MATLAB Programming for Numerical Computation

Dr Niket Kaisare
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No. #04
Lecture No. # 4.4a
Linea Equations - Gauss Siedel – 2

Hello and welcome to MATLAB programming for numerical computations. So, what we have done, in all these lectures so far is, solved various problems using a MATLAB scripts. However, if you want to develop, a generic solution technique that you can then reuse for different problems, it actually makes sense to have a MATLAB function, in order to do that.

An example that we covered in module 3, was the trapz function, to calculate integral using trapezoidal rule. So, if we want to create, a function of the same sort, for solving linear equations, that is something that I am going to talk about in today's lecture. This is we can call this as a lecture 4. 4a okay.

(Video Starts: 00:47) So let us go on to MATLAB and open our previous file, gauss siedel okay. And now what we want to do is, rather than having this as a script file, we want to have this as a function file, what it means by having this as a function file is our a and b matrices are going to be supplied to us from an external calling entity. And we are going to solve this and return our solution using this okay.

So we will say function x = gaussSiedel (A, b) okay. And we will have to delete these. I will just cut these. I will go on over here, and I will just paste them over here. So, that we have a and b in our workspace okay, so now we will put the comments after the function. So, solving Ax=b using gauss siedel and that is very much, all that is needed, with respect to initializing, let us move this n at right at the top. And n, is going to be nothing but, length (A) okay.

Usually, when you write functions of this sort, you also need to do certain error checking. For example, we want to check that, A is an n/ n square matrix. We also would like to check whether B is an n dimensional vector. I will skip these steps for now okay. And so, we will just go ahead and do this.

So, let us, we will probably need to delete, the diagonally dominant elements. And we will just call this as, As and that should be good enough. With respect to initialization, the x is indeed initialized in this way, and max error is initialized in this way. We would also like to have, this 25 as another variable called max iteration equal to, let us say 100 okay.

So, 1: maxIterations and we will also need a tolerance tolX=1.0e-4 okay. So this is what we need. And so, what we have done so far? What we have done so far is this, the things that we had coded. For example we need to do this for 25 iterations. We have now kind of given, a variable which is called max iterations. When we do, we tell our solution methodology to stop, rather than you know, deciding that in an ad hoc way, we are trying to formulize that using the tolerance which we called as tolX okay.

So, and we do not want our code to keep displaying these results. So instead, what we need to do is, if (max (err)) < tolX, so what we mean by that is, the maximum of the error is less than tolX. Then we want to break, end, and what break does is, it breaks from the for loop, which means that it is going to come out of the iteration for loop. Before, we do that we would also want to give, isSolution=true okay and we will say isSolution = false here.

Okay so that is a flag that basically tracks whether or not the solution is converged. So, if the solution is converged, I am going to say that isSolution = true otherwise, it returns this solution as false, and this is something that we want to return as well. So x, isSolution okay.

So this is what, we have so far and yeah so we will save this, and now what we need to do is, basically, call this function using our usual function calls. So gaussSiedel (A,b) okay. And if we just give this gaussSiedel (A, b), the first output argument will be returned as the A and s that is the array, that is the vector that captures the solution okay.

And, if we use this gaussSiedel (A, b) in a naïve way, what we are going to get is, basically an incorrect solution over here. And the reason, we do get that is because this overall matrix, was not diagonally dominant okay. So let us consider, the more diagonally dominant case, where basically, we have replaced the first and second rows that that should be sufficient for us, for our purpose, so we can say a Anew (1,:) = A(2,:); Anew(2,:) is going to be equal to A(3,:) and Anew (3, :) is going to be equal to A(1,:) okay.

And same thing, with bnew, also so bnew (1) is going to be equal to b(2), bnew(2) is going to be=b (3) and bnew(3) is going to be = b(1) okay. Or actually I should say, b(1,1), b(2,1) and bnew (3,1) okay. And we press enter, and we see bnew is this guy, and Anew is again rearranged matrix okay. So, now we call gauss siedel, using Anew and bnew and we get the solution, using the gauss siedel method, the solution that we stop i have end, the tolerance of 10 to the power-4 is met.

Okay so, if we want to look at our flag as well, we can write this as [x, iflag] =gaussSiedel (A, b) and the flag should say that is not converged. So the flag is 0. Flag is 0 means that flag is false that means is converged. So is a false state and if I repeat that with Anew, bnew okay, I am going to get flag = true which means that the solution has converged, within the desired tolerance value okay. And our solution, is captured in the vector x okay.

So, what we have done over here is, took the solution of the previous problem, that we did in lecture 4.4 and converted it from MATLAB script into a MATLAB functions. A MATLAB function, which we now hope to use later on for various other computations, if desired okay.

So, this is how you would build your functions, and your files in order to build a large set of programs, as you become more and more experts in MATLAB. So with that, I come to the end of lecture 4.4a and next lecture, we are going to talk about, an exciting new method called Thomas algorithm or tridiagonal matrix algorithm thank you and see you in the next lecture. (Video Ends: 09:53)