this model. Q right. So this is something to remember that this is not just a function, it is an entire model. And the model is the recourse model. It's what prescribes the reaction to what you did in the first stage. Okay. And so Q we said was something like this. We said where? Where Q is itself a model. That involves a minimisation of. The second stage cost. And subject to some second stage constraints, which is the output capacity constraint. She tore for Old Town less than equal to tea and summation of why it over t equals to TT. Right. So this was this was the simple two stage recourse model Y recourse. This is the recourse function. This is what will react to what you do in the first stage. And this reaction is constrained based on how much capacity you have available and what what is the demand that you need to satisfy. Right. So this was this was the the entire entirety of it. But then there are some things that I didn't go in that in that level of rigour with you, in the sense that if we look at this purely from a probabilistic point of view. What this is, is simply. A probabilistic concept, which is the expected value of a function over a distribution. So technically this should be this. Right. This. This should be the expected value over any distribution, picking from whether it's a continuous or a discrete distribution. But what we are doing, at least in this course, is to work from a scenario based perspective, which is that we have discrete outcomes in terms of scenarios, and every scenario is allocated a probability. Yeah. So. Is that expected value or expected probability. So it's the expected value. So it's the expected value of q given the probability p right. So if for instance you you want to search for the expected value of some function of some function of uncertainty, let's say you know that it's just this the function of uncertainty times the probability of uncertainty right. So over an infinite span can you just keep going. Yeah. So you're trying to you get the distribution. You're integrating over the distribution finding the area under the curve right. Try to find it here. Just going to keep going over along the. Whichever the range of the uncertainty is right. And so this technically should be should be the the real model and the real model. The problem with it is it contains an infinite number of outcomes. Right. And it's very difficult to solve. How do you solve the integral of a model, that of a function that is itself an optimisation model. Right. So, so this all this complicated stuff. So what we did effectively, just speaking purely from the, from the rigorous point of view is that we said. We said let's define instead of instead of, uh. EPP. Let's define a discrete approximation of this distribution that relies on n scenarios and discrete outcomes. And we will then compute. The expected value over this. This distribution of the cost function. So how did that change things? This made. A very simple. Summation. This made a very complicated integral into a simple summation, right? And this became the function that we are looking for. Times probability of s, right. So this this is where it starts. If we start from first principle we are looking at any distribution. It could be a it could be a uniform distribution a normal distribution whatever it is. And we say no. But the data we have is not continuous. It's not infinite. So we use that data to approximate that distribution. And that data comes in in terms of scenarios that are discrete realisations of uncertainty. Right. So this in effect is already an approximation of P. And then we went even further. We said okay. So now the problem is this helps me solve an important problem before which was the infinite ness of the realisation. Now I have finite realisations. But what about this? How do I estimate this probability? I have data, I don't have probability. Right. So what what I have is scenarios of data of historical historical data. Whatever happened in the past, I don't know how likely they are to happen in the future. I'm trying to plan for the future. Using data from the past, and estimations of probability from that perspective is almost impossible, right? And so then what we said is we need a further approximation to this. And we use what we call the law of large numbers. So in the law of large numbers, what we know is that a true mean value. Is actually the same as. The average over an infinite sample. Right. So we take some data x I we average. We average it. And if the sample is infinite, asymptotically this will tend towards the mean value. Right. So this is a concept of law of large numbers. And we said we can actually now translate this concept here. Because if we are able to translate the concept here, what happens is I do away with this concept of estimating a probability because everything is one over n, right. And so we build what we call the sample average approximation so that some sample average approximation. So the sample level approximation, what we call SAR the SAR model replaces this. This discrete distribution of the cost function. With. Approximating it with summation s element of SN one over n. Of the records for every realisation. Okay, so we try to approximate and then what we say is. Actually. As n tends towards infinity. This should tend towards. Exactly. Sorry. This. Sorry. Right. So as n tends towards infinity, I'm going to replicate the true original distribution that's based on the law of large numbers. And that's how we we came up with the concept of sample average approximation. From a conceptual point of view. It obviously relies on on all the other statistical tools. But from an implementation point of view this makes things way, way easier. There's nothing that's different from the previous model, except that the probability is any equal is equal, right? However, we cannot implement sample average approximation just however we want to do it, because it has to rely on this concept. Right. And the problem would be if I said okay I do sample average approximation. I generate three scenarios and I try to solve the model. What is the validity of this? It's not replicating at all what I wanted to replicate. The replication is only valid when you have very big samples, right? And so. We said that in order to analyse sample average approximation, we needed two different metrics. The first one was convergence. So we needed to understand the convergence of of the model. And the second one was some performance, some performance analysis. So in terms of convergence, what are we looking for? What we are looking for is to understand when the model performance is stabilised. So this is the objective function right. So what will happen is what you often see is something like this. It's unstable and then start stabilising. Right. So we want to know how big should our sample size be for the SA to be a valid approximation of the original model. Right. And it doesn't need to decrease by the way you can have. You can have trends that are like this as well. So it doesn't need to decrease. It's just that it needs to stabilise at some point. And once it stabilises you, you've now understood. At what point? Your sample average approximation is valid. So why not go just bigger and bigger sample continuously? You will have a tractability problem, right? It will be impossible to solve the model because the number of constraints is increasing. So in terms of sample average approximation, I just need to bring you back to something about the solution of the model. Right. We spoke about the two stage model, and the two stage model became the sample average approximation. So the sample average approximation has n. The two stage model as well has a deterministic equivalent. This equivalent model is what you actually code in Gams and solve. Right? And this is just the summation. Uh, I t c I t setup exit. Uh, plus. Um, for now, I'll just keep it with the probabilities. Element of s n of the cost of it of generation times the amount generated in each scenario. Right. So what I do is I index the second stage recourse action on scenarios to show its dependence on the scenarios. And then what I have here is just subject to first stage constraints. And second stage. Second stage. Constraints. Valid for all as in Sdn. Right. And this is where the tractability becomes an issue. So if I keep increasing n I keep increasing the sample size, I keep increasing the number of constraints. And so one is I'm not sure about feasibility. And secondly, I'm not sure about the solvability of the model. So as as the model becomes bigger and bigger, it becomes more difficult to identify points that are feasible. Right. Everything clear so far? Yeah. Point number two and their convergence. And so the performance analysis. Yeah. Yeah. So so that's that's an important part of of testing models. So the first the first part is convergence which is to test the approximation quality. Right. And the second part is how is your decision performing. So I need to know whether what I'm prescribing is good enough. And to do that I need to generate scenarios. And I need to compare in the same way that you did in machine learning. You train and you test, right? So we have two types of performance analysis. One is in sample. In-sample performance analysis. And the other is out-of-sample. Right. So in-sample analysis is analysis of the performance of your decisions on the data that you have. Out-of-sample analysis is the analysis of performance of decisions that you prescribe on data that you don't have. Right. So suppose any sample analysis will will flow like this. So generate. A sample as an. Solve the model. Fix the first stage decision. And then test. Best second Stage performance. So exactly what? So what we are trying to gain from our model is the first stage decision. Because the second stage will change depending on what the reality is, what the uncertainty realisation is. Right. So what we are trying to get is the first stage decision and how is it performing, how is that something that we are proactive about that we are fixing, performing on different scenarios of uncertainty, realisation. So in fact, suppose you solve the model you generate let's say x star. And you want to test the performance of Q. X doll. These. Over as belonging to your original sample. Right. So what? What kind of test can you do? You can compute the average value of this resource. You can compute the variance of the records. So for instance, if I wanted to compute the average value of the records it would be something like this one over n s s n q. Fixing the first stage to the optimal energy s right. So that would be the average value. The variance let's say would be something like this. One over n s s n. The actual value of the realisation minus the mean value square, right. So you can compute the variance. You can compute coefficient of variation whatever metric you want to compute. But what this is giving you is how is the performance of the first stage decision on the samples that you have? Okay. And now out-of-sample out-of-sample is a bit different. So in out-of-sample what we are doing is we are saying. Um, so I start from here. Fix first stage decision. Right. Generate. And out-of-sample set. So an out-of-sample set means that if S is an element of. This. It means that. Sorry. It means that. S is not an element of s n. Right. So this is what out-of-sample is. It means that whatever is in the out-of-sample set cannot belong to the original set on which we train the model. And this is where we come to the idea of we are trying to solve the model, to plan for things that we don't have data about, right? We have historical data. Things that happened in the past won't happen exactly as they happen in the past, in the future. So we accept this. We try to generate new scenarios and observe how is our performance on this. Right. And in terms of of developing the performance metrics, they are quite similar to, to this. So for instance, the out-of-sample will just be one over. Let's say the out-of-sample settles and prime, let's say one over n prime summation of s belonging to out-of-sample and prime q fixing the first stage again v s. Right. So this would be the mean value of out-of-sample and the same thing. You can compute the variance of out-of-sample performance etc. okay. So everything's clear right now. So these were what we covered last week, but we also covered several metrics that are very, very particular to stochastic programming. One of them is the expected value of perfect information. And the second one that I will add today and I, I uploaded the the lecture notes, the lecture notes for last week again adding this concept in in in the testing part you can just have a look at at this. Um in terms of this. So just recapping so we have our, our stochastic programming. Let's just give them names now because we need to have some bounds. So the stochastic programming model SP is just this uh minimisation c t s exit plus um some s of s n p s q x v s and subject to something. Right. So that was stochastic programming model. And then we formulated something that said, what if we already know we have an oracle that knows exactly the scenario that's going to happen, right. And the different scenarios that are going to happen. We created what we called a wait and see problem. Which is just basically this. Um. Summation. Um, over s. I t sit x s plus. Um. Sorry. Plus q x v s. Right. So this one we are able to we are able to change everything because we have a perfect oracle. So we can plan way beforehand and we know what's going to happen, etc.. And so the first thing we said was, let's try to find what is the relationship between SB and AWS. So for instance, is SP is the value of SP going to be greater than or equal to w or less than? What do you think? So just if you just look at these two models. S p will be greater, right? Why? Why is that? Okay. Yeah. Good point. So let me do this then. Yeah. Some greater still. Greater right. There is that part. There's the other side that if you think about it, w s is actually just a more constrained version of sp y. So I can convert SP into w s by doing this. And I move. I move this one. Outside here. S p s. Do this and I add a constraint where x I t s prime is equal to x I s. For all x prime s element of s n, right. Do you see that this is the exact same model as as SP, but using the concept of uh x being variable over scenarios. So what I said is I just replicate WS and then I say the setup needs to be the same across all scenarios. And this creates the productiveness. So SP is just ORS but more constrained right. And when it's more constrained the optimal value will always be higher. Okay because it's a minimisation module. So let's just go back to putting SP as how we should be. This. Um, when you say more constrained, do you mean like over France or. It's more it. It just has more constraints. So, for instance, um, if I take this model W s which, which contains some constraints already, and I add this constraint, I add, I add to it a constraint that says x I t s prime should be equal to x I t s for all s s prime. You mean actual constraints not yet constrained mathematically? No actual constraints. Yeah. The feasibility space reduces. Yeah. So if I add it, if I add this, I obtain this model. Right? Because this model has exit fixed across all scenarios. So you have the exactly exact same decision irrespective of scenarios. Here I have variability of scenarios, but we are saying it has to be the same over all scenarios. So it's the same as saying it's a fixed decision, right? And so this, this part, we can establish that SB is going to be greater than or equal to ORS. Okay. And so. Now let's look at another quite, quite simple model, which is a model that we will call EV, the expected value problem, which is just this. Um. AWS has more constraints, so I'm just going to have this confused with you. Yes, exactly. Um, no, no. So S3 has more constraints. So there is a way to convert W into SP by adding more constraints okay. Yeah, yeah. And so then let's define a simple a simple model that takes in. The expected value of data. So we call it the expected value model. And from this we can get we can get an additional relationship. So what? What do we think? What's the relationship between, uh, EV and AWS? EV is a deterministic model, right? But what I enter as data is that the average value of of all the data that I have across scenarios, right? It's it's not a two stage model. It's a deterministic model that enters a single data point. And this is this is the data point. So what do we think? Is EV going to be smaller or greater than AWS? Yeah. Yeah it should it should be smaller because we are planning just for one scenario. So if we are planning for many scenarios, we have to maintain feasibility over all the scenarios. Right. So obviously we have higher cost. There's no way for me to plan for just one out of those scenarios and incur higher cost. But because I'm now trying to protect the system against more things that can happen here. I'm protecting the system against a single possibility. Right. And another way to view it is to say that just one scenario, probability of this scenario is one. And you cut this model into EV, right. So from that perspective. We know that AWS is greater than or equal to EV. And now finally, if we look at if we look at this model, we say, let's generate the optimal solution from this. So let's call it x e star. And compute this model, which is. You will see those a lot in in the literature by the way. So this is the model that we want now to compute x I t e star. Plus, um q x I t. But X. Let me start and summation p s s v s, subject to some constraints. Okay. Yeah. Um, I think it's like expected value of an expected value problem. It's you are taking the, the expected value of, you are taking the solution of the expected value problem. And then you are computing the average value given the solution. Yeah. Twice is a process. Yeah. Finding one. You find one, and then you fix it, and then you solve the recourse. Yeah. In a way your you are doing a two step method. Yeah. But the main, the main idea of this is we can also find a bound. So for instance let's compare then v to SP. What do we think? Because Eve and SP are very very similar. The only thing that's changing is x right? I'm fixing x to something. Irrespective it's A22 step process. I took x from somewhere but I'm fixing X to something right? So what do we think? Will SP be greater than or equal to EV or. So just just think about this. One is we are minimising and we we can minimise the, the function with x. In the second one we are fixing x to something that may not be may not give you the minimal. It cannot be something that gives you a better solution. Right. Because it's a feasible it's a feasible solution of C and we are fixing it at a feasible solution. It's it cannot give you better a better solution than SP. Right. And if we go even further. You will see that. Um. What? Yeah. We don't we don't need to go further here. I think we've established this. Right. Isn't that a swimming based system? It is not a small number anyway. Well, let's suppose it is the most efficient. It will be equal, right? Yeah. It cannot be less because one is a minimisation problem. So it will never be less if it existed. If that less existed, it should have been the optimal of speed. Right. And so we have now these these bounds even greater than or equal to SP which is greater than or equal to wait and see is greater than or equal to EV. And we propose one metric during the class last week which was VPI. So the expected value of perfect information. So let me just write things here. Um Eve is greater than or equal to ESP is greater than or equal to w s. Which is greater than or equal to EV. We propose one metric in in the last class, which was the expected value of perfect information. And we call that EVP, right? Um. Expected value. Of perfect information. Right. And we said that EVP is just ESP minus w s, right. W s is the concept of having an oracle where you are predicting. So you have perfect information, right? What is the value of having perfect information is what value do you get when you don't have perfect information? What difference between this and this okay. And. Now I'm going to define a new concept called VSS, which is the value of stochastic solution. So this is important because when we are modelling stochastic programming, what we want to know is, is uncertainty valuable to be modelled? Right? If uncertainty wasn't necessary, then forget stochastic programming. Just solve the deterministic model with the average value as input and you don't lose much, right? And this is. Eve. Minus ESP. So what is Eve? Eve is the model that we solve where the input first stage decision is that of the deterministic model, the deterministic model with average data. Yeah. And so that one difference with the actual value of the model will give us how much does stochasticity add value to the model. Right. And if you look just look at these bounds. First of obviously I think I think this should be quite quite clear. All both bounds of greater than or equal to zero right. Because we are taking something bigger and smaller. Something minus something smaller. So both of them will be greater than or equal to zero. And the other interesting aspect of it is that. Both of them should be less than or equal to EV minus EV. Right again from the same thing to the furthest bounds possible. The difference between the first bounce possible should be the upper bound of the other differences. Okay. So meaning that if Eve equals to EV. Meaning that the model that you solve on average data is exactly the same model as you take the solution of the first stage from this average data and you solve the cost function, they are the same model. Then you don't have any value in adding perfect information and you don't have any value of stochasticity. Right? In if you if you think back to the first to the first lecture, what this means is if I sold my model on scenario two, which was my average scenario, I saw my model. Scenario two I get exactly the same optimal as my stochastic programming model. Yeah. Or equal to zero. Is that an optimal solution or is that one? Yes. The optimal solution then becomes very straightforward. It's just the solution of the model on average data. So you solve a deterministic problem on just scenario two which we had in that first lecture. Right. So if if these two are equal you actually have. Uh, you actually have a perfect model already. You don't need to model any stochasticity. And in general. Values and these bigger values. It depends on your interpretation. If you are comparing two models, one model has having a higher value on perfect information. It means that the model is further from the model with perfect information. So that means you. You will pay a lot to to have perfect information. And in the same way, if you have high value of stochastic solutions, it means that stochastic stochasticity plays a big role. Yeah. Any questions? I've added just this part to what this does, but I've added this part as well to to the lecture of last week because this is important. At some point you will read a lot of papers about stochastic programming. You will see this, you will see VPI, you will see all these like technologies. And this will kind of help you get an intuition of what they mean. Okay. Any questions? No. Um, so you can take, um. Yeah. Take take a five, eight minute break. Yeah. Because then it will be perfectly at three. Yeah. So take an eight minute break and we'll come back and we'll start talking a bit about multi-stage. Thank. Oh. Okay. So, um. Do you. Do you have any questions on on what we covered until now. Any any questions? No. Yeah, I. Understand the models. Like the ones you're presented. Yeah, I just don't understand. Do we use them in different scenarios or are they ones that complement the same problem when we approach it? Or what's the use of these. What's the use of this. So the main use of this is to evaluate two stage stochastic programming. So you have different evaluations right. One is you have an approximation of two stage stochastic programming. So you want to evaluate the approximation. The second one is you want to evaluate decisions right. So average value etc. etc.. And the third one is to evaluate stochastic programming. So is stochastic programming necessary for this model. And that is where this comes into play. So you build then different models that don't resemble stochastic programming. Or they resemble stochastic programming in some vague way because you fix something or you change something else and then you see what's the difference between stochastic programming and what these models produce. So in a sense, if if, for instance, I have like evolves as zero, then stochastic programming is not a useful tool for for this, it's almost something like what do you use in machine learning to validate cross-validation. Right. It's almost like cross validation. But here we are cross validating a model a structure of a model. So we have to come up with other structures to to do this cross validation. So we're essentially like looking at the same model from different angles to see. If it's yeah yeah. We look at the model, we say okay. So one of the particularities of the model is we fix the first stage. What if we don't fix it. So then we have the wait and see problem. The second is that it contains this probabilistic nature. So what if it doesn't. So we fix it to to an average value of data a deterministic. So we try to to ask what if for for different to produce different models. Yeah. Any other questions? But is everything okay so far? I mean, are you able to follow like the the flow of of the lecture and everything? Because now it's going to get a slightly. Yes. Slightly. Yes. So I mean multi-stage is as I mentioned to you, it's optional for you in the sense that, um, the first time I taught the course, I said everybody had to do multi-stage model and the projects were horrible. Right. So from that, from that point on, I said, if you do a two stage model, well, you'll get your B. And so if you're ambitious, you go for multi-stage. If you're not ambitious, you say, I'm happy with the B, I don't want to spend too much time on the project. Great. You do, but you have to do good. Two stage models, right? If you have mistakes in two stages, then everything, uh, a, B is up to 69, I think 60, 60 to 69. Yeah. But in order to have like a distinction, you need to, to give, to give me some a good multi-stage model. So obviously if you try to just do a multi-stage model and contains a lot of mistakes, it doesn't help. Okay. Yeah. Suddenly the problem is stochastic. And then at the end you do it and it says the value of 60. And is that a stochastic function? Does that count as a failure, or does that just be proved it? Um. Well, in general this can happen, but very, very rarely if it happens. One thing it shows is that your feasibility space is very restricted, so you don't have a lot of choices. So in the end you choose either A or B, and whatever the stochasticity you choose either A or B, right. So you can make some some conclusions about the structure of the model and why stochastic programming is not appropriate for this. But in general, if you have if you have a problem that only has a very restricted feasibility space, why have a problem at all? Right. So you don't really have a problem. You can just try try the different solutions. Right. Yeah. I was wondering is there any possibility of an extension because we have two assignments almost coming in two days. Yeah I think you can you can all apply for extensions, right? I mean, it cannot come from me because I've already fixed the deadline. So the teaching office cannot allow me to change the deadline. So. But from your side, if you need extensions, just I think extensions are granted quite easily. So you just ask for an extension or you apply for it. I'm not sure what the system is. And you get it. Yeah. No, not at all. No, no. Yeah. Um, yeah. Even special circumstances do not affect marks. Right. So I always grade on the merit of the work. And then afterwards, if there is like penalty for late submissions or whatever, this happens later. Yeah. And any other questions? No. Okay. So, um, so let's let's now talk about, um, multi-stage stochastic programming. I think that the the premise of it is something that many of you have already asked me before. So what what we are doing here is we have different time periods, but only to stage a two stage model. So a single point at which information is revealed. Right. So two stage stochastic programming is this is this model where you have this scenario three if you remember you have this scenario three. So you make decision X here before any realisation and then based on some realisations. Data attached to some probability. Etc. S3. You made some decisions. Why? Right. But why was a decision that was time stamped right? You had generation in time, one generation in time two. Generation time three. So there is a big assumption underlying the two stage model. The big assumption is that. All information is revealed at a single point. So in the second stage I have information about time one, up to time five or whatever. The planning horizon is. Right? And this is a huge assumption. So it will mean that let's say you are the decision maker. You solve this two stage stochastic programming model. The idea of it is you have the first stage decision. So you implement the first stage decision. You set up all your generators and everything. Now you wait. You say, okay, at the end of year one I'm going to start generating. I'm going to start generating to supply at the end of year one. I only have data for year one or during your one. I only have data for year one. I cannot use the two stage in this case. Right. Because what I need is I need data for every year and then I can start generating. Right. And so this is where stage and information revelation or things that much. So the two stage model has a single stage of information revolution. But stage and time are two very different things. And in the multi-stage structure, what we try to do, which we try to reconcile stage and time, write everything that is revealed. With time. We add it as conditional revelations. So I'll speak a bit more about how this this comes into play. So. The first thing is, what I intend to do was is to make a decision X here. Wait for information. Revelation at time one. And have, let's say scenarios S1, S2, S3 and build for one y1 s1 up until y1 s3. And then once I make these decisions. I wait for revelations of information at time two. And then make decisions for, for these revelations. Right. So what I want is this. This will give me time to. And I can continue like this. Right. I can just keep keep going on like this. So whatever I did, if you remember the two stage structure, the second stage decision was conditional on the first stage decision. Here I'm saying the second stage decision is just the the decision in the first time period. And this conditional on this, I'm going to produce the decision of the second time period and conditional on this, I'm going to produce the decision of the third time period. So in the end, the decision here will be conditional on everything you do in the past. Right. And this is this is well it's called dynamic programming in some in in some cases but most most commonly is called multi-stage stochastic programming. And this is the main idea of it. Okay. So what we will try to do now is to replicate this model. So imagine what we were doing here. We had a model nested inside another model. We had a model queue nested in the big model right? Model Q produce this decision y and model the model produce x. Here I have a big model producing x. Nested in the big model is a first model producing decision at time one generation nested in this model. Another model producing decision at time two and you keep nesting, nesting and nesting. Okay. So you have a you have a loop of models. And so we'll try we'll try to address this. We'll try to replicate this. But firstly we want to do the same thing as we did before. So let's try things that that don't work very well. Things that are naive but that are intuitive. And we'll show you why it's important for us to nest. Okay. So the first thing we do is we say, okay, um, why don't we solve. Different problems. Different two stage problems, right? No nesting, no connection. We solve two different two stage problems. And we have all the decisions for, for every t. Right. So something like this. I build. I build my my two stage tree for time one. Make decision x one. Makes decision y one. I build another two. Stage three. X2 y2 time two. Right. And I keep doing this until. I have a time. Cardinality of T, right. So if I do this, I solve the model. It's an easy solution because I'm solving t independent two stage models, and those are the independent two stage models I can formulate. As this. So suppose I call this. Z. Of approach one uh, that takes in the probability. Remember, the probability is still there because there are all two stage models and it will be a model for every time period. Right. And this will be something like this I will say minimisation. Um summation I of x I t. Plus, um, the probability. The expected value of the second stage that takes in x xt. And outputs and takes in the uncertainty realisation of T and subject to the constraints. Right. Um, I yeah, I think I better write the constraints because just just to give you a sense of what they mean. So what I will have is X it is less than f I. Why it is less than or equal to m v I t times x I t and y I t summation over I equals rt. Yeah, because I have just the decision at time t, right? I don't have cumulative decisions from the past. I'm solving independently t two stage models. Right. So if I solve this problem, the first thing you will notice is that this is the same as this model. I removed t here. I removed you here. I remove this t this t is. Yeah. So in this case, yeah, it's exactly the same model because t doesn't make any difference. T is just different decisions for t different independent models. Right. What is changing is the data. I just change the data and solve the two stage model again and I'll obtain a different solution. And so this this this model here is replicating what what we are doing above. And so now let's try to understand what what can go wrong with this model. So what do you think can go wrong with this model? Feasibility. Yes. Because one thing is here. I'm not dating from the past. Right. So my original, my original constraint is. Is this. Right. And my original constraint for output capacity is this. Now in this context, when I'm obtaining X and I'm obtaining Y. Will both of these be infeasible? Not. Not both would be feasible. Not feasible. This 1st May be infeasible, right? It won't be. So let's let's generate it then let's, let's suppose from here we are generating x I stall right and y I stall. And I know that she stole. So for every T, I'm generating some shit. So let's let's approach one. Write a one. For every T I'm generating an x, I stall. So I know that this is true. Do I have confirmation that this is true? It's the new instances of exploits. So that's why the summation. Might not be feasible, right? Because this could be Fi and it's done like the next time period. Whatever is generating is above. If I when I'm summing everything right. So if this is feasible it doesn't guarantee that this one will be feasible. But this is different. This constraint is different because if I was able to generate something that satisfies the demand with fewer. Generators. Then I should be able to satisfy that demand with with a bigger number of generators as well. Right. So feasibility of this implies feasibility of this. Right. So not all constraints suffer the same consequences. There are some constraints that will become infeasible. There are some constraints that will become feasible that will remain feasible but not necessarily optimal. But we don't care about optimality right now. Right. So this is the problem. The first problem with uh. Just. And I just start solving two stage models sequentially, one after the other. What will happen is, in the end, my solution may not be a solution that satisfies the constraints of my original problem. There are situations in which my solution will satisfy is that if I don't have any accumulation, right? So if my model doesn't contain anything where there is accumulation of number of generators from time one to time t, then it's fine, completely fine. The model is separable in T and you can solve t different two stage models. So there are situations in which I actually can solve many different two stage model. And I replicate a nested structure because the situations the model is already independent of T. Right. And so that's, that's one, one way of, of looking at it. So let's try. A different way. So the main thing we are, we are observing here is that we don't have connections between the stages. When everything is independent, we have a problem of feasibility to the original problem. So I say let's suppose that we want to create some connection. So again at time one I'm solving this and I'm solving for one s. And I take this because I said, I observed that there could be invisibility because I'm not taking care of what I set up in the past. Right? I take this, I input. Into time two. A time to I'm making decision x2 and I have decision x1. From solving the previous model. And I solve now for y2 s to obtain a solution. And I keep going on like this. Until. Until at time t I will solve for XT. And I will have information about X1 store up until x t minus one store. Right. And I will have a two stage three. Yep. Here is the real time data. It's just data you acquired. Y is something you've gotten through a model. And that's why when we say, like the previous example, x1, x2, you're splitting it. It's not like y. Well X is is not data is decisions. Right. So so you're making you're making decisions in different trees right. So you make the decision in the first tree and you obtain this X1. And then you send this x1 to here. And you take care of this x1 to prevent this in feasibility that we saw earlier. And, and then you solve again the model to obtain a new x2 a y2. So the data itself will be different data. So data here. Will be V1S1. V1, s2, v1 s3. Right. And the data here will be v2 s1 v2 s2 v2 s3. And the data here will be v. B, t, S1. Etc. is the outcome of that decision that is x. Yeah, exactly. So y is the outcome of the is the decision that's conditional on this data. So when you when the data happens then we make this decision. Why. Yeah. Okay. And so just just to just to recap here what what we did is we said we sold everything independently. So two stage independently. And I will add here the data as well just to to make things very clear. Um, this was V1S1V1S3. Um this was v two s1 v two S3 right. So yeah we we have different realisations so different trees. And we saw every tree independently. So we saw the trees independently. And we obtain different values of x different values of y. So different decisions at different time periods. And then the main observation is if we do that then. The availability of generators can be violated because I had free generators available, and it's not necessary that at the end when I add all the generator that I was I prescribed in all the different time periods, I will not exceed if I don't have this guarantee. Right? So there is a possibility that whatever solution I obtain here would be infeasible for the original problem, right? And so we said, okay. The main problem then is that I am not connecting the different stages. If I was connecting, if I said, okay, I already set up two here and now I only have five left to set up and I should choose from five, not choose from the original seven, right? So if I said this, then I would make sure that I satisfy the constraints. Right. And this is what what we are doing in the second part. In the second part, we said that it shouldn't be independent. We take the decision of the stage, we input in before we make the decision. In the second stage, we take care of what we did. In the first stage, we see how many generators do we have left, and then we take from what we have left, right. So we are trying to formulate this particular model. So Z let's say approach two again same probability distribution and t. So let's start with time. Let's talk with time period one. A time period one. This is just a minimisation of a summation of IXI1 plus expected value. Of q taking x one and v one. Right. And subject to constraints. And the constraints are XI1, XI1 is less than or equal to f I y i1 is less than or equal to VI1, VI one, XI1, and y. I won. Summation over I is equal to d one, right? In the first time period. It's exactly the same as the previous previous approach. Right. So this is basically the same as ZP1 approach one because now I don't have any conditions at the first time period. I'm solving for the first four. For the first three. And now I move on to the second tree. So the second tree then. Or three, let's say. Yeah. Let's look at the second one first in approach two is this we are minimising. Summation IXI2 plus an expected value. So from this one I'm generating XI1 star. Right? I'm generating the solution. Of the probability of something that is trying to get the best x2 given x1 stall. V one store. What happened in the post and V2 store? Right. Okay to start. The first one is going to come. Yes. Right. So not V2 store V2 uncertain. Right. So data that I have first stage realised realisation in the first period. Decision in the first period. Right. This is the data that I have. And therefore then I can say XI1 subject to constraints XI1 star plus XI2 is less than or equal to f I. Right? Because now I know how how many generators are set up in the first time period. I'm taking care of this. I'm not solving independently. And why I do is less than or equal to v I. Two uh, XI1 star. Plus Z two. Right and then summation YI2 over I is equal to d two. Everybody agrees. Yeah. This is the model that that takes in the decision of the first time period and the data from the first time period. And then we look at the model. So we look at this model. The first thing we notice is the data of the first time period doesn't matter at all. Right. So there's no connection in data in this model. In other models it can happen. But in this model V1 doesn't occur anywhere. So I can I can get rid of I can get rid of v1. Right. Yep. D2. Yes. It's good that you are all paying attention. Yes, d2. Yeah. That. So did you hear, VI? To hear y I to f I? And we are taking, uh, accumulation by accumulation on decisions that we've made in the first. In the first three. And this is the, the cost function. Um, so if you look, if you just look at the model V, one doesn't occur anywhere. Even though we have this information, we don't actually use the information. Yeah. So, for instance, if, um, when will we use. So let's suppose we are talking about, um, greenhouse emissions and we are trying to limit greenhouse emissions. Then we need the generation from time one as well. Plus the generation of time two. And we are trying to limit greenhouse emission. Then it will contain V1V1. Yeah. And so now we can generalise it. Generalise this. To t. So at time t, what this entails is x I. For time t. And X for time T here. Exactly. So all the extra x to one up until x dot t minus 1VT. VTi and then exit. Summation. Cold towel. Thou less than or equal to t minus one. Plus exit. Mhm. Yeah. All the previous ones. So t here t here. And here we'll have like this similar thing here. So we'll have it here v I t x I summation tau less than or equal to t minus one tau. And x it did and I t right. So now we can generalise this. So what is the model at time T is that model that is solving for the decision. First stage decision to be made at time t taking into account the optimal decisions that were made in time in previous time periods. And so with this there's no way we we don't satisfy this constraint for the original problem. Right. And there's no way in which we don't satisfy this constraint as well. This was already satisfied anyway. And the demand satisfaction remains the same. So. What was the problem then with this? With this model? So what's the problem with this model if everything is feasible? So everything that is feasible here is feasible for the original problem. Because now we are we are accumulating so great. What we wanted was accumulation. We are accumulating here as well. This will be feasible. This will be feasible. But it's still not a good model. Why is it not a good model? So then it's solving for solution. So what it's doing is it's optimising the first one and then using the first one to be as information for the next ones. Right. So what we are basically saying is look at the first time period, find the best thing that we can do at the at the first time period, even if it's horrible for the next time periods. Right. So what we are stuck here is in this idea of local optimality. So we are solving for now. And this can impact and damage us down the road. Right. So this is the main problem of this. It's not feasibility or anything. So we tried two different approaches. The first approach is one where we solve a. We solved the stages separately. We solved the time period separately. And this has problems of feasibility. And then we resolve feasibility by feeding information forward. And we have then problem of optimality. Right. And so this is now the main motivation for building a multi-stage stochastic program. Is that. Is that clear? Yeah. Not not not too difficult to understand. At least that part. Right. Um, so stochastic programs, what it will do is it will start with. A tree that includes conditions, right? So the main problem we have is in the way we are building the tree. So if you if you look at this here we are building independent trees. And here we are connecting the tree through a single decision. Now, the other thing I can criticise about this is that data is not connected. And that's that's a big that's a big problem. Because one thing is stages don't necessarily represent time periods, not always. So you've done I think you've done a bit of let's say humanitarian logistics in the, in the first, in the first course. Right. I think Douglass focussed a lot on humanitarian problems. So let's suppose I give you a data like this. So I have a disaster and you have people injured and things like that. And then there's a donation campaign that's going to happen. So after the disaster people start collecting donations and then bringing. Right. Can you see that the data is connected? The data is conditional. So if I have a big disaster, I have a big donation campaign and I collect a lot of donation. If I have a small disaster, I have a small donation campaign. It's pointless to have a small disaster and a huge donation campaign. And then a lot of things left over and and nothing used right. And the other way around is also pointless, is that you have a big disaster and nobody is sensitive to the disaster. It's like we don't donate, right. So data is conditional and these two are in stages. So let's suppose the donation is, I don't know, 20 to 30% of of the victim needs. So then depending on the victim needs you will have different donations. Right. So conditions in data is something very important. In a way information revelation is purely about conditions in data. I am adding like okay stages and time periods and all these things. None of these matters. If the time periods were not connected to each other, that's fine. We should solve independently, right? So stages are purely based on connections in data. And the way to build the multi-stage tree to represent this connection in data. Avoid local optimality and maintain visibility over everything. Is to build a decision tree like this. So we start here with making decisions. We start here with making decisions. In the first stage. Stage one. Exit. Okay. Across all tea. So we don't connect it with the time periods. We still want to fix. We still want to have a plan where we are able to deploy, and we don't have to react every time to change the generation installation plan. And then we have our data coming up. V1 s1 v1 s2 up until v1 s n. And then condition conditional on this. I have other data. Revelation. So decision I'm making here is decision for stage two. And this is why I do s. Right. So I'm making decisions at time to for generate the AI in scenarios. Whereas all those scenarios that happen here and then condition on this, I have further information revelation and I will make the next decision. So condition on this I am having let's say v one uh s one condition on S1V1S1 condition on s two. Generally we need to to choose a different notation for it. But just just to show you it's the first scenario that's conditioned on S1, second scenario that's condition on S1, etc.. And uh, I don't know how many scenarios, let's say s n prime condition on s two. And same thing here. We will have v one s1 condition on s n and V1S and prime condition on s n. And here we are making stage three decisions. Which are why I. I think it should be y1 y2 s, whereas is those conditional scenarios. Yeah. So this is this is the framework we are going to use to build multi-stage stochastic programming models. Okay. And uh. Yeah. I won't start covering the models now. I will cover it in the next in the next lecture. But it's just to give you an insight into where we are heading, uh, going into the next lecture and, uh. Yeah. So I will stick. I will stick here because I have another modelling challenge that we need to. We needed to do that. I forgot that we need to do. But anyway I will I will recap certain things from the from the lecture notes because I don't think I, I went into the lecture notes at all. Um, so yeah, just just recapping. Okay. Um, deterministic model. Everyone should be comfortable with this. The main issue is the lack of uncertainty consideration. We build functions to create the cost function that we call function as a model. So when a function depends on v, it's q brackets, v when it depends on first stage decision and v it's q x v right. And uh, with with this notation, we built the two stage stochastic programming model. And that as well should be quite, quite intuitive right now. So we have the first stage and the second stage. In the second stage everything happens for all time periods and we are able to build the second stage recalls all time periods. Okay. And we also saw that actually the deterministic model is actually a stochastic programming model where you have a single scenario that occurs with probability one. Right. So that's that's something that we can we can see as well. And uh, with that we build the scenario tree to make this discrete approximation of the distribution. Um, and uh, yeah, I say here that the probability should be one. But actually irrespective it can be it can be anything, but it's always equivalent to a model that has a probability one. Right. You just need to weigh the the Q in a different way. Um, then with this discrete distribution we are able to model that has n scenarios probability times q which gives us the expected value uh, across all scenarios. And using this we created the sample average approximation across n scenarios randomly generated, relying on the law of large numbers, where the limit as n tends towards infinity is asymptotically the the average value, the mean value. And uh, we see this condition here. And the sample average approximation is sort of this, uh, simple translation that changes the probability to one over n. But where we need to analyse the value of n the sample size. And this is the flowchart, the flowchart that you need to remember in terms of performance evaluation. And so. Firstly, you assume a distribution. So that's something very important. I think everyone has has gone on to this until now. Like stochastic programming relies on the probability distribution. This is the main assumption. I cannot do any stochastic program without an original assumption of a distribution. Right? So if I generate from a normal distribution, I will have a different convergence to. If I generate from a random uniform distribution to a Cauchy distribution, etc.. Right. So the distribution matters a lot, but you need to pick one. That's the that's the main drawback of stochastic program. But you need to pick a distribution, generate a large sample size sample average approximation and collect the first stage decision. Now at the first stage decisions, you either compute the recourse on scenarios that you have and then compute the statistics, or you generate additional scenarios that are out of the sample and you solve the recourse to your statistics. Right. So you have these two things. And this is something I want a bit more detail today about, which is the expected value of perfect information. And. Yeah. So so that's that's top tier for the recap of the first, uh, the first two lectures. And then we talked about decision timing. Right. So the idea that all the Y decisions are made here conditional on, conditional on scenarios that contain all data for all time periods. And we say that this is not what we want. What we want is something a bit different. So effectively what our two stage three is saying is this here no information revelation. Um uh, generate x and fix x here information revelation for the next five years. Uh, give the generation the generation plan no information revelation three in your four in your five no information revelation afterwards. Right. So this is what this is the the limitation of the two stage three okay. And then we said, well, we need to sort of expand this. We need to make decisions and allow information to be revealed with time. And this creates this sort of adaptive adaptability of decisions with time. And this is what will lead to the multi-stage model. And we tried a few naive approaches. Um, I think now maybe if you saw the slide in the beginning, it wouldn't have been too, too intuitive. I think now you can sort of understand from what I wrote, what these slides mean. So the first one, the first approach is we are solving t trees independently. So t two stage models. Right. Every two stage model is represented as this where you have decisions made at a single, uh a single time point. And uh, I talk about the problems with this model, which is in feasibility. So you may have in feasibility here in terms of the number of generators available. And we are not taking care of information that was available in the past. Right. So we just restart at zero every time. And then we say, let's address invisibility through this method where we do a generate a setup here. And we send send. And we keep sending the generator setups of previous time periods to the next time periods. And this gave us. That I showed how we came to to this model. I showed why this is the case as well. In the first time period, we are still just solving a two stage model. And the problem is we get stuck in this local optimality and we are still not connecting uncertainty. Uh, realisations. And, uh, this is, uh, I gave you the Gams code. I think all the Gams code are available. I'm not going through the game's code, by the way. So if you have issues with gams, just let me let me know so that I can go through them during the class. I'm assuming everybody knows gams. Everybody understands it. So if there are issues, just let me know, okay. And uh, so if you solve and you actually if you solve the models that I provided on the Gams code, you will see that actually you have these visibilities and you have these local optimality as well. And the main. The main problem with local optimality here is is actually more from the formulation point of view we are discounting. If you remember, I divide by one over all to the power of T, right? So it's better to set up later because then it's discounted back at the net present value it's better. But when I optimise the first time period first and then I use that to to optimise the next ones, I am not taking care of discounting at all. Okay, so this, this is where it will create some issues. And so the main thing that we are trying to do is this. So decide X, observe the first time period outcome and decide y one. And then observe the second time period outcome and decide y two, etc. until you observe the two time period outcome and decide it. So what we are trying to build here is this idea of a of a tree that that has conditions on information. Okay. And you will see here obviously there's a lot of notations involved because I need to have a notation for the first time periods scenarios. The second time period scenarios I need to have a notation for the conditions etc.. So there's a lot of notation. But the idea is that you have you can have now instead of probabilities conditional probabilities attached to it. So the probability of something will happen in time three is not just this absolute probability. It's what's the probability that something will happen in time three given what happened in time two right. So you have a series of conditional probabilities that connect this tree. And the advantage of this is that this is very, very general. If, for instance, your tree is not connected, there's no condition. All these will just be one scenario, right? It just it just becomes one scenario. One scenario. And if you keep one scenario, one scenario everywhere you come back to the to the two stage tree. Right. So this is a general representation of a multi-stage structure of information revelation okay. And I believe this is where I stopped. And then we'll go into formulating the multi stage stochastic programming models as well. Um yeah I don't think we'll have time to go through the modelling challenge, but that is something that I already provided you by the way. It's at the end of lecture two. You will see this where, where I ask that you uh that that you add demand uncertainty and then observe what happens to demand uncertainty. So add demand uncertainty. Build a two stage framework, build the deterministic equivalent, the sample average approximation, code it and solve it. And this should be quite quite nice to you. Yeah. Any questions. No. Okay. Um, yeah. So? So see you next week. By the way, in general, I finished ten minutes early. That's always my practice is to allow you to ask questions in the end, because I know not everyone is comfortable asking questions in front of the class. So if you have questions, just. I will be here. Just come and ask me, okay. I. That's okay, as long as I have an idea of what it looks like to travel. Oh. Where are you from? I'm from Mauritius. But.