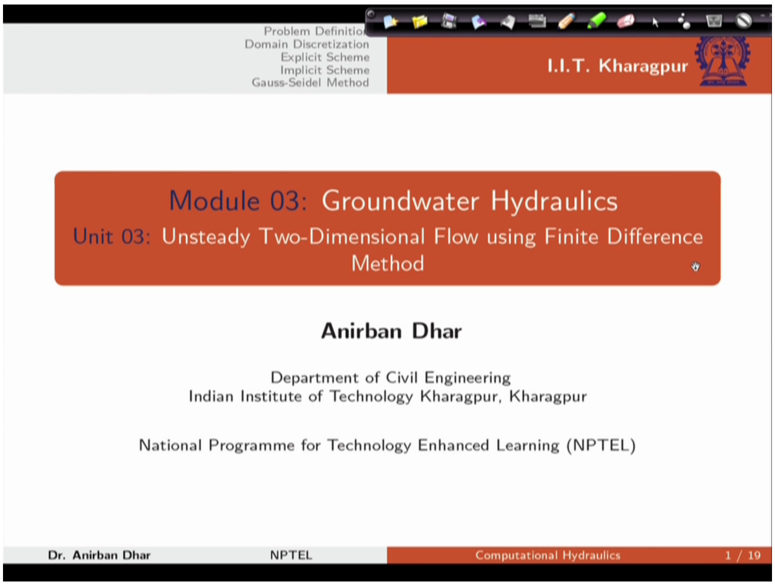
Computational Hydraulics
Professor Anirban Dhar
Department of Civil Engineering
Indian Institute of Technology Kharagpur
Lecture 33
Unsteady Two-Dimensional Flow using Finite Difference Method

Welcome to this lecture number 33 of the course computational hydraulics. We are in module 3, groundwater hydraulics. And in this particular unit I will be covering unsteady two dimensional flow using finite difference method.

(Refer Slide Time: 00:26)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Module 03: Groundwater Hydraulics

Unit 03: Unsteady Two-Dimensional Flow using Finite Difference Method

Anirban Dhar

Department of Civil Engineering
Indian Institute of Technology Kharagpur, Kharagpur

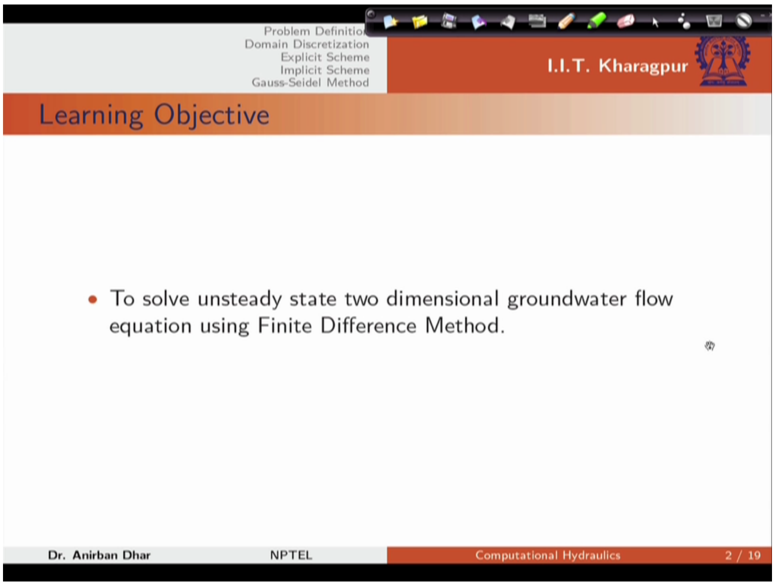
National Programme for Technology Enhanced Learning (NPTEL)

Dr. Anirban Dhar NPTEL Computational Hydraulics 1 / 19

In our previous two lecture classes also we have covered this finite difference approach. But in lecture number 31 I have covered one dimensional flow, steady flow. And lecture number 32 also I have covered steady two dimensional groundwater flow in confined aquifer system with homogeneous and isotropic condition. In this module also I will be covering the two dimensional groundwater flow with homogeneous isotropic condition. But it will be unsteady in nature.

So learning objective, at the end of this unit students will be able to solve unsteady state two dimensional groundwater flow equation using finite difference method.

(Refer Slide Time: 01:33)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

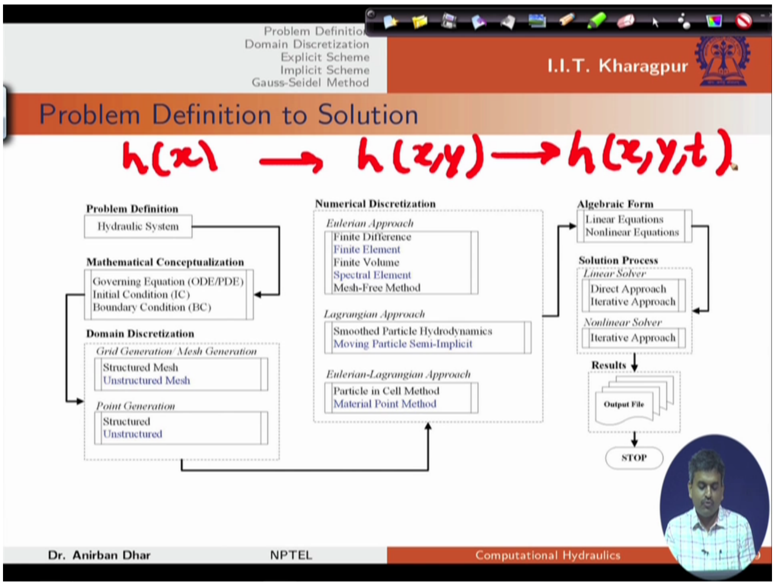
Learning Objective

- To solve unsteady state two dimensional groundwater flow equation using Finite Difference Method.

Dr. Anirban Dhar | NPTEL | Computational Hydraulics | 2 / 19

Again let us see our structure. This is problem definition. Again in problem definition for our lecture 1 to lecture 2 we have considered xy. And now in this lecture we will be considering it is a function of time. So everything depends on the conceptualization.

(Refer Slide Time: 02:05)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Problem Definition to Solution

$h(x) \longrightarrow h(x,y) \longrightarrow h(x,y,t)$

Problem Definition
Hydraulic System

Mathematical Conceptualization
Governing Equation (ODE/PDE)
Initial Condition (IC)
Boundary Condition (BC)

Domain Discretization
Grid Generation/ Mesh Generation
Structured Mesh
Unstructured Mesh
Point Generation
Structured
Unstructured

Numerical Discretization
Eulerian Approach
Finite Difference
Finite Element
Finite Volume
Spectral Element
Mesh-Free Method
Lagrangian Approach
Smoothed Particle Hydrodynamics
Moving Particle Semi-Implicit
Eulerian-Lagrangian Approach
Particle in Cell Method
Material Point Method

Algebraic Form
Linear Equations
Nonlinear Equations

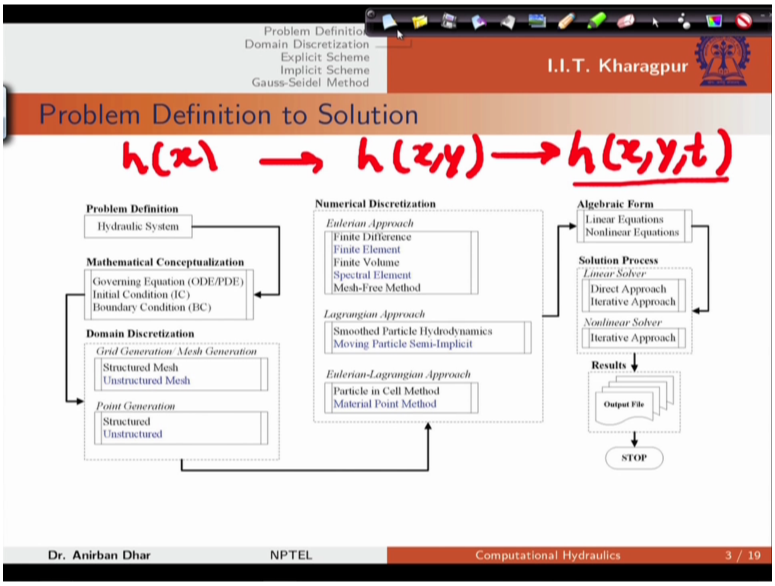
Solution Process
Linear Solver
Direct Approach
Iterative Approach
Nonlinear Solver
Iterative Approach

Results
Output File
STOP

Dr. Anirban Dhar | NPTEL | Computational Hydraulics

Whether you are considering it as a steady one dimensional, steady two dimensional or unsteady two dimensional or unsteady three dimensional system. But in this case I will show only two dimensional case. That will be much easier to understand. So in this case let us see what will be our approach for this solution.

(Refer Slide Time: 02:33)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

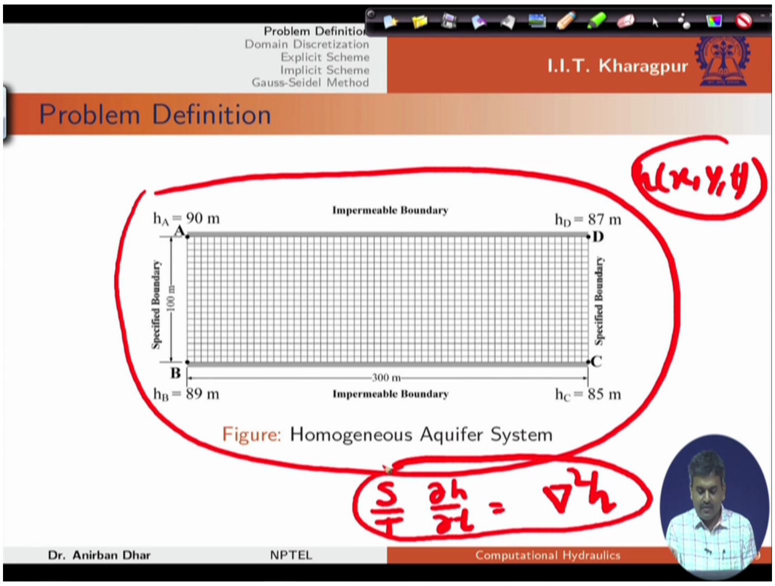
Problem Definition to Solution

$h(x) \longrightarrow h(x,y) \longrightarrow h(x,y,t)$

Dr. Anirban Dhar NPTEL Computational Hydraulics 3 / 19

Problem definition, like our steady state problem this is the case where I want to utilise the time dependent formulation for our steady state problem. So if we say that the equation is S by T, del h by del t and this is del 2 h, this is essentially 2D or homogeneous isotropic confined aquifer flow system if we consider h as a function of x, y and t only. But again we can utilise this framework for solving our steady state problem.

(Refer Slide Time: 03:40)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Problem Definition

$h(x,y,t)$

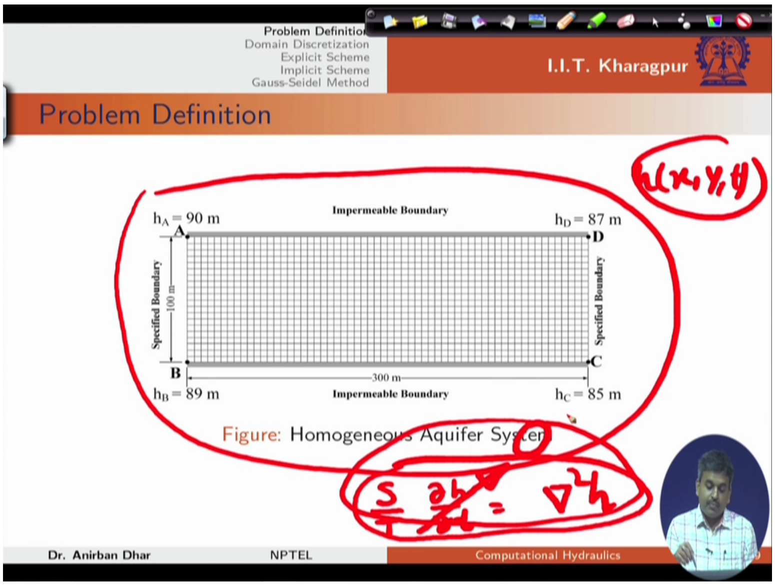
Figure: Homogeneous Aquifer System

$$\frac{S}{T}\frac{\partial h}{\partial t} = \nabla^2 h$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

(Fo) We can start with any arbitrary value or arbitrary initial value and we can go up to certain time and check whether the variation is still there. Obviously for a steady state problem if you utilise unsteady state framework, you should get the same result. Essentially what will happen? This del h by del t term, this will become very close to zero and we can solve this problem.

(Refer Slide Time: 04:22)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

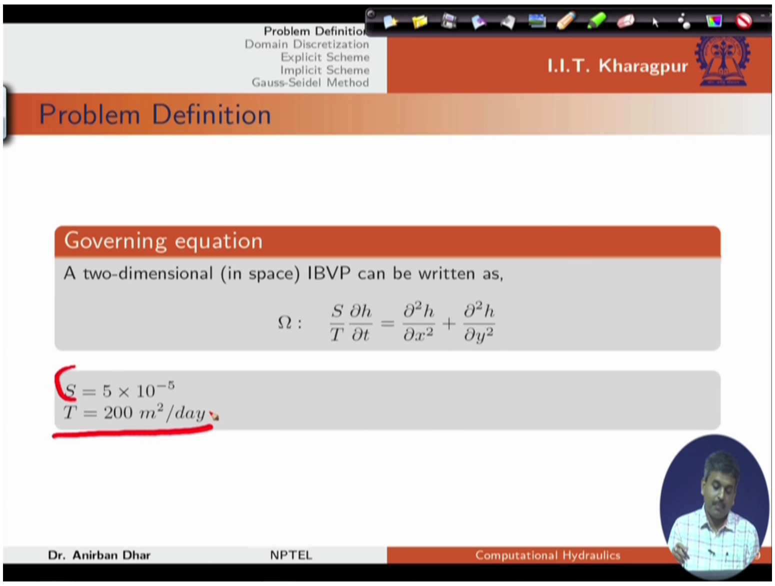
Problem Definition

Figure: Homogeneous Aquifer System

Dr. Anirban Dhar NPTEL Computational Hydraulics

So to solve this problem we need certain parameters. So obviously these are the impermeable boundaries, specified boundaries and specified point head values. Now for our problem this is two dimensional unsteady problem and in this case S value is 5 into 10 to the power minus 5. T is let us say 200 metres square per day which is transitivity and SS storativity of the system.

(Refer Slide Time: 04:59)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Problem Definition

Governing equation

A two-dimensional (in space) IBVP can be written as,

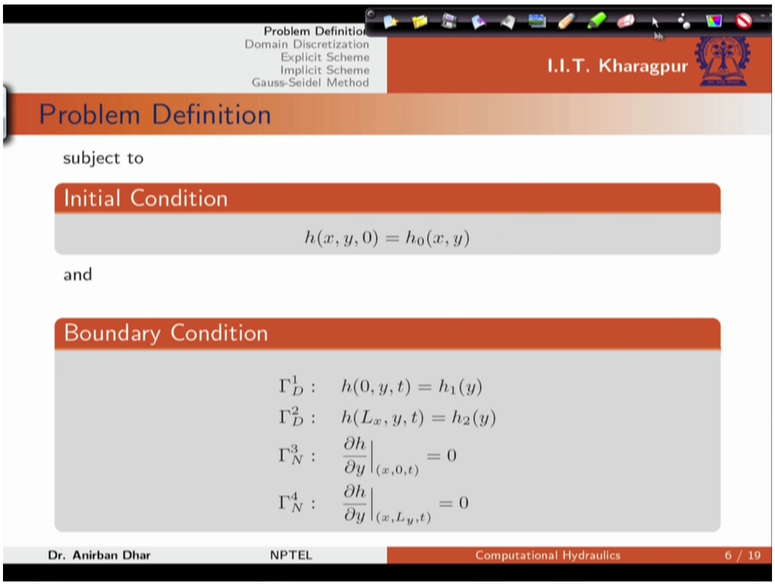
$$\Omega : \frac{S}{T}\frac{\partial h}{\partial t} = \frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2}$$

$S = 5 \times 10^{-5}$
$T = 200\ m^2/day$

Dr. Anirban Dhar NPTEL Computational Hydraulics

These are actually initial boundary condition. Initially we need to specify the variation of h within the system. This is not the function of t but this is at zero time level. We need to specify the value.

(Refer Slide Time: 05:24)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Problem Definition

subject to

Initial Condition

$$h(x,y,0) = h_0(x,y)$$

and

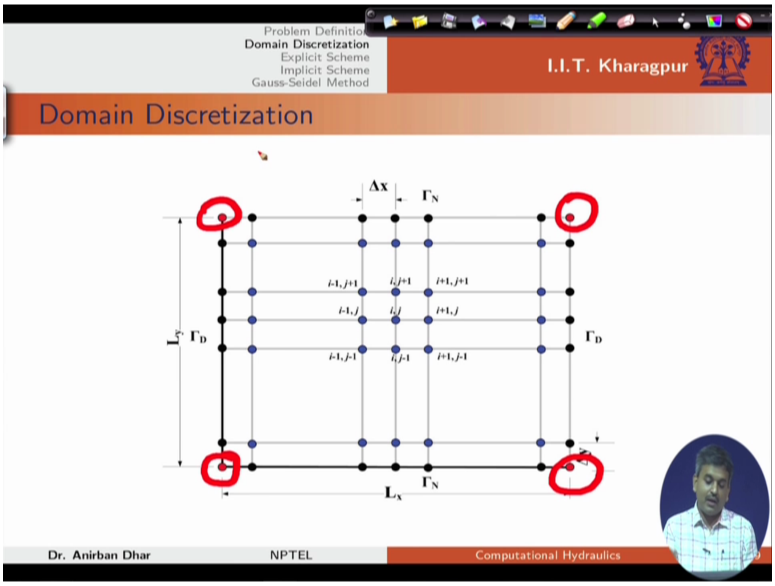
Boundary Condition

$$\begin{aligned}\Gamma_D^1 &: \quad h(0,y,t) = h_1(y)\\ \Gamma_D^2 &: \quad h(L_x,y,t) = h_2(y)\\ \Gamma_N^3 &: \quad \left.\frac{\partial h}{\partial y}\right|_{(x,0,t)} = 0\\ \Gamma_N^4 &: \quad \left.\frac{\partial h}{\partial y}\right|_{(x,L_y,t)} = 0\end{aligned}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 6 / 19

This is the domain discretization. You already know from your previous lecture class is that if this is the discretization approach with del x and del y, so we can utilise our finite difference framework. Obviously end points are important. End points either you consider it specified boundary or a Neumann boundary. Depending on that we need to specify the values.

(Refer Slide Time: 06:02)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

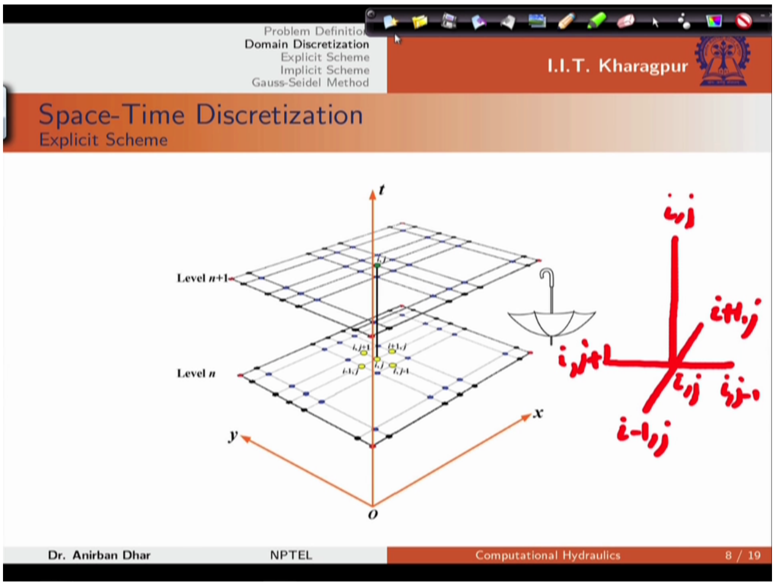
I.I.T. Kharagpur

Domain Discretization

Dr. Anirban Dhar NPTEL Computational Hydraulics

Explicit time scheme, you already know from your previous lectures that in explicit discretization we consider only one or central point in time from future and for other points we consider this i minus 1 j, ij minus1, ij plus 1, i plus 1j points there to solve this explicit problem.

(Refer Slide Time: 06:51)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

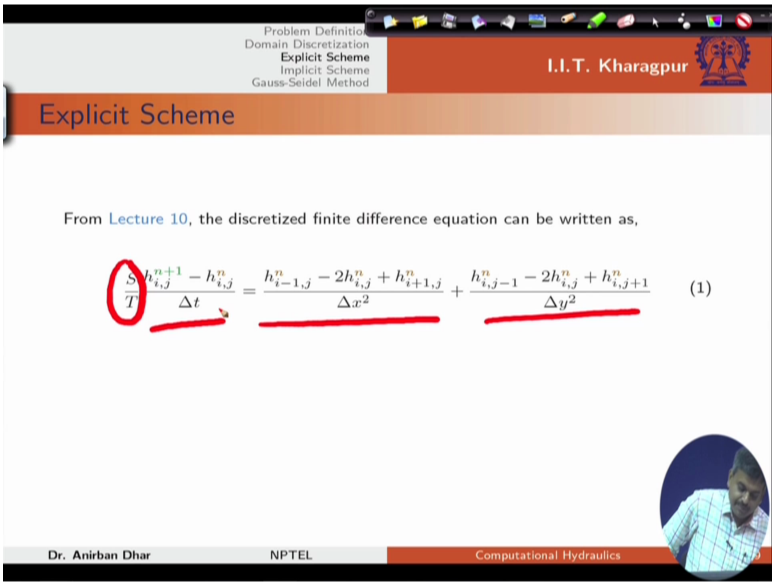
Space-Time Discretization

Explicit Scheme

Dr. Anirban Dhar NPTEL Computational Hydraulics 8 / 19

So our from lecture number 10 we can get this discretized finite difference equation. So in this case S by T is constant multiplied here. And in explicit approach obviously your space derivatives are evaluated at the present time level which is the nth level. And only time derivative that includes the future and present time levels.

(Refer Slide Time: 07:28)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Explicit Scheme

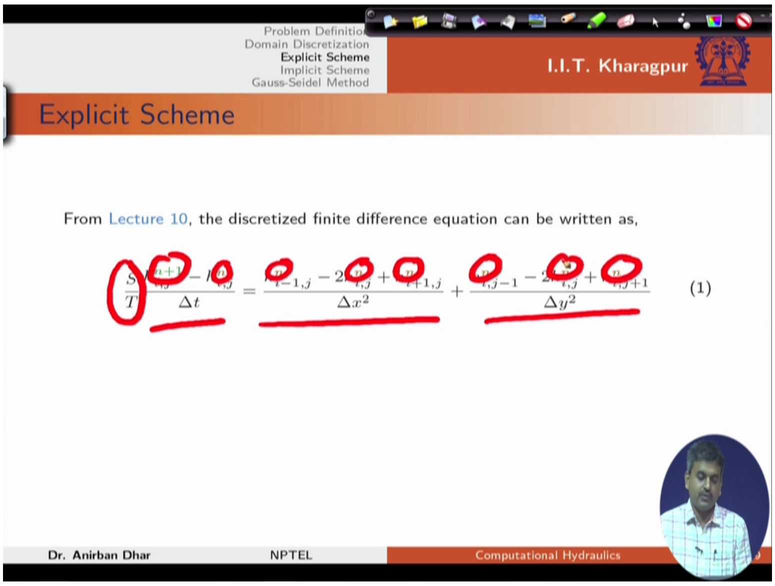
From Lecture 10, the discretized finite difference equation can be written as,

$$\frac{S}{T}\frac{h_{i,j}^{n+1}-h_{i,j}^{n}}{\Delta t}=\frac{h_{i-1,j}^{n}-2h_{i,j}^{n}+h_{i+1,j}^{n}}{\Delta x^2}+\frac{h_{i,j-1}^{n}-2h_{i,j}^{n}+h_{i,j+1}^{n}}{\Delta y^2} \quad (1)$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So this is your future time level, this is present, present, present, present. So all values we need to use the present time level.

(Refer Slide Time: 07:38)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Explicit Scheme

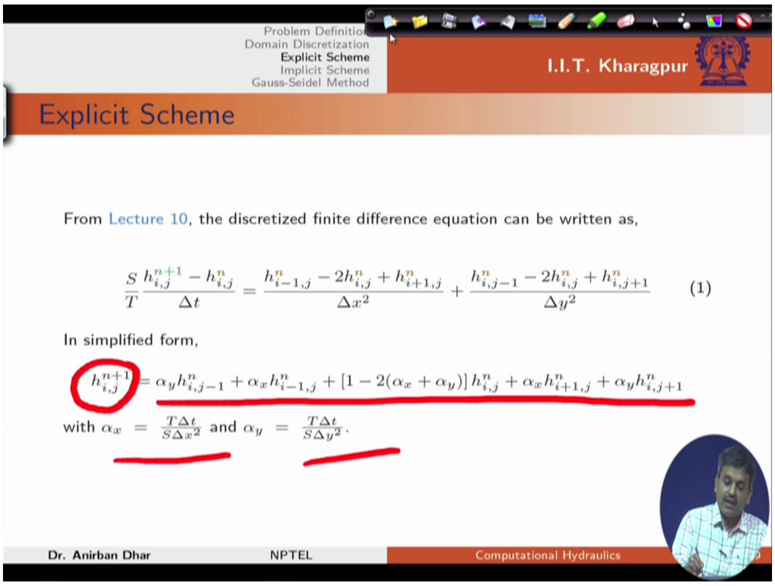
From Lecture 10, the discretized finite difference equation can be written as,

$$\frac{S}{T}\frac{h_{i,j}^{n+1}-h_{i,j}^{n}}{\Delta t}=\frac{h_{i-1,j}^{n}-2h_{i,j}^{n}+h_{i+1,j}^{n}}{\Delta x^2}+\frac{h_{i,j-1}^{n}-2h_{i,j}^{n}+h_{i,j+1}^{n}}{\Delta y^2} \quad (1)$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

If we simplify this one, what we can do? We can write it by transferring all known values on the right hand side. So left hand side, this is unknown at the future time level. Other values h ij minus 1, i minus 1j, ij, i plus 1j, ij plus 1, these values are known values. So obviously in this case we need to define this alpha x and alpha y. In our previous case that was 1 by del x square and 1 by del y square. But we need to multiply T and S here. So T delta t by S delta x square and this is T delta t by S delta y square.

(Refer Slide Time: 08:33)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Explicit Scheme

From Lecture 10, the discretized finite difference equation can be written as,

$$\frac{S}{T}\frac{h_{i,j}^{n+1}-h_{i,j}^{n}}{\Delta t}=\frac{h_{i-1,j}^{n}-2h_{i,j}^{n}+h_{i+1,j}^{n}}{\Delta x^2}+\frac{h_{i,j-1}^{n}-2h_{i,j}^{n}+h_{i,j+1}^{n}}{\Delta y^2} \quad (1)$$

In simplified form,

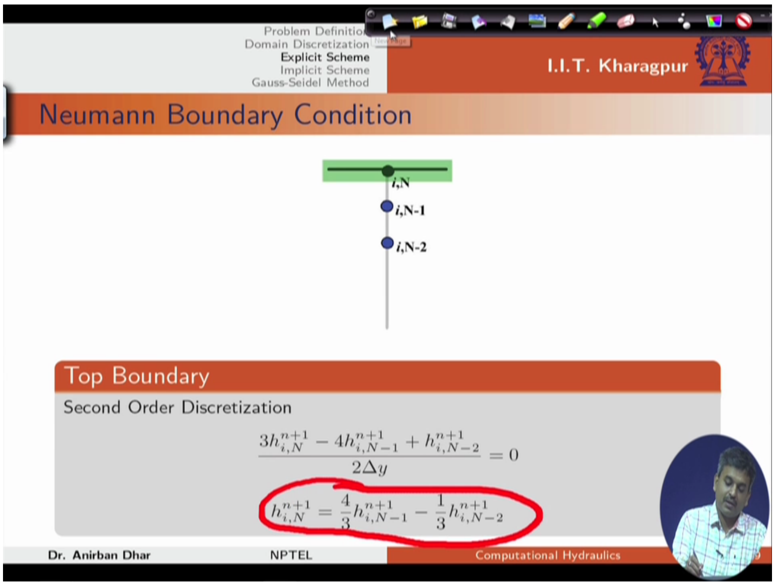
$$h_{i,j}^{n+1}=\alpha_y h_{i,j-1}^{n}+\alpha_x h_{i-1,j}^{n}+\left[1-2(\alpha_x+\alpha_y)\right]h_{i,j}^{n}+\alpha_x h_{i+1,j}^{n}+\alpha_y h_{i,j+1}^{n}$$

with $\alpha_x = \frac{T\Delta t}{S\Delta x^2}$ and $\alpha_y = \frac{T\Delta t}{S\Delta y^2}$.

Dr. Anirban Dhar NPTEL Computational Hydraulics

If we see our Neumann boundary condition, Neumann boundary condition we need to write it like this. Why? Because all values we need to evaluate at the future time level.

(Refer Slide Time: 08:56)



Neumann Boundary Condition

Top Boundary

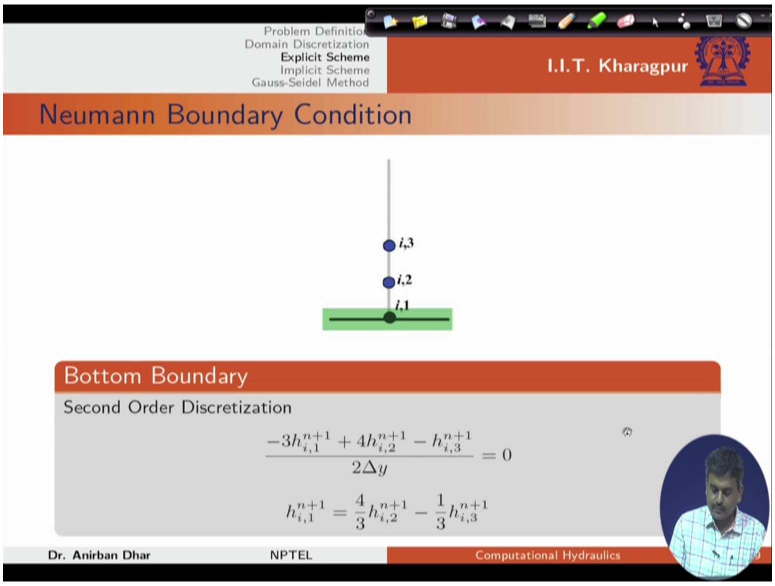
Second Order Discretization

$$\frac{3h_{i,N}^{n+1} - 4h_{i,N-1}^{n+1} + h_{i,N-2}^{n+1}}{2\Delta y} = 0$$

$$h_{i,N}^{n+1} = \frac{4}{3}h_{i,N-1}^{n+1} - \frac{1}{3}h_{i,N-2}^{n+1}$$

If we see our bottom boundary, this is same.

(Refer Slide Time: 09:05)



Neumann Boundary Condition

Bottom Boundary

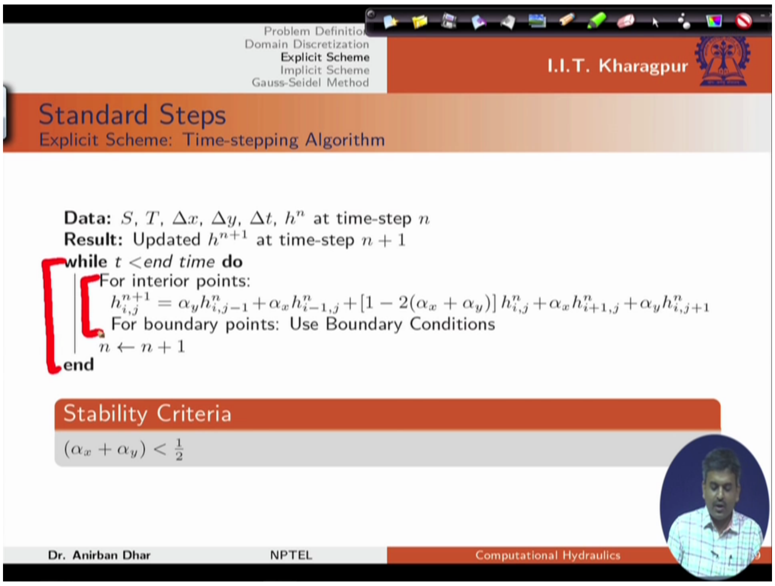
Second Order Discretization

$$\frac{-3h_{i,1}^{n+1} + 4h_{i,2}^{n+1} - h_{i,3}^{n+1}}{2\Delta y} = 0$$

$$h_{i,1}^{n+1} = \frac{4}{3}h_{i,2}^{n+1} - \frac{1}{3}h_{i,3}^{n+1}$$

Now standard steps for explicit time stepping algorithm, we need to define (del) S, T, del x, del y, del t, hn at time step n. That means at every time step we should have value available. But initially when n equals to zero that is the initial condition. Then what is the expectation from this algorithm? Updated value of n plus 1 level. For standard explicit algorithm we will have one time step. This is our time stepping and within that we will have interior points.

(Refer Slide Time: 09:59)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Standard Steps

Explicit Scheme: Time-stepping Algorithm

```
Data: $S$, $T$, $\Delta x$, $\Delta y$, $\Delta t$, $h^n$ at time-step $n$
Result: Updated $h^{n+1}$ at time-step $n+1$
while $t <$ end time do
    For interior points:
        $h_{i,j}^{n+1} = \alpha_y h_{i,j-1}^n + \alpha_x h_{i-1,j}^n + [1 - 2(\alpha_x + \alpha_y)]\, h_{i,j}^n + \alpha_x h_{i+1,j}^n + \alpha_y h_{i,j+1}^n$
        For boundary points: Use Boundary Conditions
    $n \leftarrow n+1$
end
```

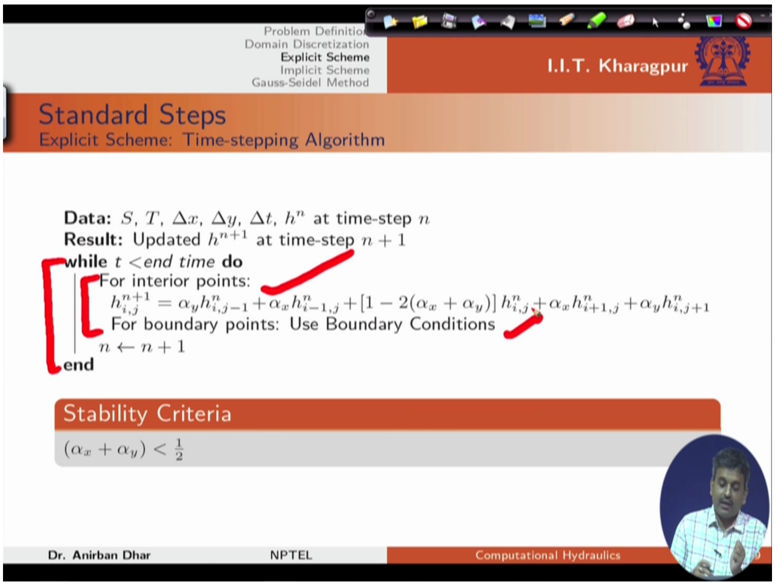
Stability Criteria

$(\alpha_x + \alpha_y) < \frac{1}{2}$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So in case of explicit algorithm first we need to solve the thing for interior points then we will apply the boundary conditions. So first we will solve it for interior points and then with all known values of the interior points we can calculate the boundary point values.

(Refer Slide Time: 10:27)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Standard Steps

Explicit Scheme: Time-stepping Algorithm

Data: S, T, Δx, Δy, Δt, h^n at time-step n
Result: Updated h^{n+1} at time-step $n+1$
while $t <$ end time do
For interior points:
$h_{i,j}^{n+1} = \alpha_y h_{i,j-1}^n + \alpha_x h_{i-1,j}^n + [1 - 2(\alpha_x + \alpha_y)]\, h_{i,j}^n + \alpha_x h_{i+1,j}^n + \alpha_y h_{i,j+1}^n$
For boundary points: Use Boundary Conditions
$n \leftarrow n+1$
end

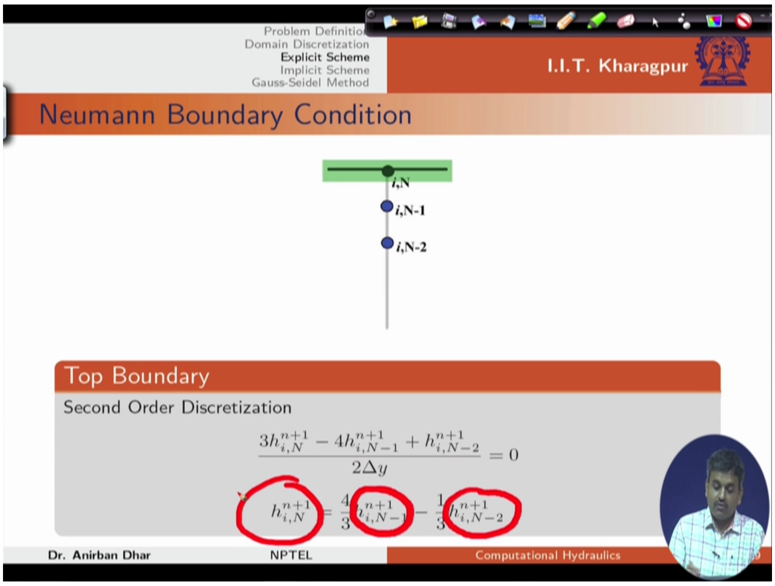
Stability Criteria

$(\alpha_x + \alpha_y) < \frac{1}{2}$

Dr. Anirban Dhar NPTEL Computational Hydraulics

That is why in our previous case where we have this Neumann boundary condition for top and bottom, these values are actually internal values or values for interior nodes. So after calculation from the interior nodes I can directly transfer the values to boundary nodes. So that is why it is written in this form.

(Refer Slide Time: 11:05)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Neumann Boundary Condition

Top Boundary

Second Order Discretization

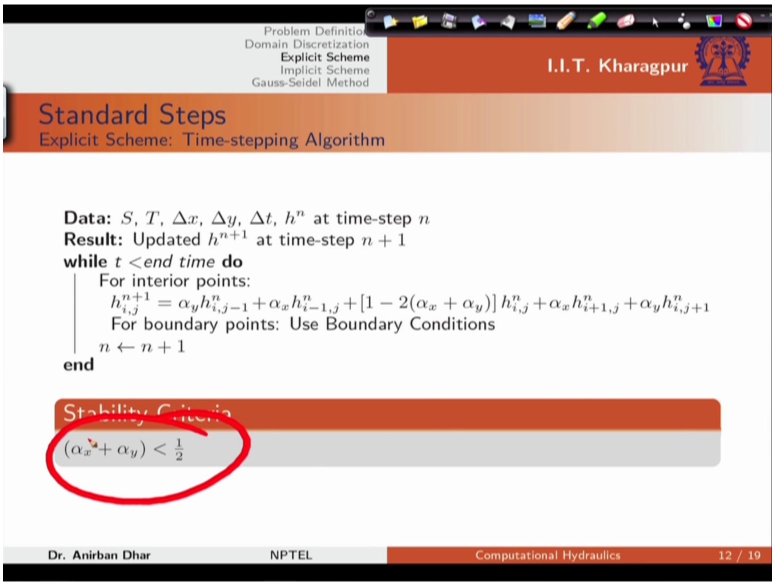
$$\frac{3h_{i,N}^{n+1} - 4h_{i,N-1}^{n+1} + h_{i,N-2}^{n+1}}{2\Delta y} = 0$$

$$h_{i,N}^{n+1} = \frac{4}{3}h_{i,N-1}^{n+1} - \frac{1}{3}h_{i,N-2}^{n+1}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So now we can proceed towards the programming thing. And interesting point is that we need to satisfy the stability criteria. We know that explicit scheme is conditionally stable. So obviously, alpha x and alpha y should be less than half. That we have seen from our stability analysis steps.

(Refer Slide Time: 11:54)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Standard Steps

Explicit Scheme: Time-stepping Algorithm

```
Data: $S$, $T$, $\Delta x$, $\Delta y$, $\Delta t$, $h^n$ at time-step $n$
Result: Updated $h^{n+1}$ at time-step $n+1$
while $t <$ end time do
    For interior points:
      $h_{i,j}^{n+1} = \alpha_y h_{i,j-1}^n + \alpha_x h_{i-1,j}^n + [1 - 2(\alpha_x + \alpha_y)]\, h_{i,j}^n + \alpha_x h_{i+1,j}^n + \alpha_y h_{i,j+1}^n$
      For boundary points: Use Boundary Conditions
    $n \leftarrow n+1$
end
```

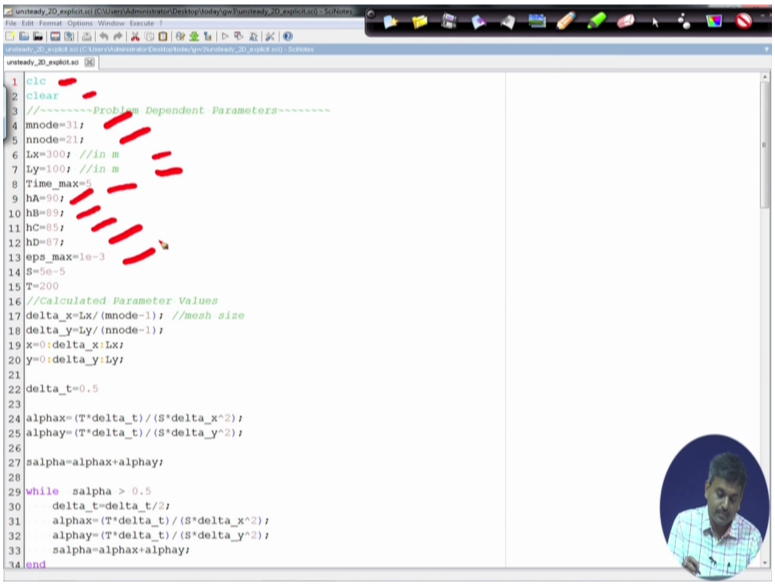
Stability Criteria

$(\alpha_x + \alpha_y) < \frac{1}{2}$

Dr. Anirban Dhar NPTEL Computational Hydraulics 12 / 19

So if we open our source code which is again from scilab. Okay this is unsteady explicit. And unsteady explicit, let us start the problem with clc and clear. That means clearing console and clearing variables. M node is the number of nodes in x direction, n node is the number of nodes in the y direction. Lx, Ly, T max let us say time maximum is 5 units here, 5 days maybe. This is hA equals to 90, 89, 85, 87. Epsilon max, this is required for the convergence of the space loop.

(Refer Slide Time: 13:09)



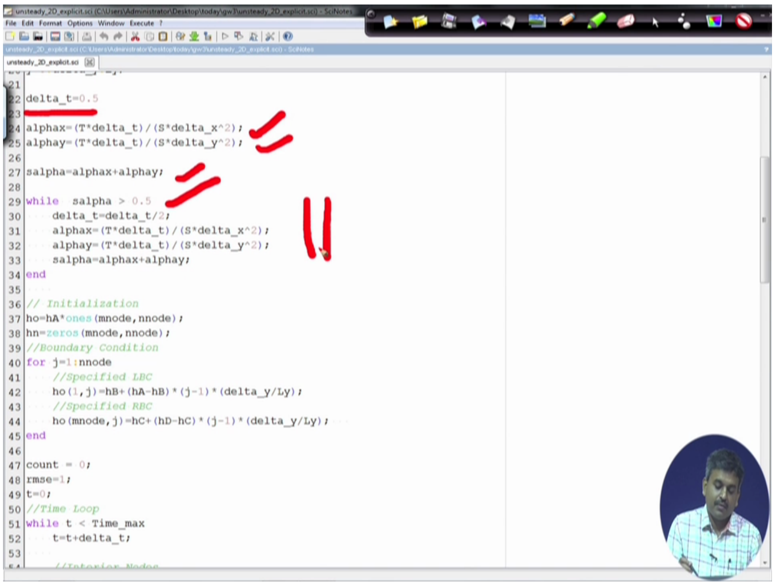
unsteady_2D_explicit.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_explicit.sci) - SciNotes
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci

```
clc
clear
//~~~~~~~~~Problem Dependent Parameters~~~~~~~~
mnode=31;
nnode=21;
Lx=300; //in m
Ly=100; //in m
Time_max=5
hA=90;
hB=89;
hC=85;
hD=87;
eps_max=1e-3
S=5e-5
T=200
//Calculated Parameter Values
delta_x=Lx/(mnode-1); //mesh size
delta_y=Ly/(nnode-1);
x=0:delta_x:Lx;
y=0:delta_y:Ly;

delta_t=0.5

alphax=(T*delta_t)/(S*delta_x^2);
alphay=(T*delta_t)/(S*delta_y^2);

salpha=alphax+alphay;

while  salpha > 0.5
    delta_t=delta_t/2;
    alphax=(T*delta_t)/(S*delta_x^2);
    alphay=(T*delta_t)/(S*delta_y^2);
    salpha=alphax+alphay;
end
```

Delta x, delta y, let us say delta t equals to point 5. We are assuming the delta t value. Then we have to again recalculate it based on alpha x and alpha y value. Okay. Initially I have calculated this delta t equals to point 5. Now alpha x can be calculated like this, alpha x and alpha y based on our definition. So s alpha or sum alpha is alpha x plus alpha y.

Ideally this (alph) s alpha, if this (alph) s alpha is greater than point 5, I should reduce this delta t value so that we can solve this problem. This criterion should be satisfied otherwise we cannot solve this explicit problem.

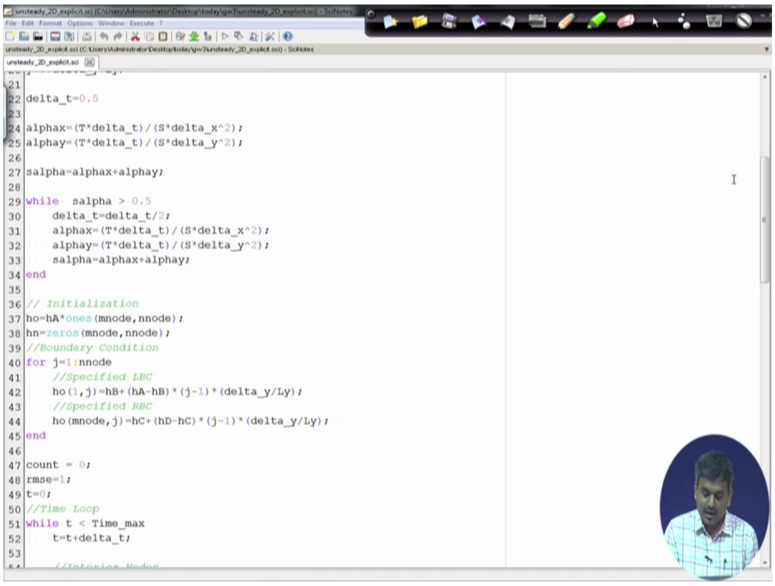
(Refer Slide Time: 14:12)


unsteady_2D_explicit.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_explicit.sci) - SciNotes
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci

```
delta_t=0.5

alphax=(T*delta_t)/(S*delta_x^2);
alphay=(T*delta_t)/(S*delta_y^2);

salpha=alphax+alphay;

while  salpha > 0.5
    delta_t=delta_t/2;
    alphax=(T*delta_t)/(S*delta_x^2);
    alphay=(T*delta_t)/(S*delta_y^2);
    salpha=alphax+alphay;
end

// Initialization
ho=hA*ones(mnode,nnode);
hn=zeros(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
    ho(1,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
    //Specified RBC
    ho(mnode,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
end

count = 0;
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;
```

So if delta t equals to delta t by 2, then again I can calculate delta t and calculate s alpha. So within while loop I can check and assign new delta t value based on the requirement of the problem. It also depends on the parameter values t, s and other values.

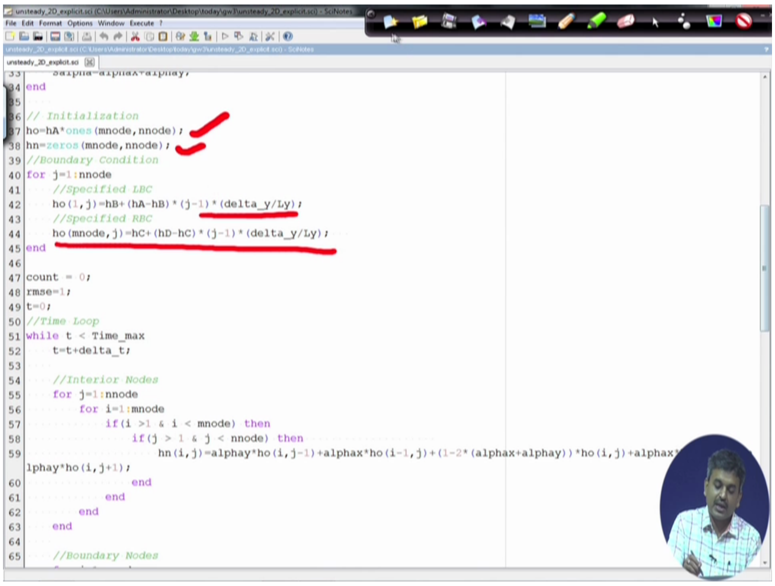
(Refer Slide Time: 14:39)


unsteady_2D_explicit.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_explicit.sci) - SciNotes
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci

```
delta_t=0.5

alphax=(T*delta_t)/(S*delta_x^2);
alphay=(T*delta_t)/(S*delta_y^2);

salpha=alphax+alphay;

while  salpha > 0.5
    delta_t=delta_t/2;
    alphax=(T*delta_t)/(S*delta_x^2);
    alphay=(T*delta_t)/(S*delta_y^2);
    salpha=alphax+alphay;
end

// Initialization
ho=hA*ones(mnode,nnode);
hn=zeros(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
    ho(1,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
    //Specified RBC
    ho(mnode,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
end

count = 0;
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;
```

So initialisation this ho is the old time level, hn this is a new time level. Ho, let us say we are starting with the value 90 because we are solving that steady state problem again with the unsteady state framework. So boundary condition, we need to initially specify the boundary condition. Ho or initial case or left and right boundaries I have specified, this specified boundary conditions here.

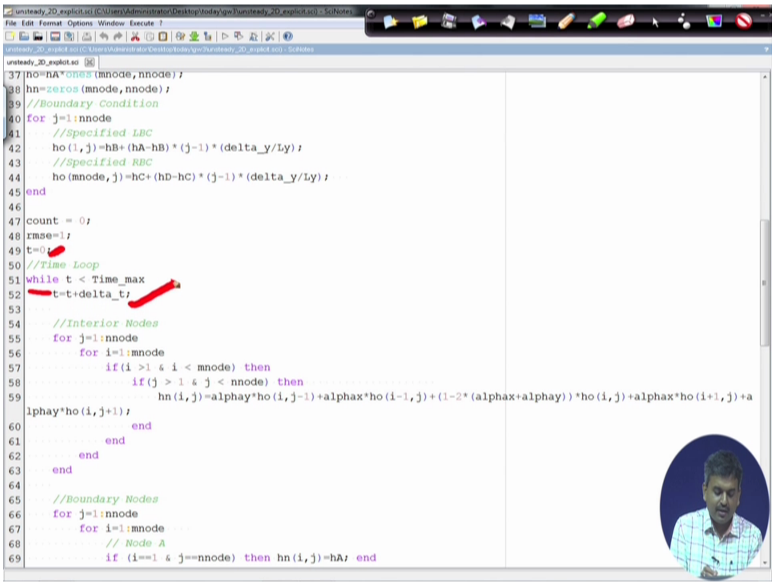
(Refer Slide Time: 15:16)



```
    salpha=alphax+alphay;
end

// Initialization
ho=hA*ones(mnode,nnode);
hn=zeros(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
    ho(1,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
    //Specified RBC
    ho(mnode,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
end

count = 0;
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;

    //Interior Nodes
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=alphay*ho(i,j-1)+alphax*ho(i-1,j)+(1-2*(alphax+alphay))*ho(i,j)+alphax
lphay*ho(i,j+1);
                end
            end
        end
    end

    //Boundary Nodes
```

Now if I see my general format I have count, rmse equals to 1, t equals to zero here. Now in this case we need to do certain things. First is our time loop, this while t less than t max. We have specified that t max value and we are starting from t equals to zero point. Obviously then the program will be executed within this while loop, t equals to t plus delta t. So this is the updated time level.

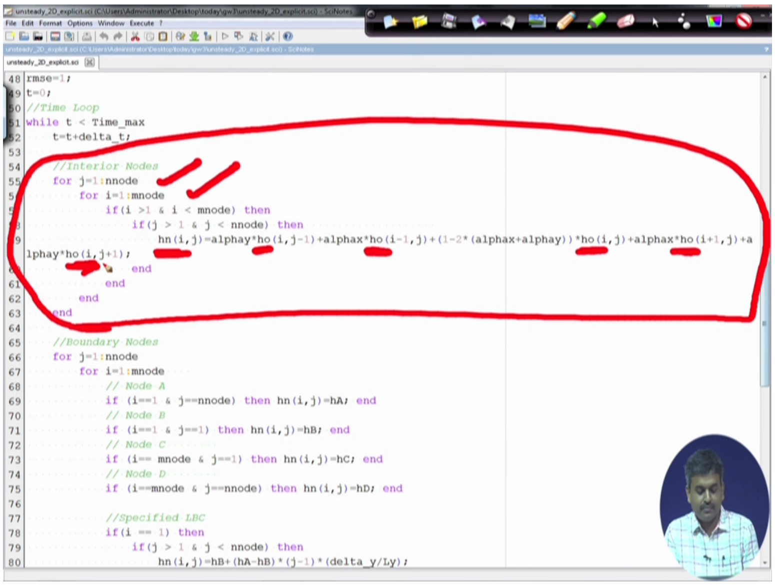
(Refer Slide Time: 16:12)



```
ho=hA*ones(mnode,nnode);
hn=zeros(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
    ho(1,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
    //Specified RBC
    ho(mnode,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
end

count = 0;
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;

    //Interior Nodes
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=alphay*ho(i,j-1)+alphax*ho(i-1,j)+(1-2*(alphax+alphay))*ho(i,j)+alphax*ho(i+1,j)+a
lphay*ho(i,j+1);
                end
            end
        end
    end

    //Boundary Nodes
    for j=1:nnode
        for i=1:mnode
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
```

Now in this case first we need to solve the thing for interior nodes. So j starting from 1 to n node, i m node, i greater than 1, i less than m node. So for all interior points we can solve using our standard explicit equation. In this case hn is the new time level value, ho which is specified on the right hand side, all old time level values.

(Refer Slide Time: 16:53)

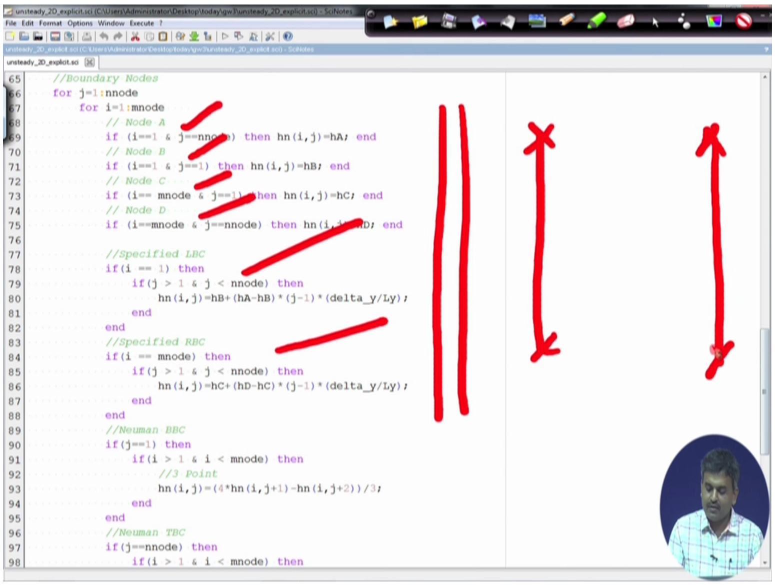


unsteady_2D_explicit.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_explicit.sci) - SciNotes
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci

```
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;

    //Interior Nodes
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=alphay*ho(i,j-1)+alphax*ho(i-1,j)+(1-2*(alphax+alphay))*ho(i,j)+alphax*ho(i+1,j)+alphay*ho(i,j+1);
                end
            end
        end
    end

    //Boundary Nodes
    for j=1:nnode
        for i=1:mnode
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
```

Now after getting these values we need to update the things. Boundary nodes, these are specified. We should not worry about the boundary nodes because boundary nodes are specified values. So for point A, point B, point C, point D we can directly specify. Again for left boundary, right boundary we can directly specify the things. So for left boundary, right boundary, including the end points I have specified the values.

(Refer Slide Time: 17:31)

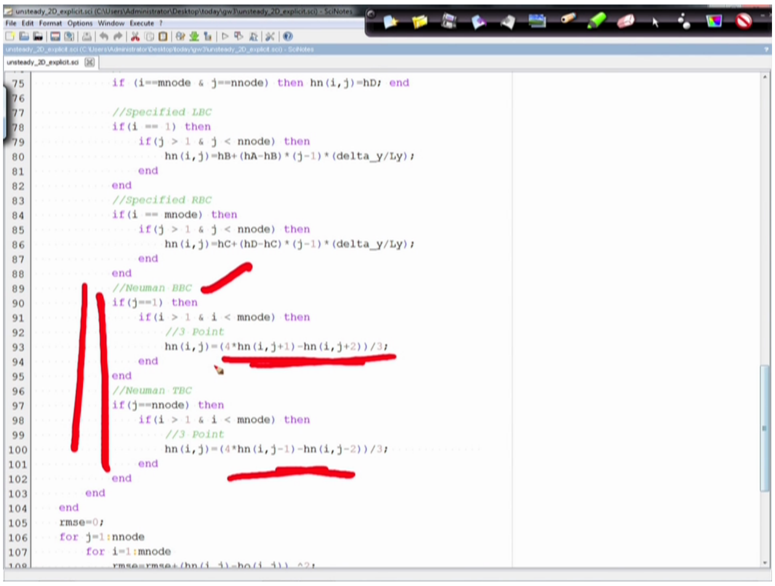


unsteady_2D_explicit.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_explicit.sci) - SciNotes
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci

```
    //Boundary Nodes
    for j=1:nnode
        for i=1:mnode
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
                end
            end
            //Specified RBC
            if(i == mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
                end
            end
            //Neuman BBC
            if(j==1) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
                end
            end
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
```

With this we need to specify the things for Neumann boundary condition. So Neumann boundary condition, in this case this is bottom boundary. So for bottom boundary we can directly write our three point case and we can update the solutions.

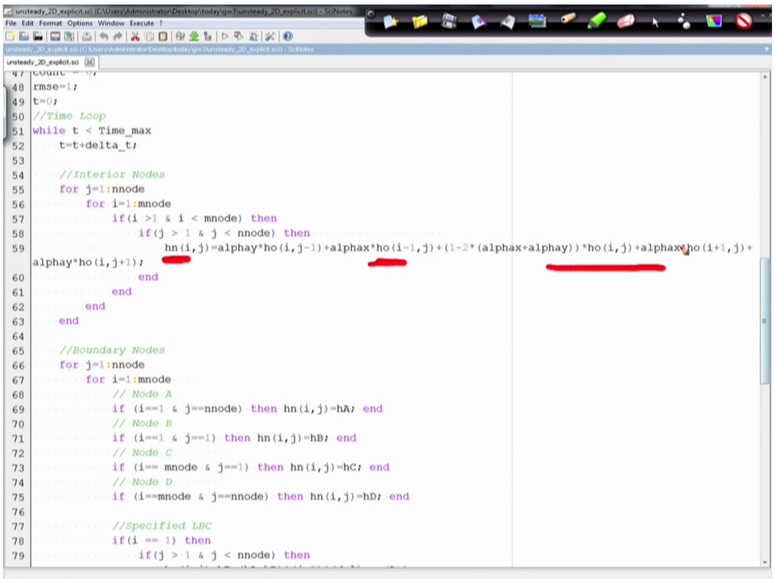
(Refer Slide Time: 18:04)



```
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
                end
            end
            //Specified RBC
            if(i == mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
                end
            end
            //Neuman BBC
            if(j==1) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
                end
            end
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j-1)-hn(i,j-2))/3;
                end
            end
        end
    end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
```

In this case where we have used our interior points we can see that on the left hand side we have n which is future time level value and ho which is previous time level value on the right hand side.

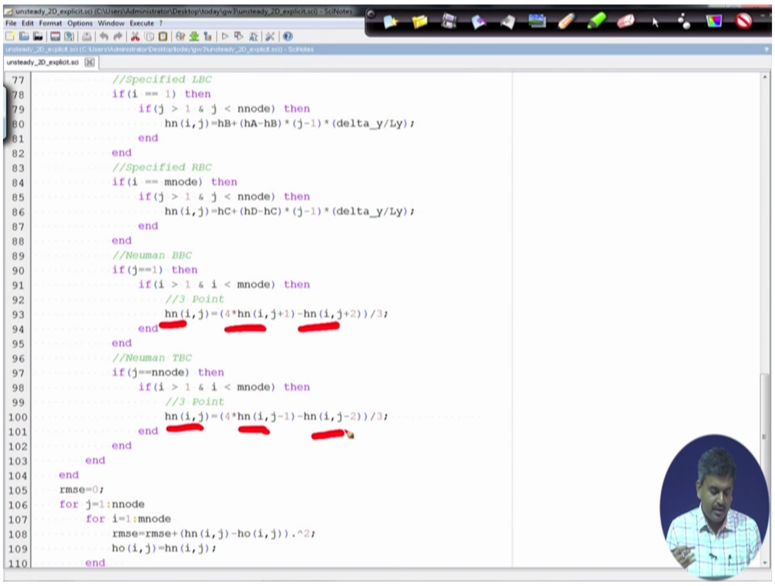
(Refer Slide Time: 18:23)



```
count = 0;
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;

    //Interior Nodes
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=alphay*ho(i,j-1)+alphax*ho(i-1,j)+(1-2*(alphax+alphay))*ho(i,j)+alphax*ho(i+1,j)+
alphay*ho(i,j+1);
                end
            end
        end
    end

    //Boundary Nodes
    for j=1:nnode
        for i=1:mnode
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
```

But in case of boundary conditions we are specifying both new time level values here on both sides. That means based on the new time level values only we are calculating the updated values.

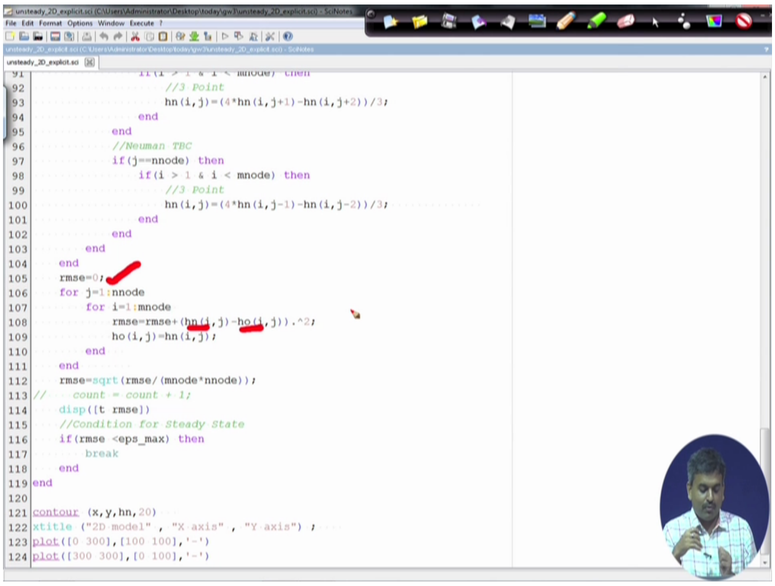
(Refer Slide Time: 18:48)



```
            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
                end
            end
            //Specified RBC
            if(i == mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
                end
            end
            //Neuman BBC
            if(j==1) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
                end
            end
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j-1)-hn(i,j-2))/3;
                end
            end
        end
    end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
```

So obviously after getting all information about the future time level we can calculate the rmse for that one. So rmse equals to zero and rmse equals to rmse h new minus h old square. That means I am checking whether the previous time level value and future time level value, both are same or not. That means for a steady state problem if you are utilising unsteady state framework, these two values should be same and you should get rmse value close to zero.

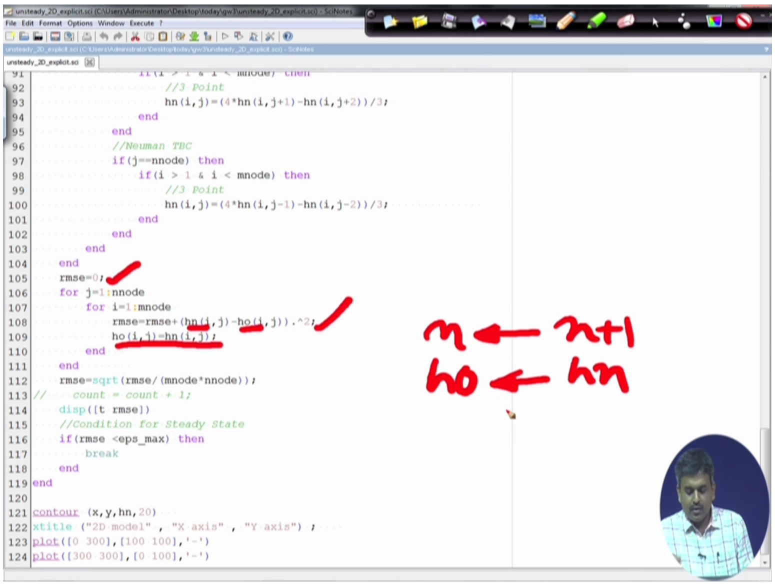
(Refer Slide Time: 19:39)



```
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
                end
            end
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j-1)-hn(i,j-2))/3;
                end
            end
        end
    end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
    end
    rmse=sqrt(rmse/(mnode*nnode));
//    count = count + 1;
    disp([t rmse])
    //Condition for Steady State
    if(rmse <eps_max) then
        break
    end
end

contour (x,y,hn,20)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

But this step will not be required for our time dependent problems. Time dependent problems only this specification n, n plus 1. What it means? That whatever value is available for nth or future time level, that will transfer to h not or ho.

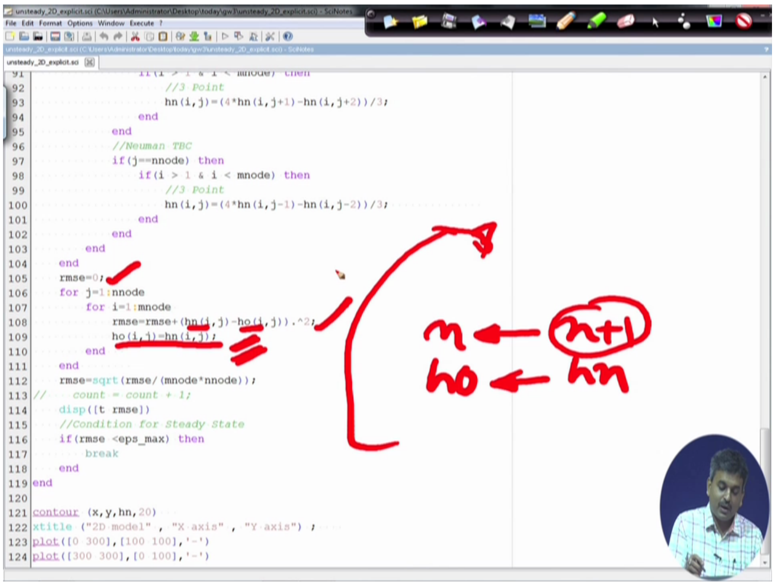
(Refer Slide Time: 20:09)



```
            if(i > 1 & i < mnode) then
                //3 Point
                hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
            end
        end
        //Neuman TBC
        if(j==nnode) then
            if(i > 1 & i < mnode) then
                //3 Point
                hn(i,j)=(4*hn(i,j-1)-hn(i,j-2))/3;
            end
        end
    end
end
rmse=0;
for j=1:nnode
    for i=1:mnode
        rmse=rmse+(hn(i,j)-ho(i,j)).^2;
        ho(i,j)=hn(i,j);
    end
end
rmse=sqrt(rmse/(mnode*nnode));
//    count = count + 1;
disp([t rmse])
//Condition for Steady State
if(rmse <eps_max) then
    break
end
end

contour (x,y,hn,20)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

So this means that for the next time level when we will be calculating the next time level value then this n plus 1 level value will be the present time level value for the next step. So that is why we need to transfer these values.

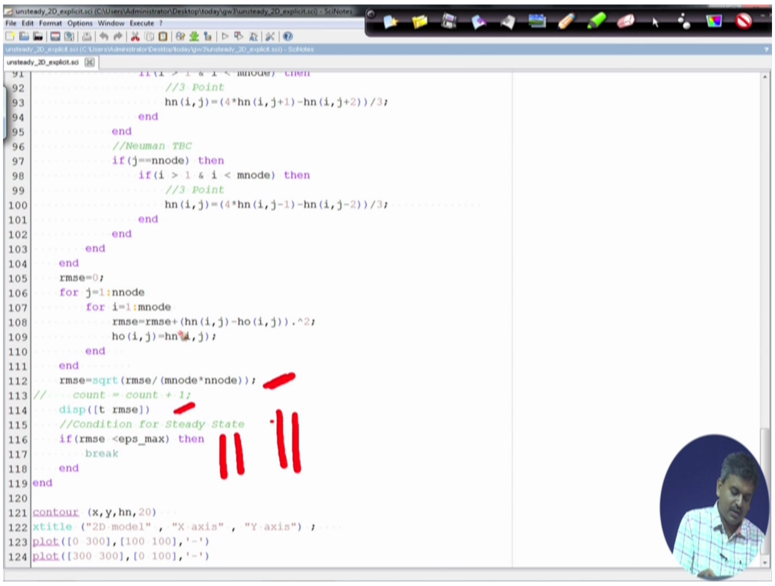
(Refer Slide Time: 20:30)



```
            if(i > 1 & i < mnode) then
                //3 Point
                hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
            end
        end
        //Neuman TBC
        if(j==nnode) then
            if(i > 1 & i < mnode) then
                //3 Point
                hn(i,j)=(4*hn(i,j-1)-hn(i,j-2))/3;
            end
        end
    end
end
rmse=0;
for j=1:nnode
    for i=1:mnode
        rmse=rmse+(hn(i,j)-ho(i,j)).^2;
        ho(i,j)=hn(i,j);
    end
end
rmse=sqrt(rmse/(mnode*nnode));
//    count = count + 1;
disp([t rmse])
//Condition for Steady State
if(rmse <eps_max) then
    break
end
end

contour (x,y,hn,20)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

And this is rmse calculation. We can directly specify rmse and t. And for steady state problem, condition for steady state, I have clearly mentioned this. This is not required for temporal problems. If you have a time dependent problem, obviously you can avoid this steps. You don't need to calculate rmse here. And finally if this rmse is less than epsilon max which is the specified range or which is the specified value for rmse below which your rmse value should be, we should break. Break means it will come out of this while loop.

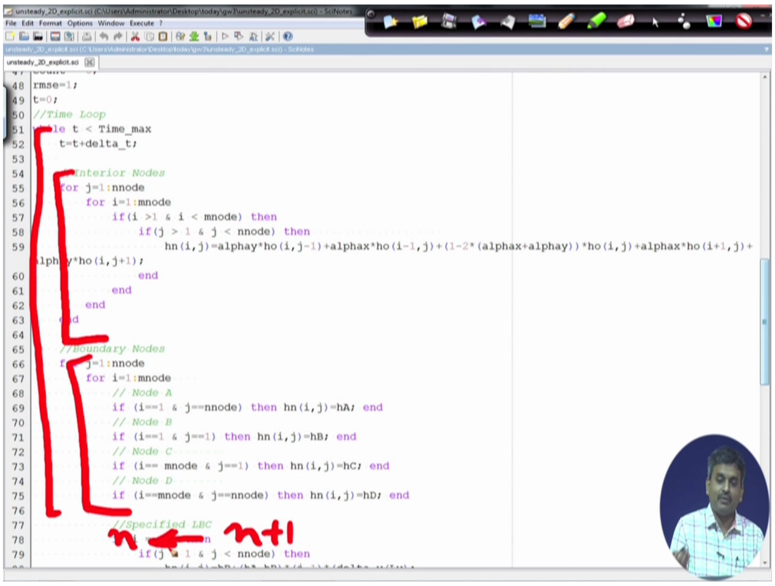
(Refer Slide Time: 21:23)



```
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j+1)-hn(i,j+2))/3;
                end
            end
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //3 Point
                    hn(i,j)=(4*hn(i,j-1)-hn(i,j-2))/3;
                end
            end
        end
    end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
    end
    rmse=sqrt(rmse/(mnode*nnode));
//    count = count + 1;
    disp([t rmse])
    //Condition for Steady State
    if(rmse <eps_max) then
        break
    end
end

contour (x,y,hn,20)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

And finally we need to draw this x, y, and hn. Hn means variation at future time level. So at this level we have seen in explicit algorithm two things are important. We have one time loop. Then in space loop we need to solve interior points first, then boundary points, then only we can get the solution and finally we need to transfer this n plus 1 values to nth level so that we can use it for future time level as previous time level value.

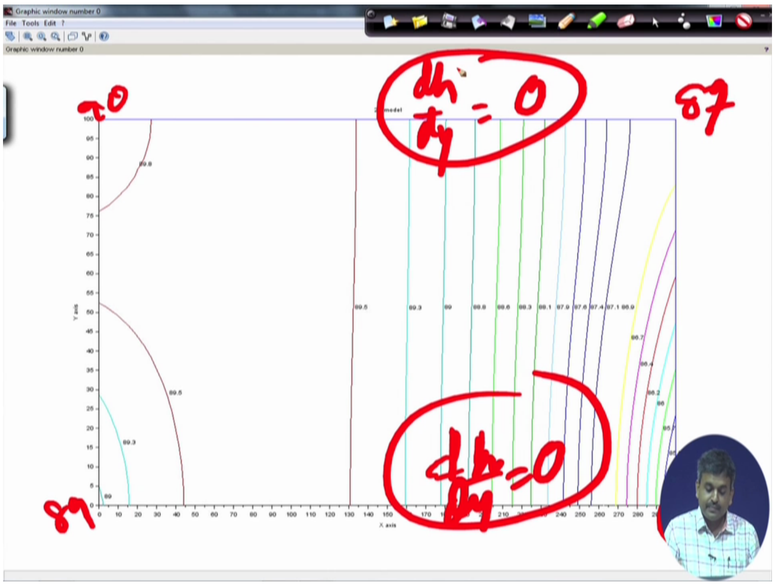
(Refer Slide Time: 22:09)



```
rmse=1;
t=0;
//Time Loop
while t < Time_max
    t=t+delta_t;

    //Interior Nodes
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=alphay*ho(i,j-1)+alphax*ho(i-1,j)+(1-2*(alphax+alphay))*ho(i,j)+alphax*ho(i+1,j)+
alphay*ho(i,j+1);
                end
            end
        end
    end

    //Boundary Nodes
    for j=1:nnode
        for i=1:mnode
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            
                if(j > 1 & j < nnode) then
```

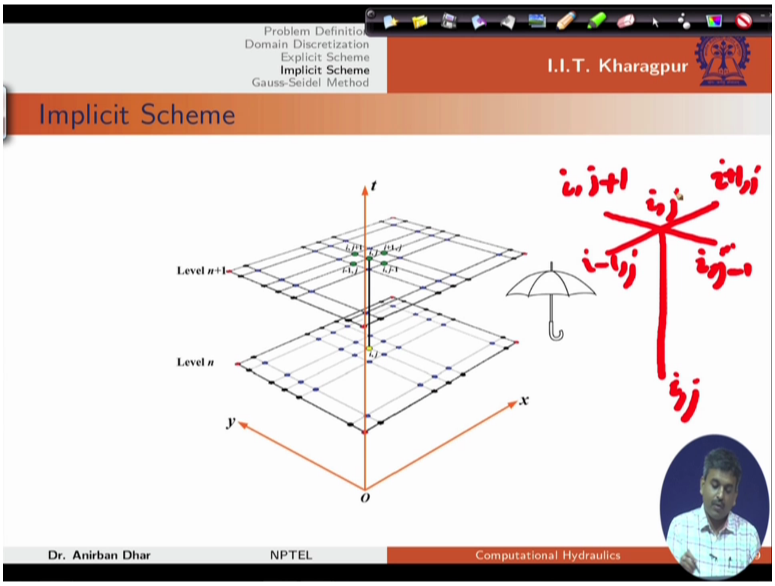
Now I can run it, just evaluate it. So on the left hand side you can see these are time levels, on the right hand side we have values. Now it is interesting part is this is somewhat different contours we are getting. But the solutions are matching because this side we have specified with 90, this is 89, this is 89, this is 87, this is 85. So more or less again in this case dh by dy, this is satisfied. This is dh by dy equals to zero. These two are satisfied here.

(Refer Slide Time: 23:14)



So we can say that you have got almost a steady state solution from your unsteady state framework. Now if we see the standard implicit algorithm. In implicit algorithm for central point only we are using the previous time level value, ij point. For other points we can have, this is ij plus 1, i plus 1j, this is i minus 1j, ij minus 1, ij. For these points we can get the implicit formulation.

(Refer Slide Time: 24:17)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

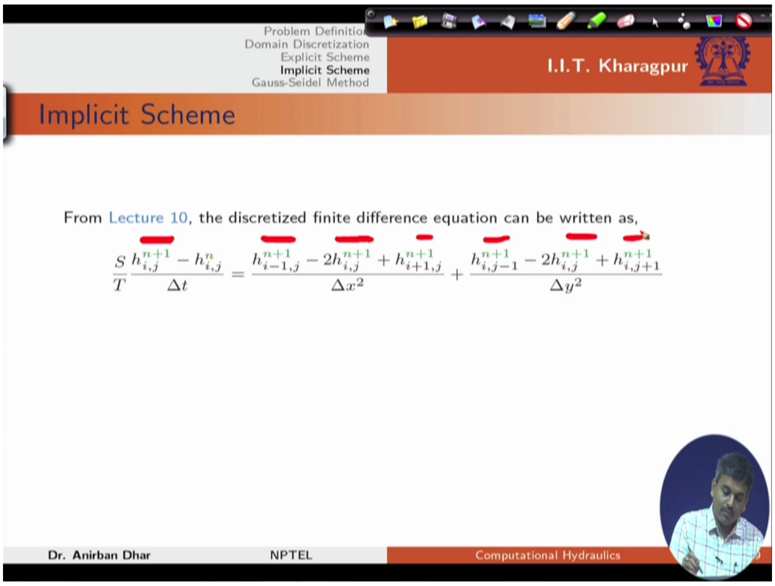
I.I.T. Kharagpur

Implicit Scheme

Dr. Anirban Dhar NPTEL Computational Hydraulics

From our lecture number 10 we can discretize it. And interesting point is that, these all are future time level values. That means spatial derivatives are evaluated at future time level. That is a basic thing for implicit discretization. That is from discretization point of view.

(Refer Slide Time: 24:42)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Implicit Scheme

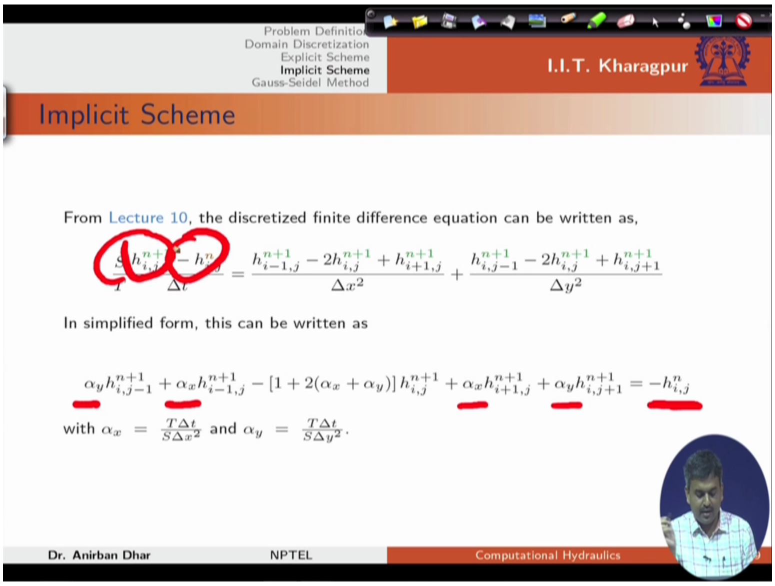
From Lecture 10, the discretized finite difference equation can be written as,

$$\frac{S}{T}\frac{h_{i,j}^{n+1}-h_{i,j}^{n}}{\Delta t}=\frac{h_{i-1,j}^{n+1}-2h_{i,j}^{n+1}+h_{i+1,j}^{n+1}}{\Delta x^2}+\frac{h_{i,j-1}^{n+1}-2h_{i,j}^{n+1}+h_{i,j+1}^{n+1}}{\Delta y^2}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now we can see what is the solution approach here? Again we can write everything in simplified format and we can transfer this thing on the right hand side. So we will get everything in terms of alpha y, alpha x, alpha x alpha y. And this is on the right hand side as negative because this is there.

(Refer Slide Time: 25:13)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Implicit Scheme

From Lecture 10, the discretized finite difference equation can be written as,

$$\frac{S}{T}\frac{h^{n+1}_{i,j}-h^{n}_{i,j}}{\Delta t}=\frac{h^{n+1}_{i-1,j}-2h^{n+1}_{i,j}+h^{n+1}_{i+1,j}}{\Delta x^2}+\frac{h^{n+1}_{i,j-1}-2h^{n+1}_{i,j}+h^{n+1}_{i,j+1}}{\Delta y^2}$$

In simplified form, this can be written as

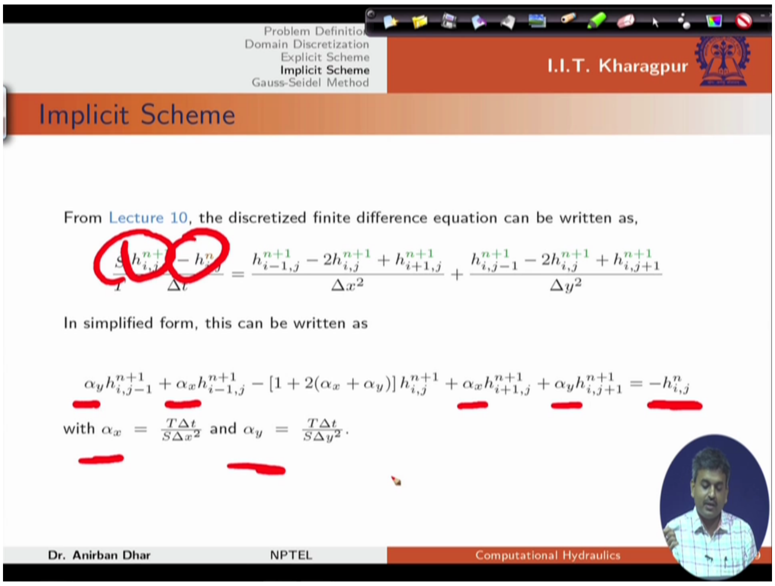
$$\alpha_y h^{n+1}_{i,j-1}+\alpha_x h^{n+1}_{i-1,j}-[1+2(\alpha_x+\alpha_y)]\,h^{n+1}_{i,j}+\alpha_x h^{n+1}_{i+1,j}+\alpha_y h^{n+1}_{i,j+1}=-h^{n}_{i,j}$$

with $\alpha_x = \frac{T\Delta t}{S\Delta x^2}$ and $\alpha_y = \frac{T\Delta t}{S\Delta y^2}$.

Dr. Anirban Dhar NPTEL Computational Hydraulics

And alpha x and alpha y, these are defined like our previous explicit algorithm.

(Refer Slide Time: 25:19)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Implicit Scheme

From Lecture 10, the discretized finite difference equation can be written as,

$$\frac{S}{T}\frac{h^{n+1}_{i,j}-h^{n}_{i,j}}{\Delta t}=\frac{h^{n+1}_{i-1,j}-2h^{n+1}_{i,j}+h^{n+1}_{i+1,j}}{\Delta x^2}+\frac{h^{n+1}_{i,j-1}-2h^{n+1}_{i,j}+h^{n+1}_{i,j+1}}{\Delta y^2}$$

In simplified form, this can be written as

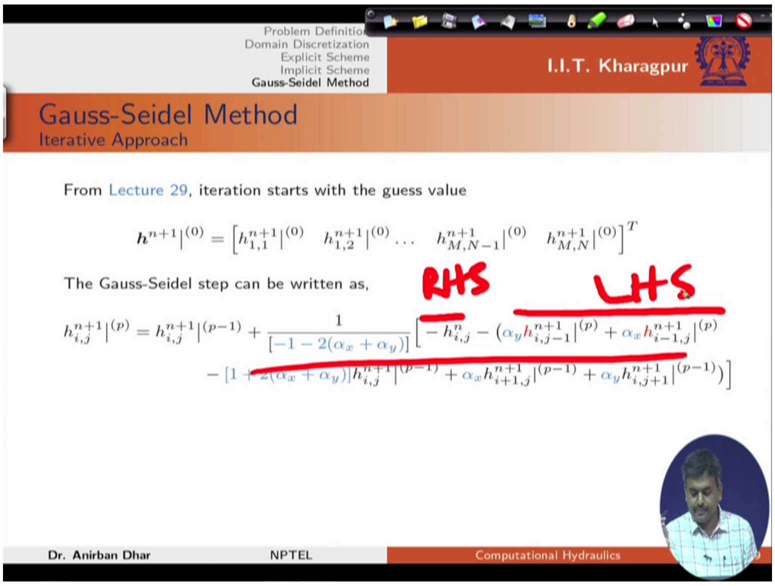
$$\alpha_y h^{n+1}_{i,j-1}+\alpha_x h^{n+1}_{i-1,j}-[1+2(\alpha_x+\alpha_y)]\,h^{n+1}_{i,j}+\alpha_x h^{n+1}_{i+1,j}+\alpha_y h^{n+1}_{i,j+1}=-h^{n}_{i,j}$$

with $\alpha_x = \frac{T\Delta t}{S\Delta x^2}$ and $\alpha_y = \frac{T\Delta t}{S\Delta y^2}$.

Dr. Anirban Dhar NPTEL Computational Hydraulics

It should be noted that implicit algorithm is unconditionally stable. So we don't need that kind of criteria we have utilised for our previous case. Now we can use our Gauss Seidel approach to solve this one. If we use our Gauss Seidel thing, this is same, this is your right hand side and rest of the things, this is left hand side.

(Refer Slide Time: 25:58)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Gauss-Seidel Method
Iterative Approach

From Lecture 29, iteration starts with the guess value

$$\boldsymbol{h}^{n+1}|^{(0)} = \left[h_{1,1}^{n+1}|^{(0)} \quad h_{1,2}^{n+1}|^{(0)} \dots \quad h_{M,N-1}^{n+1}|^{(0)} \quad h_{M,N}^{n+1}|^{(0)}\right]^T$$

The Gauss-Seidel step can be written as,

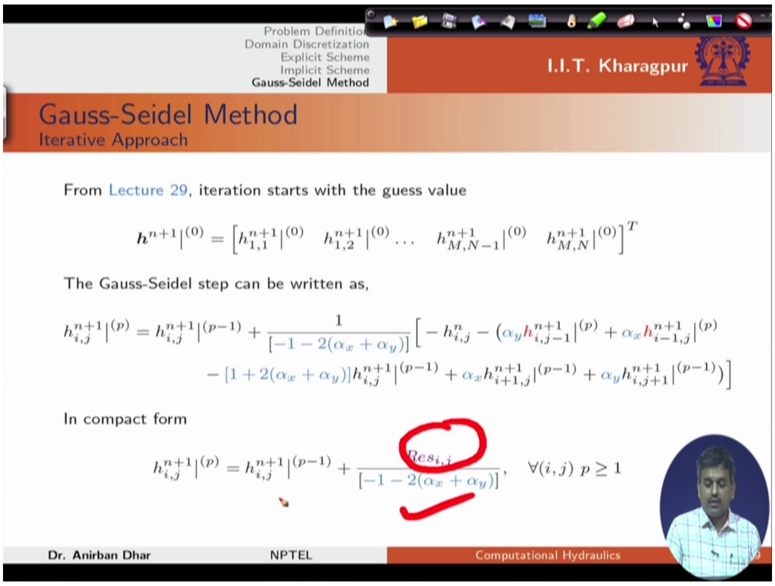
RHS LHS

$$h_{i,j}^{n+1}|^{(p)} = h_{i,j}^{n+1}|^{(p-1)} + \frac{1}{[-1-2(\alpha_x+\alpha_y)]}\Big[-h_{i,j}^{n} - (\alpha_y h_{i,j-1}^{n+1}|^{(p)} + \alpha_x h_{i-1,j}^{n+1}|^{(p)}$$
$$- [1+2(\alpha_x+\alpha_y)]h_{i,j}^{n+1}|^{(p-1)} + \alpha_x h_{i+1,j}^{n+1}|^{(p-1)} + \alpha_y h_{i,j+1}^{n+1}|^{(p-1)})\Big]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now right hand side minus left hand side, that is the basic things for Gauss Seidel algorithm. So we have this residual thing. So residual ij divided by the central coefficient here. So now with this format we need to iterate.

(Refer Slide Time: 26:27)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Gauss-Seidel Method
Iterative Approach

From Lecture 29, iteration starts with the guess value

$$\boldsymbol{h}^{n+1}|^{(0)} = \left[h_{1,1}^{n+1}|^{(0)} \quad h_{1,2}^{n+1}|^{(0)} \ldots \quad h_{M,N-1}^{n+1}|^{(0)} \quad h_{M,N}^{n+1}|^{(0)}\right]^T$$

The Gauss-Seidel step can be written as,

$$h_{i,j}^{n+1}|^{(p)} = h_{i,j}^{n+1}|^{(p-1)} + \frac{1}{[-1-2(\alpha_x+\alpha_y)]}\Big[-h_{i,j}^{n} - (\alpha_y h_{i,j-1}^{n+1}|^{(p)} + \alpha_x h_{i-1,j}^{n+1}|^{(p)}$$
$$- [1+2(\alpha_x+\alpha_y)]h_{i,j}^{n+1}|^{(p-1)} + \alpha_x h_{i+1,j}^{n+1}|^{(p-1)} + \alpha_y h_{i,j+1}^{n+1}|^{(p-1)})\Big]$$

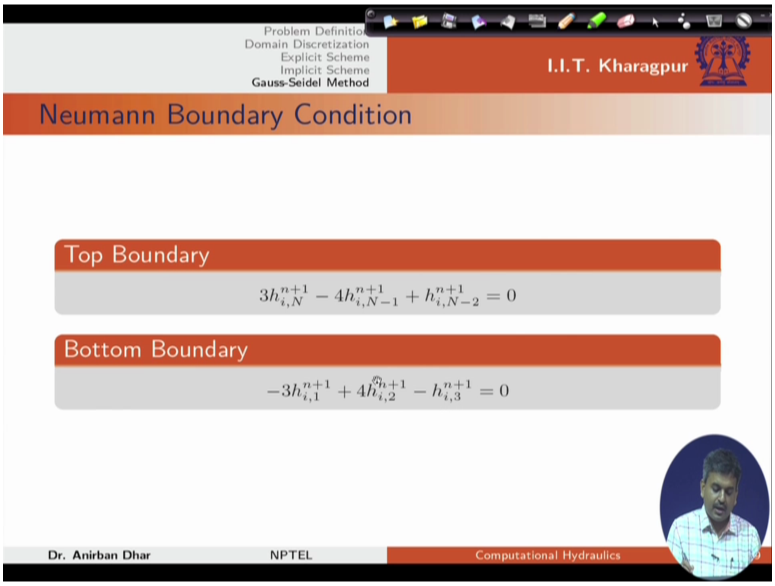
In compact form

$$h_{i,j}^{n+1}|^{(p)} = h_{i,j}^{n+1}|^{(p-1)} + \frac{Res_{i,j}}{[-1-2(\alpha_x+\alpha_y)]}, \quad \forall (i,j)\ p \geq 1$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So, how it is different from our previous explicit algorithm? Algorithm wise there is again time loop but there is no space loop iteration available for explicit. Explicit is one step method. But in case of our implicit algorithm we need to iterate on the space loop also. For top boundary and bottom boundary, these are the conditions that we need to implement.

(Refer Slide Time: 27:09)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Neumann Boundary Condition

Top Boundary

$$3h_{i,N}^{n+1} - 4h_{i,N-1}^{n+1} + h_{i,N-2}^{n+1} = 0$$

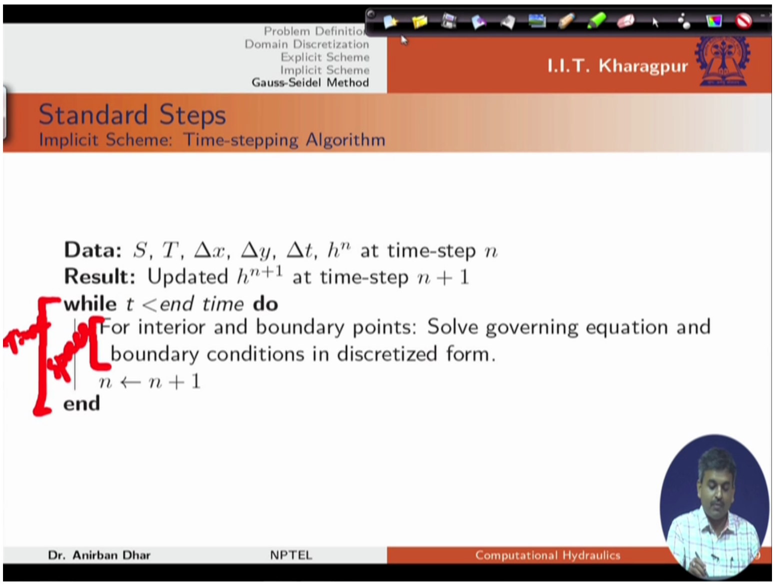
Bottom Boundary

$$-3h_{i,1}^{n+1} + 4h_{i,2}^{n+1} - h_{i,3}^{n+1} = 0$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now standard steps, like our explicit algorithm we have similar kind of steps. But in this case we need to solve this simultaneously. In our explicit algorithm we have seen that first you need to solve the interior points then you need to update the boundary points based on interior values. So here outside is one time loop and inside we have a space loop. This is time loop, this is space loop. In space we need to solve this problem.

(Refer Slide Time: 28:03)



Problem Definition
Domain Discretization
Explicit Scheme
Implicit Scheme
Gauss-Seidel Method

I.I.T. Kharagpur

Standard Steps

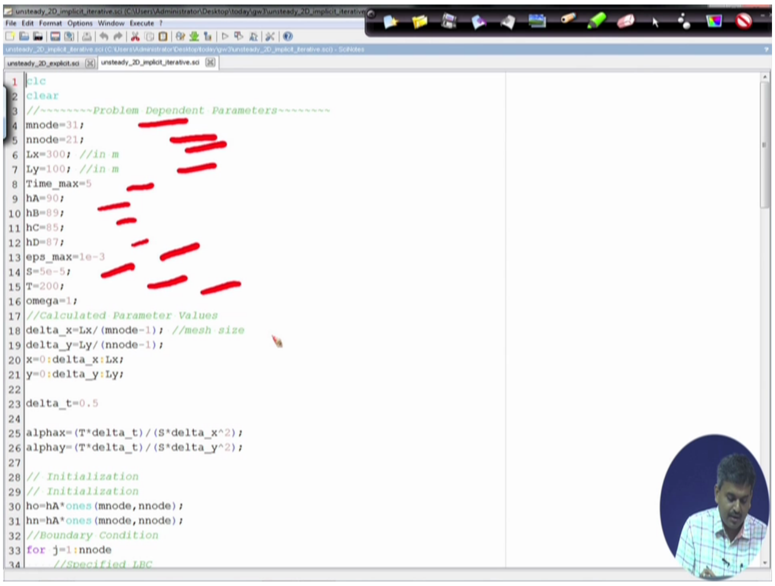
Implicit Scheme: Time-stepping Algorithm

```
Data: $S$, $T$, $\Delta x$, $\Delta y$, $\Delta t$, $h^n$ at time-step $n$
Result: Updated $h^{n+1}$ at time-step $n+1$
while $t <$ end time do
    For interior and boundary points: Solve governing equation and
    boundary conditions in discretized form.
    $n \leftarrow n+1$
end
```

Dr. Anirban Dhar NPTEL Computational Hydraulics

So let us say how we can solve it using our iterative approach. So these parameters are same. M node, n node, 300, 100, this is 5, 98, 89, 85, 87, epsilon max is 10 to the power minus 5, S is 5 into 10 to the power minus 5, T is 200, omega is 1. We are considering Gauss Seidel here.

(Refer Slide Time: 28:42)

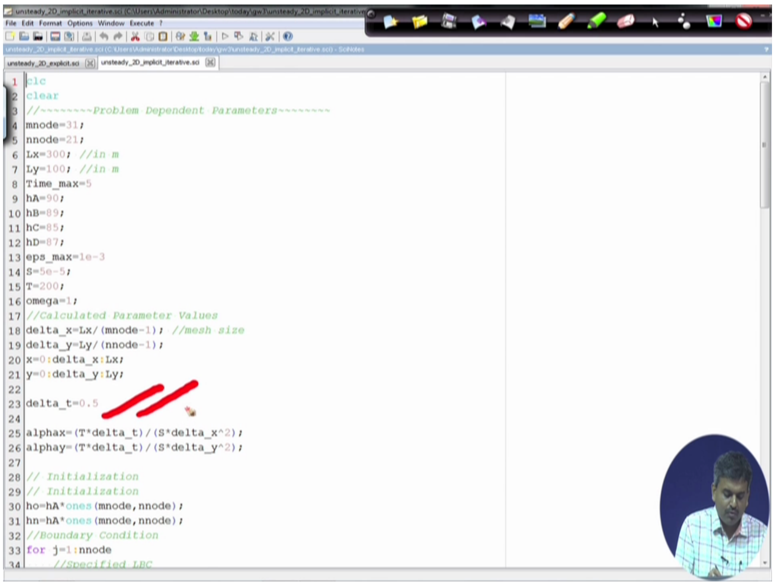


unsteady_2D_implicit_iterative.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_implicit_iterative
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci
unsteady_2D_implicit_iterative.sci

```
clc
clear
//~~~~~~~~Problem Dependent Parameters~~~~~~~~
mnode=31;
nnode=21;
Lx=300; //in m
Ly=100; //in m
Time_max=5
hA=90;
hB=89;
hC=85;
hD=87;
eps_max=1e-3
S=5e-5;
T=200;
omega=1;
//Calculated Parameter Values
delta_x=Lx/(mnode-1); //mesh size
delta_y=Ly/(nnode-1);
x=0:delta_x:Lx;
y=0:delta_y:Ly;

delta_t=0.5

alphax=(T*delta_t)/(S*delta_x^2);
alphay=(T*delta_t)/(S*delta_y^2);

// Initialization
// Initialization
ho=hA*ones(mnode,nnode);
hn=hA*ones(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
```

Calculative parameter values delta x, x, delta t again we don't need any update for delta t values. We can directly specify because we don't need to satisfy any condition here. So we can avoid that step.

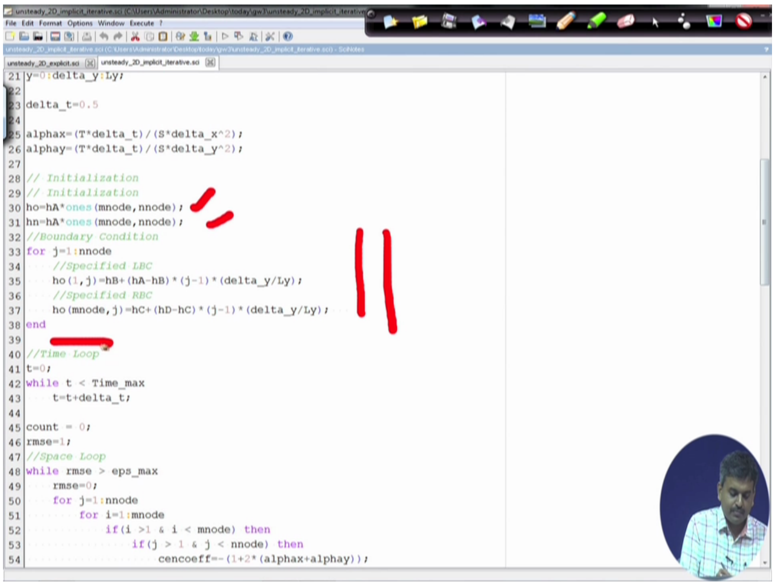
(Refer Slide Time: 29:00)



```
clc
clear
//~~~~~~~~Problem Dependent Parameters~~~~~~~~
mnode=31;
nnode=21;
Lx=300; //in m
Ly=100; //in m
Time_max=5
hA=90;
hB=89;
hC=85;
hD=87;
eps_max=1e-3
S=5e-5;
T=200;
omega=1;
//Calculated Parameter Values
delta_x=Lx/(mnode-1); //mesh size
delta_y=Ly/(nnode-1);
x=0:delta_x:Lx;
y=0:delta_y:Ly;

delta_t=0.5

alphax=(T*delta_t)/(S*delta_x^2);
alphay=(T*delta_t)/(S*delta_y^2);

// Initialization
// Initialization
ho=hA*ones(mnode,nnode);
hn=hA*ones(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
```

Now initialization, ho and hn we can initialise. These are the boundary conditions for initial condition. And there starts the time loop.

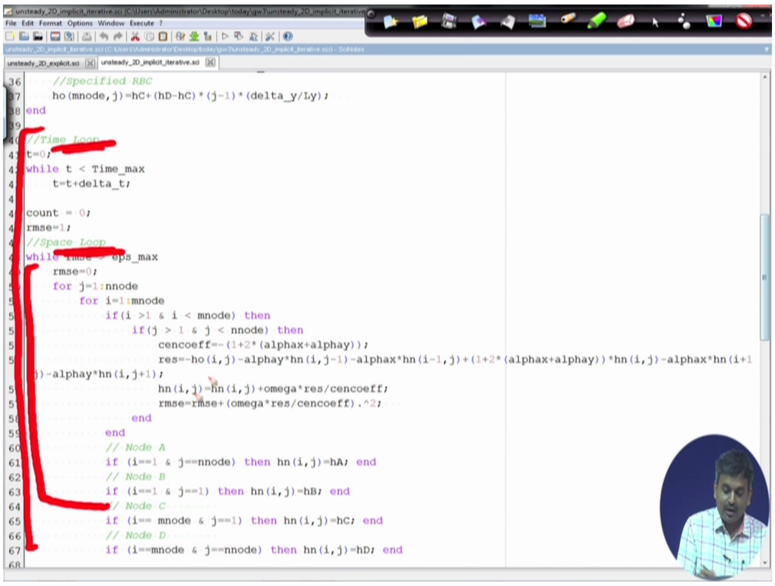
(Refer Slide Time: 29:22)



```
y=0:delta_y:Ly;

delta_t=0.5

alphax=(T*delta_t)/(S*delta_x^2);
alphay=(T*delta_t)/(S*delta_y^2);

// Initialization
// Initialization
ho=hA*ones(mnode,nnode);
hn=hA*ones(mnode,nnode);
//Boundary Condition
for j=1:nnode
    //Specified LBC
    ho(1,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
    //Specified RBC
    ho(mnode,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
end

//Time Loop
t=0;
while t < Time_max
    t=t+delta_t;

count = 0;
rmse=1;
//Space Loop
while rmse > eps_max
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    cencoeff=-(1+2*(alphax+alphay));
```

So in this case this is my time loop and again within this time loop I have a space loop. It is clearly written. So time loop starts with t equals to zero. In this case t equals to t plus delta t. Again I am starting this space loop to get correct value at the particular time level.

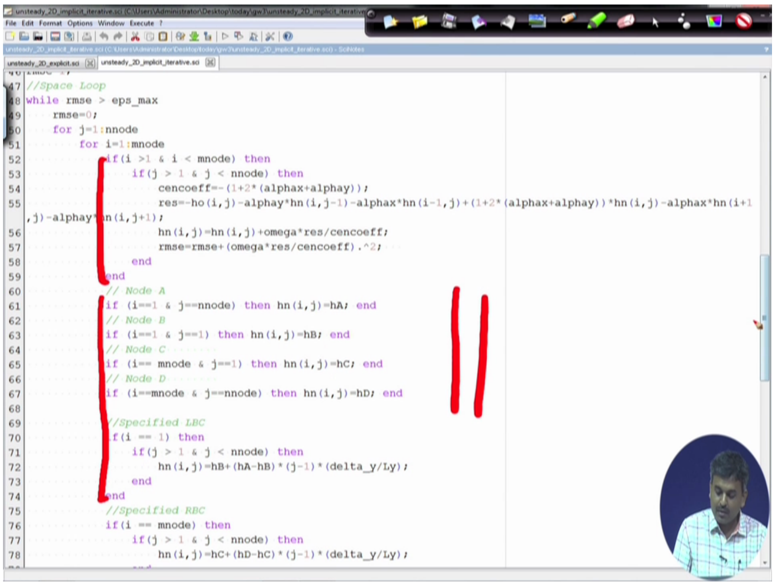
(Refer Slide Time: 29:47)



```
    //Specified RBC
    ho(mnode,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
end

//Time Loop
t=0;
while t < Time_max
    t=t+delta_t;

count = 0;
rmse=1;
//Space Loop
while rmse > eps_max
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    cencoeff=-(1+2*(alphax+alphay));
                    res=-ho(i,j)-alphay*hn(i,j-1)-alphax*hn(i-1,j)+(1+2*(alphax+alphay))*hn(i,j)-alphax*hn(i+1
,j)-alphay*hn(i,j+1);
                    hn(i,j)=hn(i,j)+omega*res/cencoeff;
                    rmse=rmse+(omega*res/cencoeff).^2;
                end
            end
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end
```

Now in this case first I have interior points within this space loop, implemented the boundary points, these are the four corner points.

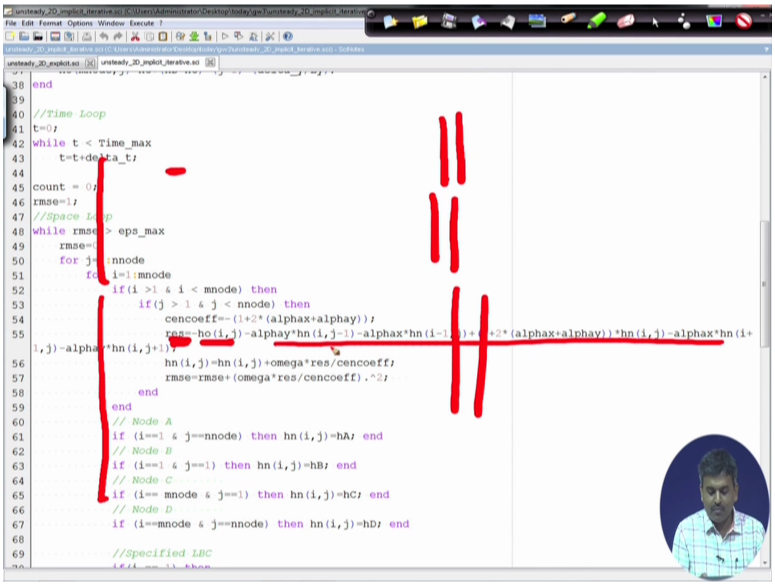
(Refer Slide Time: 30:08)



```
//Space Loop
while rmse > eps_max
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    cencoeff=-(1+2*(alphax+alphay));
                    res=-ho(i,j)-alphay*hn(i,j-1)-alphax*hn(i-1,j)+(1+2*(alphax+alphay))*hn(i,j)-alphax*hn(i+1
,j)-alphay*hn(i,j+1);
                    hn(i,j)=hn(i,j)+omega*res/cencoeff;
                    rmse=rmse+(omega*res/cencoeff).^2;
                end
            end
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
                end
            end
            //Specified RBC
            if(i == mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
```

Then like our previous case, left boundary, right boundary, all are at hn level. This should be clear that in case of our interior case also whatever residual we calculating, this is right hand side minus left hand side. This right hand side obviously that is a specified value or previous time level value and this is present time value. All are at hn level.

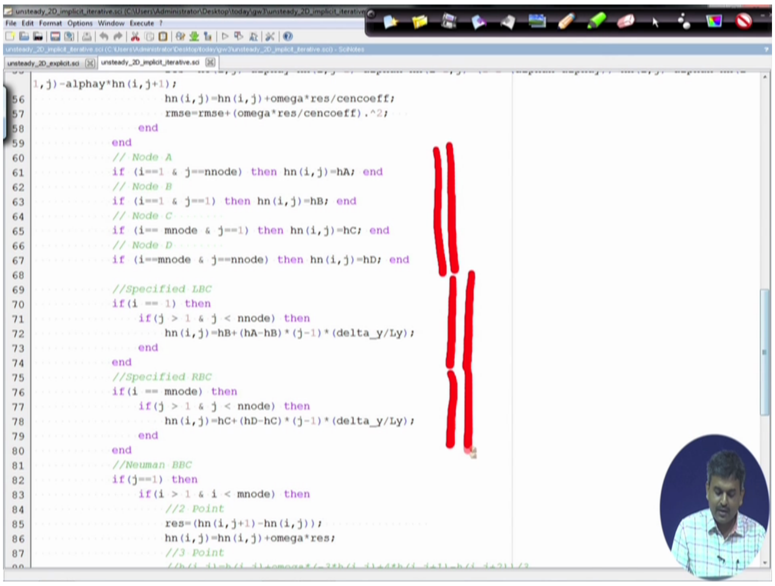
(Refer Slide Time: 30:42)



```
end

//Time Loop
t=0;
while t < Time_max
    t=t+delta_t;

count = 0;
rmse=1;
//Space Loop
while rmse > eps_max
    rmse=0
    for j=1:nnode
        for i=1:mnode
            if(i >1 & i < mnode) then
                if(j > 1 & j < nnode) then
                    cencoeff=-(1+2*(alphax+alphay));
                    res=-ho(i,j)-alphay*hn(i,j-1)-alphax*hn(i-1,j))+(1+2*(alphax+alphay))*hn(i,j)-alphax*hn(i+
1,j)-alphay*hn(i,j+1);
                    hn(i,j)=hn(i,j)+omega*res/cencoeff;
                    rmse=rmse+(omega*res/cencoeff).^2;
                end
            end
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
```

So we can directly get the rmse from here. Now in this case we can specify A, B, C, D values. Then specified left boundary, specified right boundary condition.

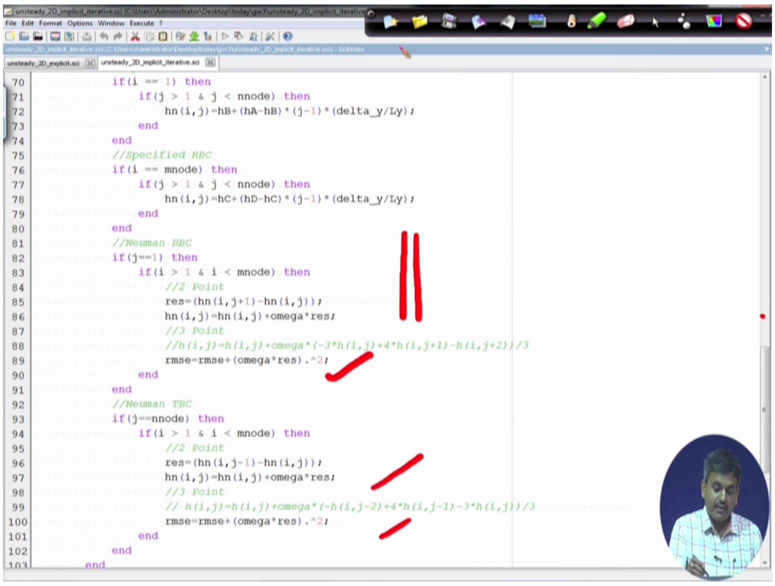
(Refer Slide Time: 31:03)



```
1,j)-alphay*hn(i,j+1);
                    hn(i,j)=hn(i,j)+omega*res/cencoeff;
                    rmse=rmse+(omega*res/cencoeff).^2;
                end
            end
            // Node A
            if (i==1 & j==nnode) then hn(i,j)=hA; end
            // Node B
            if (i==1 & j==1) then hn(i,j)=hB; end
            // Node C
            if (i== mnode & j==1) then hn(i,j)=hC; end
            // Node D
            if (i==mnode & j==nnode) then hn(i,j)=hD; end

            //Specified LBC
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
                end
            end
            //Specified RBC
            if(i == mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
                end
            end
            //Neuman BBC
            if(j==1) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j+1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
```

And Neumann boundary condition. In implementation of Neumann boundary condition I have used this two point thing. Two point thing means this is first order thing. Again I can calculate this residual and get the values here.

(Refer Slide Time: 31:29)

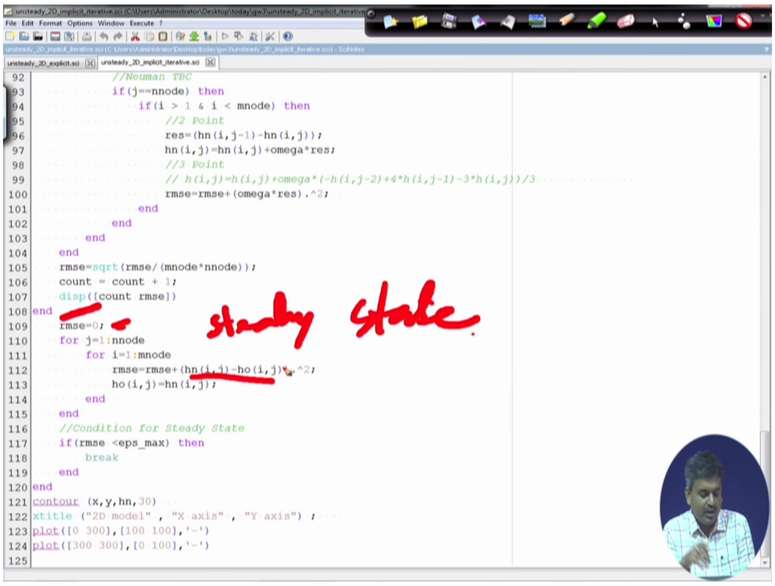


unsteady_2D_implicit_iterative.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_implicit_iterative
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci
unsteady_2D_implicit_iterative.sci

```
            if(i == 1) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hB+(hA-hB)*(j-1)*(delta_y/Ly);
                end
            end
            //Specified RBC
            if(i == mnode) then
                if(j > 1 & j < nnode) then
                    hn(i,j)=hC+(hD-hC)*(j-1)*(delta_y/Ly);
                end
            end
            //Neuman BBC
            if(j==1) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j+1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
                    //h(i,j)=h(i,j)+omega*(-3*h(i,j)+4*h(i,j+1)-h(i,j+2))/3
                    rmse=rmse+(omega*res).^2;
                end
            end
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j-1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
                    // h(i,j)=h(i,j)+omega*(-h(i,j-2)+4*h(i,j-1)-3*h(i,j))/3
                    rmse=rmse+(omega*res).^2;
                end
            end
        end
```

Now this is the end of my loop. Now I need to calculate this rmse. Based on rmse this space loop will be executed because if rmse is greater than epsilon maximum obviously we need iterations. So after convergence from this space loop again we need to calculate this rmse. But this rmse is related to steady state condition. So whether your steady state value and your previous and present time values are converging or not.

(Refer Slide Time: 32:29)



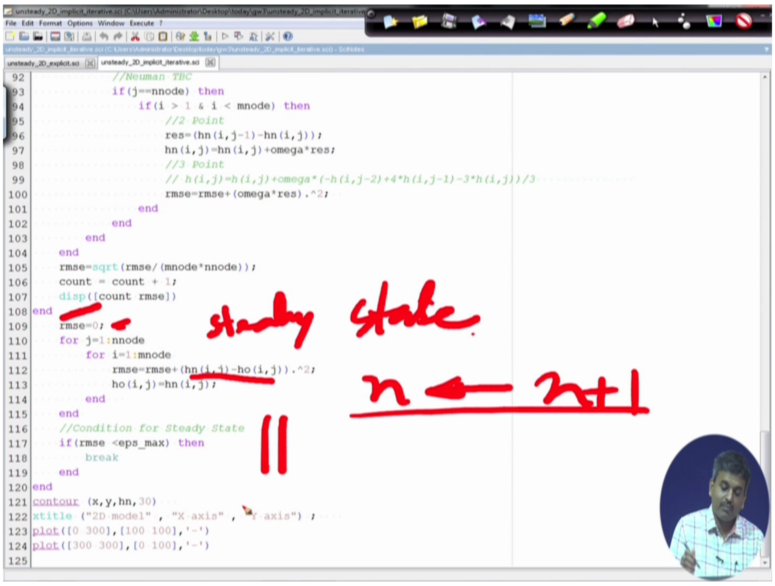
unsteady_2D_implicit_iterative.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_implicit_iterative
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci
unsteady_2D_implicit_iterative.sci

```
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j-1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
                    // h(i,j)=h(i,j)+omega*(-h(i,j-2)+4*h(i,j-1)-3*h(i,j))/3
                    rmse=rmse+(omega*res).^2;
                end
            end
        end
    end
    rmse=sqrt(rmse/(mnode*nnode));
    count = count + 1;
    disp([count rmse])
end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
    end
    //Condition for Steady State
    if(rmse <eps_max) then
        break
    end
end
contour (x,y,hn,30)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

steady state

Now we need to specify this thing, that future time level value will be transferred here. So this is the essential part of your time loop. Now again we can use this steady state condition to get the solution. That means we can check whether rmse is less than 10 to the power minus 5 here.

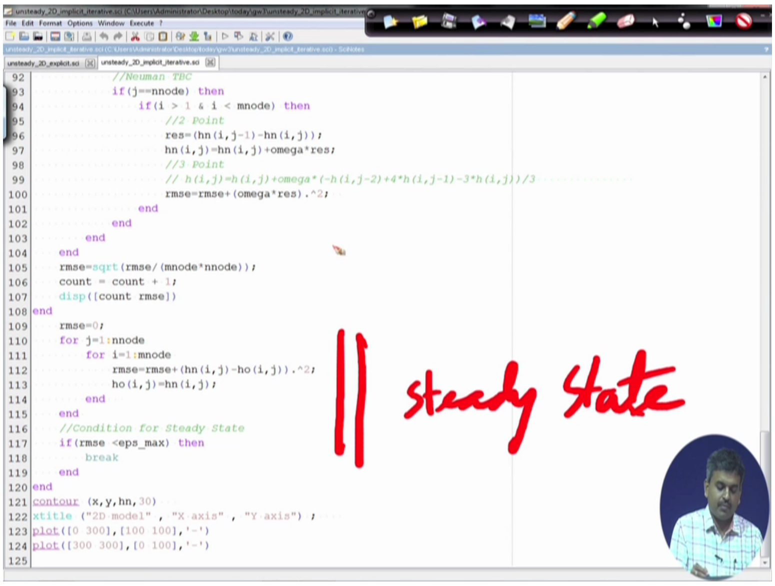
(Refer Slide Time: 32:51)


unsteady_2D_implicit_iterative.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_implicit_iterative
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci
unsteady_2D_implicit_iterative.sci
steady state
n ← n+1
||

```
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j-1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
                    // h(i,j)=h(i,j)+omega*(-h(i,j-2)+4*h(i,j-1)-3*h(i,j))/3
                    rmse=rmse+(omega*res).^2;
                end
            end
        end
    end
    rmse=sqrt(rmse/(mnode*nnode));
    count = count + 1;
    disp([count rmse])
end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
    end
    //Condition for Steady State
    if(rmse <eps_max) then
        break
    end
end
contour (x,y,hn,30)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

Both the rmse I have considered as this upper limit is same. So it should be clear that this part, rmse is for steady state condition. If it is a purely time dependent problem we should avoid or we should delete this rmse related lines from your time loop. But if you are considering steady state problem we need to include this.

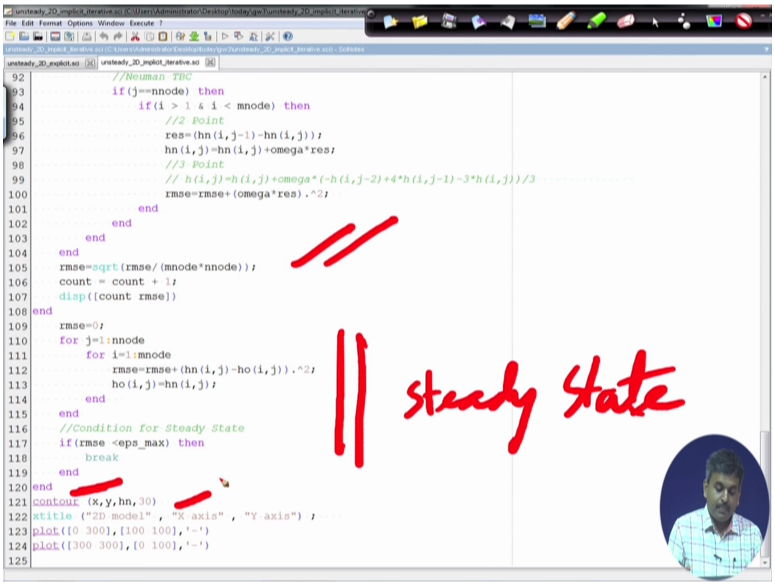
(Refer Slide Time: 33:26)


unsteady_2D_implicit_iterative.sci (C:\Users\Administrator\Desktop\today\gw3\unsteady_2D_implicit_iterative
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci
unsteady_2D_implicit_iterative.sci
|| steady state

```
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j-1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
                    // h(i,j)=h(i,j)+omega*(-h(i,j-2)+4*h(i,j-1)-3*h(i,j))/3
                    rmse=rmse+(omega*res).^2;
                end
            end
        end
    end
    rmse=sqrt(rmse/(mnode*nnode));
    count = count + 1;
    disp([count rmse])
end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
    end
    //Condition for Steady State
    if(rmse <eps_max) then
        break
    end
end
contour (x,y,hn,30)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

But this rmse is essential because you need convergence for your (sa) space loop. So your time loop ends here. Now we can plot our xy based on hn values.

(Refer Slide Time: 33:42)

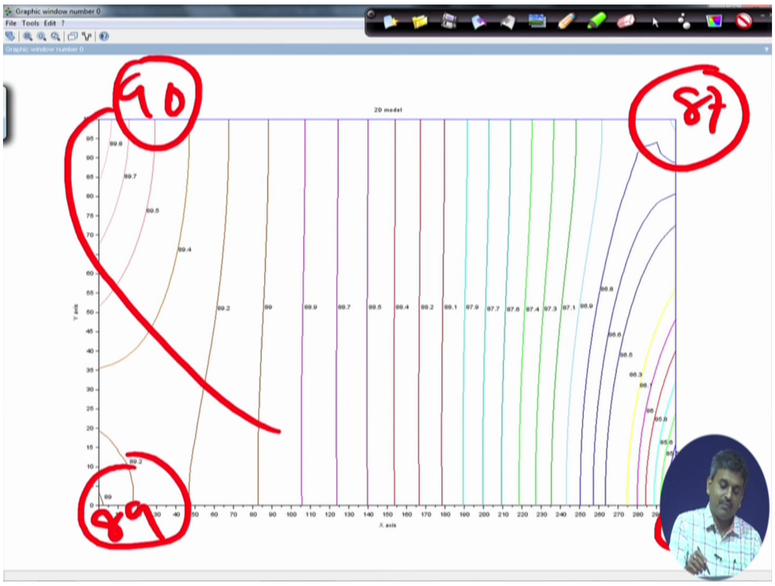


```
            //Neuman TBC
            if(j==nnode) then
                if(i > 1 & i < mnode) then
                    //2 Point
                    res=(hn(i,j-1)-hn(i,j));
                    hn(i,j)=hn(i,j)+omega*res;
                    //3 Point
                    // h(i,j)=h(i,j)+omega*(-h(i,j-2)+4*h(i,j-1)-3*h(i,j))/3
                    rmse=rmse+(omega*res).^2;
                end
            end
        end
    end
    rmse=sqrt(rmse/(mnode*nnode));
    count = count + 1;
    disp([count rmse])
end
    rmse=0;
    for j=1:nnode
        for i=1:mnode
            rmse=rmse+(hn(i,j)-ho(i,j)).^2;
            ho(i,j)=hn(i,j);
        end
    end
    //Condition for Steady State
    if(rmse <eps_max) then
        break
    end
end
contour (x,y,hn,30)
xtitle ("2D model" , "X axis" , "Y axis") ;
plot([0 300],[100 100],'-')
plot([300 300],[0 100],'-')
```

unsteady_2D_implicit_iterative.sci
File Edit Format Options Window Execute ?
unsteady_2D_explicit.sci
unsteady_2D_implicit_iterative.sci
|| steady state

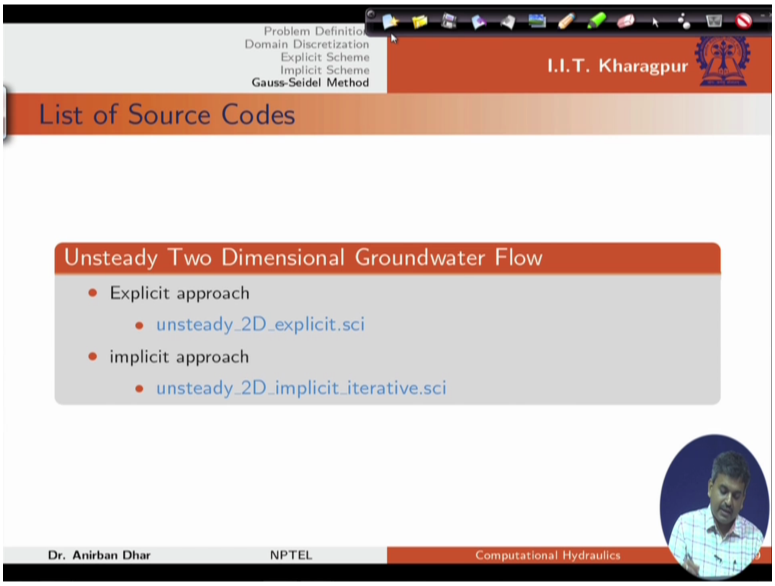
So if you run it and select it and evaluate it, so some number of iterations will be required for this case. Again interesting part is that we are quickly getting convergence. Because del t criteria is not there so our time step is larger compared to that used in our explicit case. So we are getting solution which is similar to our steady state case. You can see the variation here. 90, 89, this 85, obviously this implicit scheme is much better compared to our explicit scheme and that we can see from the solution approach.

(Refer Slide Time: 34:47)



And our algorithm also, this is much faster in this case. Because you don't need that time restriction or your time discretization restriction for your implicit case. So in this particular lecture I have covered unsteady two dimensional flow. And explicit algorithm I have covered in this code is unsteady 2D explicit and this unsteady 2D implicit iterative. So you can use these codes to get different solutions either for steady state or unsteady state problems.

(Refer Slide Time: 36:06)



List of Source Codes

Unsteady Two Dimensional Groundwater Flow

- Explicit approach
  - unsteady_2D_explicit.sci
- implicit approach
  - unsteady_2D_implicit_iterative.sci

Next lecture class I will be covering the finite volume solution approach for this groundwater flow problems. Thank you.