Inverse Methods in Heat Transfer
Prof. Balaji Srinivasan
Department of Mechanical Engineering
Indian Institute of Technology - Madras

Lecture - 42
MHMCMC for Inverse Problems -- MATLAB Demo

(Video Starts: 00:19)

Welcome back, in this final video of week 7. I would like to show you a demonstration of the Metropolis Hastings Markov chain Monte Carlo. we just saw what the algorithm was in the previous video if not if you have not seen it, I request you to take a look at it without which most of this will not make sense. So, we are doing a parameter estimation but now it is no longer an offline Bayesian as we had done in the previous videos.

But it is actually a metropolis Hastings MHMCMC algorithm, let me just put this Metropolis Hastings Markov chain Monte Carlo. we are using the same problem as before. Now this problem I have just copied it from the previous uh video. So, it says generate 11 samples of the heat flux etcetera, but we are not going to do that we are going to try to use this MHMCMC Bayesian approach instead of the offline sampling given above.

So, what we will try to do is we will give some initial guess. So, here it is 900 to 1500. So, we will just give one initial guess somewhere within this range, let us say we give 900 and let us see how quickly it jumps in. Now remember typically we need a burning sum typically in need a burn-in period which we are ignoring now. So, maybe we will include it towards the end of this video just to see how that works.

So, now very simple initially we have the same parameters as before the same thermal conductivity length of the slab, temperature, the same measurements, the same uncertainty in the measurements nothing really has changed. Now all I am doing is giving some initial guesswork here. So, let us give this as 900 and we give an initial guess for q as 900. Now remember you now have a new proposal region.

The Proposal region means this if you gave qs 900 you want to now take the next point somewhere in the neighbourhood of q and you are going to do that with the Gaussian. Now as usual our T simulation is just this is our forward model, written that a little bit more clearly

forward model is this which we got from physics of the problem. Once you come to machine learning we will see how to make this forward model if we do not know it through physics and we can still combine it with something like MHMCMC.

So, that would be some interesting results that we look at a couple of weeks from one once we come to machine learning here is our MHMCMC algorithm same as what we wrote last time. You first initialize q which we have done to 900 and then you calculate a next sample I am always taking a sample around the previous value of q with the standard deviation of 90. I am going to keep this constant; you can vary this if you want but I am going to keep this constant just for convenience.

We calculate the new probability we calculate the new old probability if the ratio is greater than one that is a mu is more probable, we accept it if mu is less probable then we actually accept it with the ratio of P new by P old simple idea. So, the first step let me actually make take this into a clear step and call this step 1.1. This is just for your notes as you go through this hopefully this will make things a little bit clearly, we want to create some NS samples.

Now I have decided that I am creating only 11 samples, this might be too lower value because we are randomly sampling x here is where offline Bayesian might work well because we have already given some range and already somebody has decided a nice range but as you will see this will not go not do badly. So, you generate some NS samples the idea is this you start at the second sample and you generate the next sample in the Markov chain.

So, this next sample will be centered. So, this is a Gaussian with mean as the previous q and standard deviation as Sigma P, which was in our case this was 90 we have kept that as a standard. So, this Rand n that you see here simply means a normal random variable this MATLAB automatically generates for you any other programming language you use also will generate it for you it is there within standard libraries, there is a standard procedure to do.

So, which also looks remarkably like this a selection this is called an acceptance rejection ratio but I will not discuss that for now. For now, we let us say we have a method of generating this normal random variable. So, this is q new is now a random variable with centre at mu and a standard deviation of Sigma p, q old was the same as new in fact let me just change the order of these statements.

So, that it does not respond, if you wish we can call this q old plus Sigma well let me leave it as good just. So, that it is not confusing I will move this here just playing around with a little bit of the notation just. So, that when you read it, it is a little bit clear um later on after the class. Now here is my T simulation with the old q this is the error and sum of error square with old q.

So, remember we are actually comparing two different q's and we already have this accepted within the sample we are. Now deciding should we accept q new or not. So, for that we calculate this. Now some of you who are good programmers will know that the way I have programmed it is not efficient but nonetheless given the algorithm I have given above and given that we are writing very simple quotes, the lack of efficiency is not a big deal I am just writing it in a way. So, that it is clear when you read it later.

So, this of course is ax plus b, I will put brackets here just. So, that you guys can compare it with the formula that we do this bracket serves no purpose other than to separate out ax plus b. Same ax same a same b except of course a is dependent on q by K. So, this is q old by k this is q new by k same thing this is P Sim with the new q. Similarly, error with new q. all right so, P old is the numerator of the PDF q new and sorry q old.

And P new is the numerator of the PDF q new we are only comparing numerators. Now notice this is e to the power minus half sold by Sigma Square this is half s new by Sigma Square this is Sigma M corresponds to the measurement. So, this is the error of the measurement. So, it does not change regardless of whether I use a new q or the old q the measurement does not depend on heat flux or the error of the measurement the uncertainty in the measurement does not depend on the heat flux.

Now we had already decided that A is minimum of 1, P new by P old. Now if it turns out that P new is higher, we will accept it if it is lower than we will accept it with probability. So, now notice U is a uniform random variable this is just like throwing a toss with continuous values between 0 and 1 and in let us say A is 30 or 8 is 0.3. So, in 30 of the cases it will turn out that U is less than A, we will accept 30 percent of the value.

So, basically, we will accept these values with probability 0.3. Now once I run through this entire process, what will happen at the end of this is at this point all the samples are generated. So, what I will do is I will run the code till here I will run the code till here just to show you what happens. So, when we come here, we can now write what the samples are. So, you will see this samples let me write this again.

The first sample was 900 the second sample was 1029.8, 1175.2. Now you can see slowly it is moving its way up here. So, it has decided that these are the samples. Now let us see how well or how poorly the samples do I took very few samples, but let us still plot the probability density function here and see what it looks like. Now this process, this portion is exactly the same as the offline Bayesian.

So, remember for the offline Bayesian here is an offline Bayesian code of course this was with a and b but the offline Bayesian all we did was we calculated these sums P1 by Sigma P1, Sigma P2 by Sigma P1, Sigma P3 by Sigma P1 all that is what we calculated with the given samples. It is just that the samples were pre-generated and kept as uniform on the other hand these samples were generated on the fly, we started randomly with 900 and we generated these samples.

So, as you see the job of the Metropolis Hastings is until here. This is end of MHMCMC sampling procedure. After it, it is simple evaluation. So, let us call this step four, step three, evaluation of the integrals. So, our quantities of Interest what is the maximum a posteriority value etcetera is calculated at this point. Now how do we calculate it same as before we calculate P, we calculate P1 you calculate P2 you calculate P3.

If you have a prior you incorporate that also here and then you calculate the PPDF. So, if I continue, we can actually see what uh the probability density function looks like there seems to be some issue here let me just check what the problem is. So, if you run this program, you see this. So, you can notice this, this was our initial sample then a few samples here and you see a nice probability distribution forming.

You will see a nice probability distribution forming here and you can see a little bit of the computation is waste but at least after a few things it has some more zeroed in on the right region. Now what happens if we increase the number of samples or if I change the initial guess

to let us say 1200. So, I could change the initial guess to 1200 and I could run it and we can see what happens here.

You see it starts at 1200 and it goes all over the place we can now look at what this looks like when we write qs it started at 1200 went to 1236.9 and it just hovers around the same place. Notice that it is intelligent enough to just keep on hovering around the right place, it does not go off far away from where most of the probability is concentrated. In fact, I can also look at let us say what the value of qm is? it comes out to 1244.1 which is fairly close to what we had earlier.

We can also look at the standard deviation which is 11.7. So, what this tells you is that the original probability distribution of this is has a problem within the sample sizes around 11.7. I can also increase the number of samples. Let us say instead of 11 samples I put 501 samples and run it what you will notice is that you see how many points are clustered right around the maximum and it almost never takes points from outside this range.

I can try this also with let us say an initial yes of 900 and you will see that from 950 etcetera once it is zeroed in here it just stayed here and it put a lot of points. Of course, I have kept from 1 to NS, I can keep a burn in Period of let us say b is I had 500 points. So, let us say I can take around 10 or 20 this is our burn in period and instead of taking the average from the first point I can take from b plus 1.

So, I can go from here till there and I can check what it looks like. So, you will see oh of course I should have cleared everything till now and we can rerun the entire thing. So, that it comes only from the beginning. So, I made a mistake by plotting both of these together instead of doing that we can plot in just from the burning period and that will work just fine. So, let us remove this the point here is even if you start at a bad initial guess in fact let me show you, you could have a really bad initial guessing.

But even start on the higher end let us say we started at 1500 and we can see what it looks like. So, if we started with a bad initial guess 1500 it will work its way back and come to the right region again and you can again check what qm is? qm is the mean q it turns out to be 1242. You see it is remarkably stable and this is the power of this MHMCMC algorithm regardless of how bad your initial guess was, even if you started right at the edge.

Again, I will show you a very bad guess and we can see how well this works you can start with even 2000 which is well outside the range of where we guessed initially you will see it starts with 2000 but it will quickly come back to the same range and we can check what qm is again 1242.9. So, we always recover the same estimate for what a decent heat flux for this problem is. Of course, if we throw in a burn in period all these initial you know seven eight initial guesses would actually have disappeared.

And we would have been able to do the problem in a much more efficient fashion. So, this burn-in period is an important thing in practical cases, for our problem it is a simple one parameter problem. So, it does not matter there. So, long and short of the MHMCMC process is you start from some arbitrary initial condition, let us say 1200 and you can add as many samples as you want.

And typically, you will always converge to a decent value for the parameters. As I said it is very efficient when you come to a large number of parameters. So, it turns out that it is as sufficient for large parameters as it is for small parameters it is just that a pure Monte Carlo, a simple Monte Carlo sort of the offline Bayesian becomes lesser lesson efficient as the number of dimensions increases.

This does not care too much about the number of dimensions. So, typically in the stock market, in large parameter problems MHMCMC is used significantly. Now what we will start in the next week is everywhere here we were being we were using our forward model from physics of the problem. Now what happens if the physics is difficult or the physics is not known in such cases, we use machine learning.

So, we will spend about four weeks looking at machine learning and we will try to interface it with the methods that we have used so far with an inverse problem. So, I hope you enjoyed that portion I will see you next week, thank you.

(Video Ends: 19:08)