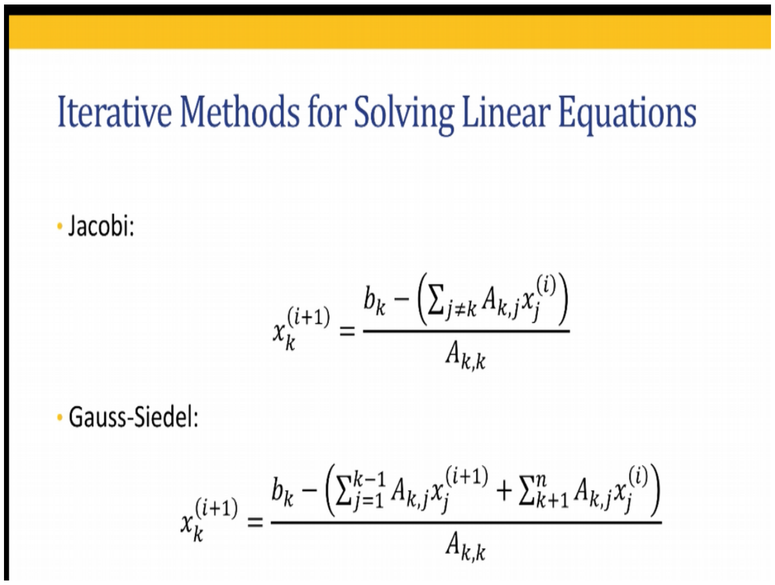
# MATLAB Programming for Numerical Computation

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

**Module No. #04**
**Lecture No. #4.4**
**Linea Equations - Gauss Siedel**

Hello and welcome to MATLAB programming for numerical computations. We are in module number 4 where we have been discussing solving linear equations of the type ax =b. This is lecture 4.4 and today, we are going to change the gears a little bit, and talk about what are known as iterative methods. The most popular iterative methods are Gauss Siedel method and the Jacobi Iterations.

(Refer Slide Time: 00:36)



## Iterative Methods for Solving Linear Equations

- Jacobi:

$$x_k^{(i+1)} = \frac{b_k - \left(\sum_{j\neq k} A_{k,j} x_j^{(i)}\right)}{A_{k,k}}$$

- Gauss-Siedel:

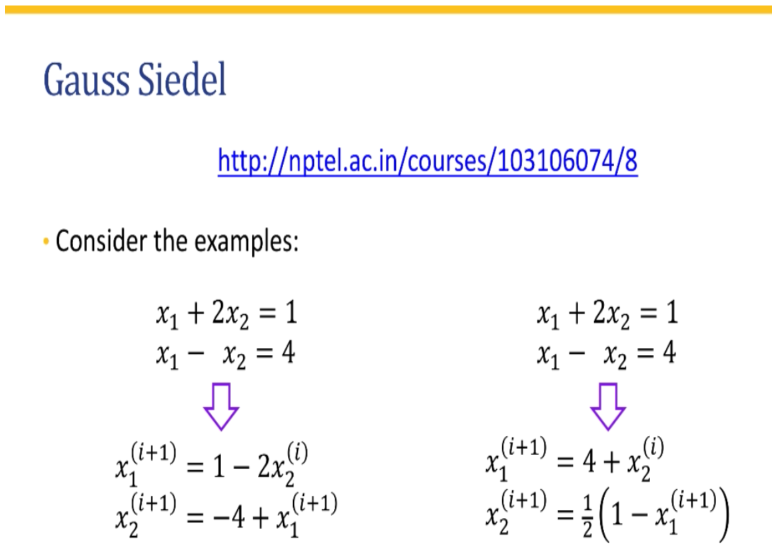
$$x_k^{(i+1)} = \frac{b_k - \left(\sum_{j=1}^{k-1} A_{k,j} x_j^{(i+1)} + \sum_{k+1}^{n} A_{k,j} x_j^{(i)}\right)}{A_{k,k}}$$

In Gauss Siedel and Jacobi iterations, what we do is, we write, rewrite each and every equation. So that the first equation can be used to obtain the solution, to the first variable x, the second equation is rewritten to obtain solution to second variable x(2), so on and so forth up to x(n). Which means for example, if we were to take the first equation, we will write $A_{1,1}$ x(1) = b1 – $A_{1,2}$ x(2) – $A_{1,3}$ x(3) and so on up to $A_{1,n}$ x(n) okay, then divide the whole thing by $A_{1,1}$.

What that will end up giving is, the first equation will then we used to find the solution for x1. Likewise we rewrite, the second equation, to take all the terms from the left-hand side except $A_{2,2}$ x(2). All other terms in the second equation, are taken to the right-hand side and we use the second equation to get x(2), so on and so forth. That is the strategy that is used both in Jacobi iterations as well as the Gauss Siedel iterations.

The difference between Jacobi iterations and gauss Siedel iterations, is that the, Jacobi iterations uses the solution vector x that was obtained in the previous iteration. Whereas, Gauss Siedel method uses, only the latest value of x.

(Refer Slide Time: 01:58)



## Gauss Siedel

http://nptel.ac.in/courses/103106074/8

- Consider the examples:

$$x_1 + 2x_2 = 1 \qquad x_1 + 2x_2 = 1$$
$$x_1 - x_2 = 4 \qquad x_1 - x_2 = 4$$

$$x_1^{(i+1)} = 1 - 2x_2^{(i)} \qquad x_1^{(i+1)} = 4 + x_2^{(i)}$$
$$x_2^{(i+1)} = -4 + x_1^{(i+1)} \qquad x_2^{(i+1)} = \frac{1}{2}\left(1 - x_1^{(i+1)}\right)$$

Gauss Siedel method was discussed in, computational techniques course module 3, lecture 6 okay. And the link for that is given over here. The example that we considered, is x1 + 2 x2 =1, x1 - x2 =4. The 2 ways to do this is, we call this as equation 1 and this as equation 2, to get these expressions, or we call this equation as equation number 1 and we call this equation as equation number 2 to get this expression.

What we do we mean is can rewrite this as x(1) =1 - 2x to as shown over here, and x(2) =4 - x(1) the whole thing divided by – 1, which is - 4+x1 okay. So that is written over here, so that is the first way of doing things. The second way of doing things is, we can write x (1) =4+x2 as the first equation. And x(2) =1- x(1) the whole thing divided by 2. So, we are going to look at, Gauss Siedel method using this as well as using this method.

(Video Starts: 03:08) So let us go to MATLAB okay. Now that we are at MATLAB, let us do the Gauss Siedel iterations. So let us first, we need to initialize our solution. So our solution, we will initialize as x = (0, 0) okay. And then the first iterations is going to be x (1) =1 – 2* x (2), 1 - 2 * x (2) and the second is x (2) is going to be = - 4+ x(1), - 4+ x (1) okay and we will just call this as iter1 okay.

Again, we are not writing the script. I am just showing you directly by typing it on the command prompt. You can do this for relatively small problems into 3 or 4 variables in a few minutes. I am going to show you how to write a function, in order to do this using the Gauss Siedel method okay. Anyway, this is the result after the first iterations. We get our x value as [1, - 3] okay.

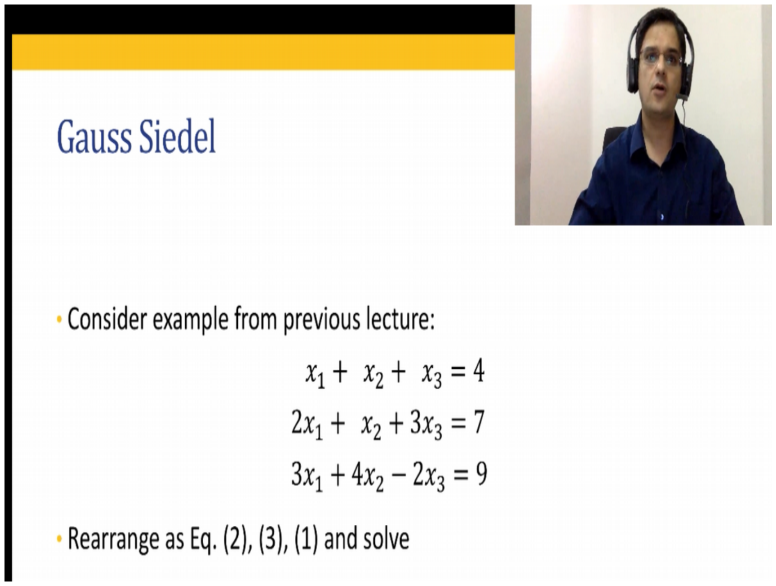
Let us repeat, this for iteration number 2. After iteration, after the second iteration x is [7, 3]. Let us do this for iteration number 3. x becomes [- 5, - 9] iteration 4, iteration 5, iteration 6, iteration 7, iteration number 8, iteration number 9 and iteration number 10 okay. At the end of 10th iteration the value of x becomes 1027 and 1023 as you can see the values are diverging. (Video Ends: 04:53)

In this case, when we use the first equation, to get compute x (1) and second equation, to get compute x (2) the value, the solution is diverging okay. (Video Starts: 05:08) Lets now see, what happens, if this is, what we were going to use, and now in this case, we will use a loop, again x = (0,0 ) okay. And let us use a loop for iter =1:10. So we will just repeat this for 10 iterations.

In this case, so x (1) is going to be equal to 4+x(2) and our x(2) is going to be equal to, let us see x (2) is going to be equal to half multiplied by 1 – x (1). 0.5 * 1 - x(1) okay, and end okay. So let us see what the solution for x is. Now x we have gotten as 2.998 and - 0.9990 okay. Let us go and do one more iteration again. And we will see, what the value of x is. The value of x is 3.0010 and - 1.005.

If we were to do, another iteration okay, x is going to be 2.9995. Let us do one more iteration and another one, okay just we will just stop, after one more, and this is the x solution, that we are getting the solution is, slowly and surely converging to (3, - 1). (Video Ends: 06:45)

(Refer Slide Time: 01:58)



## Gauss Siedel

- Consider example from previous lecture:

$$x_1 + x_2 + x_3 = 4$$
$$2x_1 + x_2 + 3x_3 = 7$$
$$3x_1 + 4x_2 - 2x_3 = 9$$

- Rearrange as Eq. (2), (3), (1) and solve

So, this is the just a very quick recap of, some of the theoretical underpinning, behind the gauss siedel and Jacobi iteration. So, that you can get a feel for, the fact that gauss siedel or Jacobi methods are not guaranteed to converge.

Okay, so now let us consider, the equations are the example from the previous lecture. We have these 3 equations and 3 unknowns. What we are going to do, in order to make the system diagonally dominant, is that the first equation, is going to be this guy, and the second equation is going to be this guy, and the third equation is going to be this guy. Our guy 4 over here is the largest value. So we want to put that, as this particular equation as equation number 2. Then this equation ended up becoming equation number 1. And this ended up becoming equation number 3.

So, if this is the, first equation, that is 2, 1, 3 is the first equation. So let us go here. (Video Starts: 07:50) clc edit gauss siedel. What we want to do, rearrange to get diagonally dominant matrix okay. So we want to rearrange as 2, 3, 1 right. So, As, we just called that as a new variable. New matrix is going to be =Ab(2) that the second row, followed by Ab the third row, followed by Ab the first row okay. So this is rearranged and we will delete this okay.

So now, we have set up the problem. Now what we want to do is, we want to solve this using gauss Seidel method. We will define, our error and our solution over here. Initializing x = zeros

(3, 1). Actually what I will do is n =3. There are 3 equations and 3 unknown's n =3 and x =zeros (n, 1) and I will have error = zeroes (n, 1) also okay. Keep in mind, I am using the word err and, not error because error is a keyword in MATLAB. So we want to avoid using known keywords in MATLAB and gauss g s iterations okay.

So, for iter =1 to, let us say we want to do 25 iterations okay. And we are going to use equation number 1 2 3 and so on up to n. So k = (1, n) and end the loop here as well okay. So, our x(k) is going to be equal to numerator divided by denominator. The denominator it is easier to put. So, the denominator is A(k, k). So, we will just put that first A(k, k) and let us see what the numerator is going to be. The numerator is going be Bk. Now the Bk is as kth row and last element right, again this has to be As (k, k) okay.

So Bs was nothing but sorry, the Bk is nothing but the last column in the As matrix, so the kth element last column is going to be As kth row and the last column. So, that is why we have As (k, -end) okay. (Video Ends: 11:04). What we have over here is, summation of A (k, j) * x (j) okay. (Video Starts: 11:12) A (k,j) is going to be As (k, j) * x(j) okay. And we are going to sum it over from j =(1: k – 1) right okay.

Now again, matrix multiplication actually works, in this particular case, because As(k, 1: k – 1) is going to give us a row vector, and x(j) (1: k – 1) is going to give us a column vector, (k, k+1: n) * x(k+1:n) okay. So, this is going to be the b part, this is going to be the first summation, and this is going to be the second summation over here okay. And x(k) is nothing but numerator divided by As( k, k)okay.

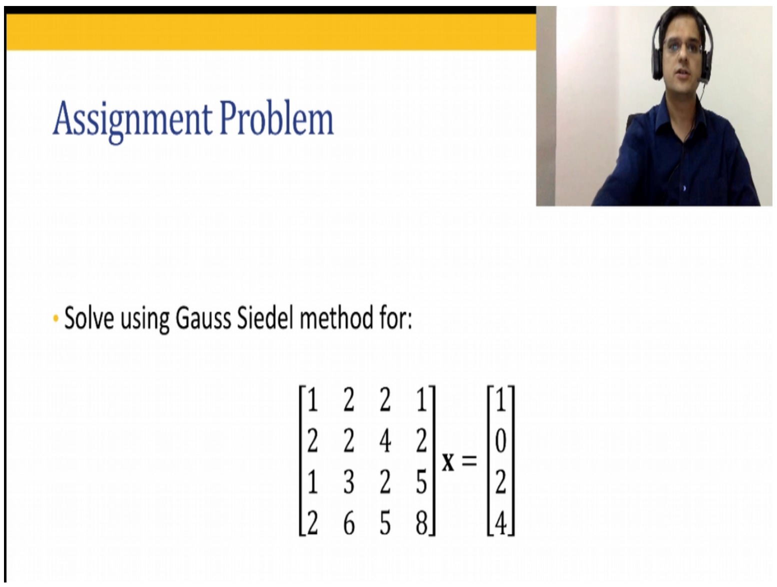
In order to calculate our error, what we want to do is, first store our x(k) in a dummy variable called xold. So xold = x(k) and we can have the command over here. err(k) is nothing but abs(xold) or rather abs(k) – xold okay. And we have done this, and now at end of every iterations let say we want to print few things. So, we want to print, the iteration number, iter num 2, str iter and error =num2str (max (err)) okay. And let us, save this, and run this and see what we are going to get.

So, gauss Siedel, oops! So there is an error. Let us go over here. So, there is an error, in this display, unbalanced parenthesis, which basically means that, I did not have, the brackets over here, so that was the error. So, again in the case of debugging, what we realized was, there was error, and the error said that, there were unbalanced parenthesis, with basically meant that we have to check our brackets. So, let us just clear this screen and run the gauss Siedel again okay.

So, what happens is, an iteration number 1. The error is 3.5 and that error drops quite significantly up to 7e -5 and we can say that, whenever the error has dropped below 10 to the power of minus 4. We should, we will say, that we are happy with the convergence. So, we will stop this at iteration number 23 itself okay. And if we type x, the solution that we are going to get is 1.0001, 2.0000 and 1.0000. If you recall from the previous lecture, this was indeed the solution. (Video Ends: 14:53)

So, this is what we have done. What we have done so far is, look at the Gauss Siedel method and solved a 3-dimensional problem, using the Gauss Siedel method. We will have 2 assignments for you. The first assignment is, you take this code for Gauss Siedel method, that will be posted online, and you modify this code to run this using the Jacobi iterations. That is your first assignment.

(Refer Slide Time: 15:23)



## Assignment Problem

- Solve using Gauss Siedel method for:

$$\begin{bmatrix} 1 & 2 & 2 & 1 \\ 2 & 2 & 4 & 2 \\ 1 & 3 & 2 & 5 \\ 2 & 6 & 5 & 8 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ 4 \end{bmatrix}$$

The second assignment problem is going to be, that you solve using the gauss Siedel method instead of a 3/ 3, problem you solve 4 /4 problem and get the solution x. Keep in mind this is the same problem that was given as home assignment, in the previous lecture okay.

So, what I suggest to you, if possible is after the end of this particular video, you go back and try to resolve your Gauss Siedel method in MATLAB without looking at this video. And once you are able to complete the solution, you can then go back look at this video and compare your solution, with the solution that I have given.

In case you are stuck, then you can go back and replay this video step by step and get yourself help to get unstuck from this, from this, from this problem okay. So, this is how I would suggest, you use this video, in order to understand, the Gauss Siedel method better. With that I come to the end of lecture 4.4. So thank you and see you in the next lecture.