

Computational Hydraulics
Professor Anirban Dhar
Department of Civil Engineering
Indian Institute of Technology Kharagpur
Lecture 38
Gradually Varied Flow: Implicit Approach

Welcome to this lecture class of the course computational hydraulics. Where are in model 4, surface water hydraulics. And this is unit number 2, gradually varied flow, implicit approach.

(Refer Slide Time: 00:35)

The image shows a presentation slide with a white background and a dark blue header. The header contains a navigation menu on the left with items: Problem Definition, General Structure, Backward Euler Method, Implicit Runge-Kutta Methods, and References. On the right of the header, it says 'I.I.T. Kharagpur' next to the institute's logo. The main content area features a dark blue rounded rectangle with the text 'Module 04: Surface Water Hydraulics' and 'Unit 02: Gradually Varied Flow-Implicit Approach' in white. Below this, the name 'Anirban Dhar' is centered, followed by his affiliation: 'Department of Civil Engineering, Indian Institute of Technology Kharagpur, Kharagpur' and 'National Programme for Technology Enhanced Learning (NPTEL)'. At the bottom, there is a dark blue footer with 'Dr. Anirban Dhar' on the left, 'NPTEL' in the center, and 'Computational Hydraulics 1 / 21' on the right.

In our previous lecture class we have discussed about gradually varied flow using explicit approach. Now learning objective of this particular lecture class. At the end of this lecture class students will be able to solve gradually varied flow problem for open channels using implicit method.

(Refer Slide Time: 01:08)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Learning Objective

- To solve gradually varied flow problem for open channels using implicit methods.

Dr. Anirban Dhar NPTEL Computational Hydraulics 2 / 21

Now this gradually varied flow we are considering for second channels. Problem definition to solution, this is our general framework. In our previous lecture class we have got solution which is y as a function of x from the governing equation dy by dx equals to S not minus S_f 1 minus Fr square. And this is initial value problem.

So from our definition we do not need boundary condition in this case or initial condition, one sided condition is there. So this is our first order ODE differential equation.

(Refer Slide Time: 02:18)

Problem Definition to Solution

$y(x)$ $\frac{dy}{dx} = \frac{S_0 - S_f}{1 - Fr^2}$ IVP

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

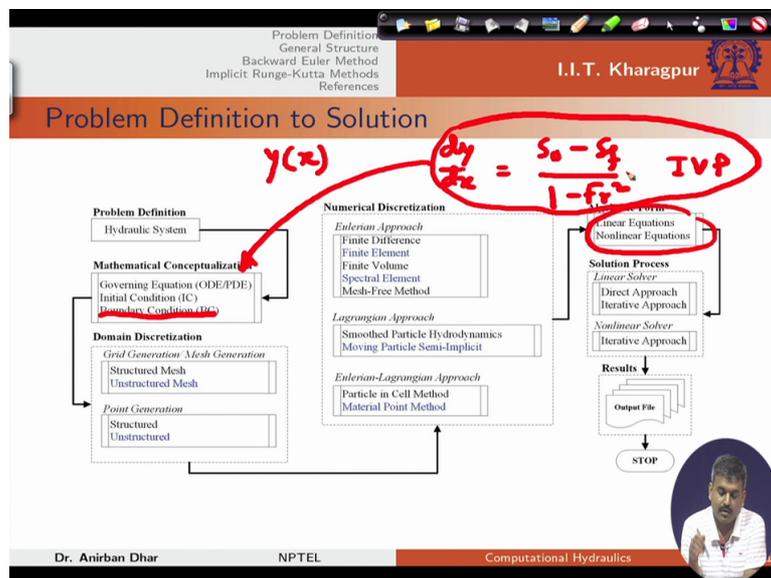
Results
Output File
STOP

Dr. Anirban Dhar NPTEL Computational Hydraulics 3 / 21

Now to solve this with the structure of explicit method we can directly transfer all the known quantities on the right hand side and only unknown quantities on the left hand side. So directly we can get the solution using explicit approach. However if we utilise the implicit approach for the solution of GVF or gradually varied flow, we need to solve it either using this linear equations or nonlinear equations as per our requirement.

But in this case the term SF which is a function of y and Fr which is again function of y, these are non linear in nature. Obviously we will be getting non linear equations. So we need the solution approach for those governing equations or discretized governing equations.

(Refer Slide Time: 03:33)



Now let us consider our basic problem statement. So problem definition is governing equation for gradually varied flow in prismatic channels can be written as $\frac{dy}{dx} = \frac{S_0 - S_f}{1 - Fr^2}$. With $S_f = \frac{n^2 Q^2}{148.6 A^2 R^{4/3}}$. And in this case n is Mannings roughness, Q is discharge. Obviously for a channel maybe it is a specified discharge.

R to the power $4/3$, R is hydraulic radius, A is area, T is top width, Q is again discharge, g is in this case this is acceleration due to gravity, A is area and we have this initial condition on one side. So these are the expressions that we can directly utilise for our problem.

(Refer Slide Time: 05:00)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Problem Definition

Governing Equation for Gradually Varied Flow in prismatic channel can be written as,

Initial Value Problem

$$\frac{dy}{dx} = \Psi(x, y) \quad \text{with} \quad \Psi(x, y) = \frac{S_0 - S_f}{1 - Fr^2} = \frac{S_0 - \frac{n^2 Q^2}{R^{4/3} A^2}}{1 - \frac{Q^2 T}{g A^3}}$$

Initial Condition: $y|_{x=0} = y_0$

where

y = depth of flow	x = coordinate direction
S_f = friction slope $(= \frac{n^2 Q^2}{R^{4/3} A^2})$	Fr = Froude number $(= \sqrt{\frac{Q^2 T}{g A^3}})$
S_0 = bed slope	Q = discharge
T = top width	g = acceleration due to gravity
R = hydraulic radius	A = cross-sectional area

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now what is given and what is required for our problem solution? Given, let us say that we have a given channel cross section type. Channel cross section type is rectangular. Then y_0 which is the initial value at x equals to zero, this is point 8 metres. Then B which is the bottom width of rectangular channel, this is 15 metres. g is 9.81 metres per second square. Then S_0 is 0.0008. n is point not 15. L_x , length of the domain or length of the problem which we need to consider.

L_x is 200 metres and Q is 20 metres cube per second. Now in this case required is that we need to identify the type of GVF profile from the problem and plot of the GVF profile.

(Refer Slide Time: 06:19)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Problem Statement

Given

Channel Cross-Section Type: **Rectangular**

$y_0 = 0.8m$
 $B = 15m$
 $g = 9.81m/s^2$
 $S_0 = 0.0008$
 $n = 0.015$
 $L_x = 200m$
 $Q = 20m^3/s$

Required

Identify the type of GVF Profile: ?
 Plot of the GVF Profile.

Dr. Anirban Dhar NPTEL Computational Hydraulics

So this is for rectangular cross section. In our previous lecture class we have seen that we can have two different slopes for a trapezoidal cross section. So this is general slope feature. If this m equals to zero m1 and m2 then we can get rectangular cross section in that case.

(Refer Slide Time: 06:59)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Rectangular Cross-section

$A = By$
 $P = B + 2y$
 $R = \frac{A}{P}$
 $T = B$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So in this case we have top width equals to bottom width or channel width and y. So this area is B into y, P is B plus 2y considering depth on both sides. So this is the weighted perimeter for the channel.

(Refer Slide Time: 07:25)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Rectangular Cross-section

$A = By$
 $P = B + 2y$
 $R = \frac{A}{P}$
 $T = B$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now R which is hydraulic radius is A by P. This is By by B plus 2y. Now in this case R is again function of y. For given B your flow depth can vary. So in this case R is function of y.

(Refer Slide Time: 07:54)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Rectangular Cross-section

$A = By$
 $P = B + 2y$
 $R = \frac{A}{P} = \frac{By}{B+2y}$
 $T = B$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So in this case we can find out the critical depth to define the GVF or gradually varied flow profiles. We need the level for NDL which is normal depth line and critical depth line. Critical depth line is corresponding to this Fr equals to 1 and normal depth line we can find out from Mannings equation.

(Refer Slide Time: 08:35)

Problem Definition
Critical Depth

I.I.T. Kharagpur

For critical depth, $Fr = 1$

$$Fr = \sqrt{\frac{Q^2 T}{g A^3}} = 1$$

In case of rectangular channel, $A = By$ and $T = B$

$$\sqrt{\frac{Q^2 T}{g A^3}} = 1$$

$$y_c = \left(\frac{Q^2}{g B^2}\right)^{\frac{1}{3}}$$

GVF
NDL
CDL Fr=1

Dr. Anirban Dhar NPTEL Computational Hydraulics

So for the first case if Fr equals to 1 then $Q^2 T$ by gA^3 this is 1 and from there we can get this QC which is $Q^2 gB^3$ to the power 1/3rd.

(Refer Slide Time: 08:56)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Problem Definition

Critical Depth

For critical depth, $Fr = 1$

$$Fr = \sqrt{\frac{Q^2 T}{g A^3}} = 1$$

In case of rectangular channel, $A = By$ and $T = B$

$$\sqrt{\frac{Q^2 T}{g A^3}} = 1$$

$$y_c = \left(\frac{Q^2}{g B^2}\right)^{\frac{1}{3}}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now for rectangular channels if we need this normal depth we can use Mannings equation. In this case please remember that we are using S not. And rectangular channel case if we represent y_n as normal depth so this is area and this is perimeter.

(Refer Slide Time: 09:27)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Problem Definition

Normal Depth

Normal depth can be calculated from Manning's equation (uniform flow),

$$Q = \frac{1}{n} R^{\frac{2}{3}} S_0^{\frac{1}{2}} A$$

In case of rectangular channel, $A = By_n$ and $P = B + 2y_n$

$$Q = \frac{1}{n} \left(\frac{By_n}{B + 2y_n}\right)^{\frac{2}{3}} S_0^{\frac{1}{2}} B y_n$$

In function form,

$$G(y_n) = S_0^{\frac{1}{2}} B^{\frac{5}{3}} \left(\frac{y_n}{B + 2y_n}\right)^{\frac{2}{3}} y_n - Q = 0$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now for a given discharge slope and n value we can find out unit value of y_n . So that is the normal depth in this case.

(Refer Slide Time: 09:48)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Problem Definition

Normal Depth

Normal depth can be calculated from Manning's equation (uniform flow),

$$Q = \frac{1}{n} R^{2/3} S_0^{1/2} A$$

In case of rectangular channel, $A = By_n$ and $P = B + 2y_n$

$$Q = \frac{1}{n} \left(\frac{By_n}{B + 2y_n} \right)^{2/3} S_0^{1/2} By_n$$

In function form,

$$G(y_n) = S_0^{1/2} B^{5/3} \left(\frac{y_n}{B + 2y_n} \right)^{2/3} y_n - Q = 0$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So we can write this as general function G y_n which is nothing but this quantity minus the quantity on left hand side and we can equate it with zero.

(Refer Slide Time: 10:08)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Problem Definition

Normal Depth

Normal depth can be calculated from Manning's equation (uniform flow),

$$Q = \frac{1}{n} R^{2/3} S_0^{1/2} A$$

In case of rectangular channel, $A = By_n$ and $P = B + 2y_n$

$$Q = \frac{1}{n} \left(\frac{By_n}{B + 2y_n} \right)^{2/3} S_0^{1/2} By_n$$

In function form,

$$G(y_n) = S_0^{1/2} B^{5/3} \left(\frac{y_n}{B + 2y_n} \right)^{2/3} y_n - Q = 0$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now this is one non linear equation. Now if you want to solve this problem we need to use our Newton Raphson technique and from our Newton Raphson technique we can get the solution by iterative approach. This y_n P, P is iteration count in this case. So y_n P minus 1, this is coming from previous iteration. So we can start with y_n zero and maybe we can start from y_c value because we have already got the y_c.

If we have this y_c value or CDL location then either this NDL on the upper side or the lower side. So it is safe if we start from this y_c which is the guess value for the normal depth iteration process.

(Refer Slide Time: 11:30)

Problem Definition
Normal Depth

I.I.T. Kharagpur

From Newton-Raphson method,

$$y_n^{(p)} = y_n^{(p-1)} - \frac{G(y_n^{(p-1)})}{G'(y_n^{(p-1)})}$$

where

$$G'(y_n) = \frac{S_0^{1/2} B^{5/3} y_n^{2/3} (5B + 6y_n)}{3n (B + 2y_n)^{5/3}}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now in this case we need to find out what is this G prime? G prime is nothing but this is dG divided by dYn. Now this quantity we need on this denominator and this should be calculated based on the previous iteration value and this is our original function value which is capital G function. Now from this iteration process we can get this y.

(Refer Slide Time: 12:18)

Problem Definition
Normal Depth

I.I.T. Kharagpur

From Newton-Raphson method,

$$y_n^{(p)} = y_n^{(p-1)} - \frac{G(y_n^{(p-1)})}{G'(y_n^{(p-1)})}$$

where

$$\frac{dG}{dy_n} = G'(y_n) = \frac{S_0^{1/2} B^{5/3} y_n^{2/3} (5B + 6y_n)}{3n (B + 2y_n)^{5/3}}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now if we see this implicit Runge Kutta methods, in our previous (lec) lecture class we have seen that this counter which is i in case of explicit method this was up to i minus 1. So this was up to i minus 1.

(Refer Slide Time: 12:58)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^{i-1} c_{ij}^y K_j \right)$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now in this case we are going up to i. That means whatever value of this increment K_i is there on the left hand side, we are also incorporating that K_i value during calculation of ψ_i . So in a way this increment equation is implicit in nature and we have different weights W_j , K_j . J is running from 1 to m , m is the number of weights or number of points or number of increments that are required for calculation of this weight.

(Refer Slide Time: 13:41)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^{i-1} c_{ij}^y K_j \right)$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now in this case we have C_i or increment for x direction and increment for our y . So C_x this i is increment corresponding to i th increment parameter. And this C_{ij} y , this is general increment parameter for y_m .

(Refer Slide Time: 14:20)

The slide is titled "Implicit Runge-Kutta Methods" and is part of a presentation from I.I.T. Kharagpur. It includes a navigation menu at the top with options like "Problem Definition", "General Structure", "Backward Euler Method", "Implicit Runge-Kutta Methods", and "References". The main text states: "The Runge-Kutta method is defined as weighted assembly of increments by," followed by the equation
$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$
 and "with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right)$ ". Handwritten red annotations highlight c_i^x and c_{ij}^y in the equations. A small circular portrait of Dr. Anirban Dhar is visible in the bottom right corner of the slide.

Now in this case if we consider this Butcher Tableau, this can be expressed in general terms with this format. So we have let us say m terms or m increments. Now for each increment we need increment in case of x . So this is corresponding to increment for x or x_n and on the right hand side, this is for x . On this side we have increment for y and the bottom portion we have different weights. So if we consider the portion which is for y that is one matrix and size is m by m matrix.

(Refer Slide Time: 15:35)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right)$

Complete Butcher Tableau (Butcher, 2008) can be expressed as

c_1^x	$c_{1,1}^y$	$c_{1,2}^y$...	$c_{1,m-1}^y$	$c_{1,m}^y$
c_2^x	$c_{2,1}^y$	$c_{2,2}^y$...	$c_{2,m-1}^y$	$c_{2,m}^y$
...
c_{m-1}^x	$c_{m-1,1}^y$	$c_{m-1,2}^y$...	$c_{m-1,m-1}^y$	$c_{m-1,m}^y$
c_m^x	$c_{m,1}^y$	$c_{m,2}^y$...	$c_{m,m-1}^y$	$c_{m,m}^y$
	W_1	W_2	...	W_{m-1}	W_m

Dr. Anirban Dhar NPTEL Computational Hydraulics 10 / 21

So each row is representing the coefficient of the corresponding increments that are required for the calculation of flow depth y . And the first term in this row, this is corresponding to increment for x . So this is corresponding to x .

(Refer Slide Time: 16:06)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right)$

Complete Butcher Tableau (Butcher, 2008) can be expressed as

c_1^x	$c_{1,1}^y$	$c_{1,2}^y$...	$c_{1,m-1}^y$	$c_{1,m}^y$
c_2^x	$c_{2,1}^y$	$c_{2,2}^y$...	$c_{2,m-1}^y$	$c_{2,m}^y$
...
c_{m-1}^x	$c_{m-1,1}^y$	$c_{m-1,2}^y$...	$c_{m-1,m-1}^y$	$c_{m-1,m}^y$
c_m^x	$c_{m,1}^y$	$c_{m,2}^y$...	$c_{m,m-1}^y$	$c_{m,m}^y$
	W_1	W_2	...	W_{m-1}	W_m

Dr. Anirban Dhar NPTEL Computational Hydraulics 10 / 21

And these values are corresponding to different coefficients that are required for the calculation for coefficient here.

(Refer Slide Time: 16:19)

Problem Definition
 General Structure
 Backward Euler Method
 Implicit Runge-Kutta Methods
 References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right)$

Complete Butcher Table (Butcher, 2008) can be expressed as

c_1^x	$c_{1,1}^y$	$c_{1,2}^y$...	$c_{1,m-1}^y$	$c_{1,m}^y$
c_2^x	$c_{2,1}^y$	$c_{2,2}^y$...	$c_{2,m-1}^y$	$c_{2,m}^y$
...
c_{m-1}^x	$c_{m-1,1}^y$	$c_{m-1,2}^y$...	$c_{m-1,m-1}^y$	$c_{m-1,m}^y$
c_m^x	$c_{m,1}^y$	$c_{m,2}^y$...	$c_{m,m-1}^y$	$c_{m,m}^y$
	W_1	W_2	...	W_{m-1}	W_m

Dr. Anirban Dhar NPTEL Computational Hydraulics 10 / 21

And these weights are related to the final calculation.

(Refer Slide Time: 16:27)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right)$

Complete Butcher Tableau (Butcher, 2008) can be expressed as

c_1^x	$c_{1,1}^y$	$c_{1,2}^y$...	$c_{1,m-1}^y$	$c_{1,m}^y$
c_2^x	$c_{2,1}^y$	$c_{2,2}^y$...	$c_{2,m-1}^y$	$c_{2,m}^y$
...
c_{m-1}^x	$c_{m-1,1}^y$	$c_{m-1,2}^y$...	$c_{m-1,m-1}^y$	$c_{m-1,m}^y$
c_m^x	$c_{m,1}^y$	$c_{m,2}^y$...	$c_{m,m-1}^y$	$c_{m,m}^y$
	W_1	W_2	...	W_{m-1}	W_m

Dr. Anirban Dhar NPTEL Computational Hydraulics 10 / 21

Now if we express it in terms of reduced matrix form, we can write this as column vector. First one is one column vector and this is a row vector. So in general we can express this as column vector and we can take this transpose here and capital Cy, this is the a full matrix here. So this is the general representation of using Butcher Tableau.

(Refer Slide Time: 17:10)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta Methods

The Runge-Kutta method is defined as weighted assembly of increments by,

$$y_{n+1} = y_n + \sum_{j=1}^m W_j K_j$$

with $K_i = \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right)$

Complete Butcher Tableau (Butcher, 2008) can be expressed as

c_1^x	$c_{1,1}^y$	$c_{1,2}^y$...	$c_{1,m-1}^y$	$c_{1,m}^y$
c_2^x	$c_{2,1}^y$	$c_{2,2}^y$...	$c_{2,m-1}^y$	$c_{2,m}^y$
...
c_{m-1}^x	$c_{m-1,1}^y$	$c_{m-1,2}^y$...	$c_{m-1,m-1}^y$	$c_{m-1,m}^y$
c_m^x	$c_{m,1}^y$	$c_{m,2}^y$...	$c_{m,m-1}^y$	$c_{m,m}^y$
	W_1	W_2	...	W_{m-1}	W_m

In reduced matrix form

$$\begin{pmatrix} c^x \\ C^y \\ W^T \end{pmatrix}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 10 / 21

Now using this tableau we can represent our Runge Kutta methods. And if we start from our well known basic approach which is backward Euler. In backward Euler let us say that this is Cx 1 and this is Cy 1-1. That means only one term is there. And on the bottom we have the W1 which is again equal to 1. So what will be the structure of our increment? Increment will

be K_1 equals to $\Delta x \psi$, in this case x_n plus, this is one so directly we can add that Δx comma we should have y_n .

Again the coefficient is 1, so in this case we can write it as K_1 and this is our case and which is a coefficient of this K_1 . And finally we can add this y_n equals to y_n plus W_1 which is again K_1 in this case because W_1 equals to 1. So from this process we have identified the backward Euler method.

(Refer Slide Time: 19:23)

The slide is titled "Backward Euler Method" and includes a navigation menu with options like "Problem Definition", "General Structure", "Backward Euler Method", "Implicit Runge-Kutta Methods", and "References". The I.I.T. Kharagpur logo is visible in the top right corner.

The main content of the slide is:

Considering Butcher Tableau as

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

$$C_1^T = 1 \quad C_1^Y = 1$$

$$N_1 = 1$$

$$K_1 = \Delta x \psi(x_n + \Delta x, y_n + K_1)$$

$$y_{n+1} = y_n + K_1$$

A small circular portrait of a man is located in the bottom right corner of the slide.

So in general terms if we write this is x_n plus 1, y_n plus 1. This is y_n plus 1 but y_n plus 1 is unknown thing. So we cannot solve it using our usual explicit approach. We need to solve this equation here.

(Refer Slide Time: 19:51)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Backward Euler Method

Considering Butcher Tableau as

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Backward Euler Method

$$y_{n+1} = y_n + \Delta x \Psi(x_{n+1}, y_{n+1})$$

Order of Backward Euler method: $\mathcal{O}(\Delta x)$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now in this case again we can write a general function because y_{n+1} is unknown so I can transfer this y_{n+1} and other terms on the left hand side. So y_{n+1} , this is coming from here. This is my second term and the third term which is unknown quantity. So this part is unknown and this is known quantity in this case.

(Refer Slide Time: 20:38)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Backward Euler Method

Considering Butcher Tableau as

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Backward Euler Method

$$y_{n+1} = y_n + \Delta x \Psi(x_{n+1}, y_{n+1})$$

Order of Backward Euler method: $\mathcal{O}(\Delta x)$

In function form,

$$F(y_{n+1}) = y_{n+1} - \Delta x \Psi(x_{n+1}, y_{n+1}) - y_n = 0$$

Unknown Known

Dr. Anirban Dhar NPTEL Computational Hydraulics

So again this is non linear equation and we can solve it to get this y_{n+1} . Now in this process if we write this general equation which is $y_{n+1} = P$, $y_{n+1} = P - 1$ which is again value which is coming from previous iteration. And these are the function values and derivative of the function value.

(Refer Slide Time: 21:16)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Backward Euler Method

From Newton-Raphson method,

$$y_{n+1}^{(p)} = y_{n+1}^{(p-1)} - \frac{F(y_{n+1}^{(p-1)})}{F'(y_{n+1}^{(p-1)})}$$

where

$$F'(y_{n+1}) = 1 - \Delta x \left[\left(1 - \frac{Q^2}{B^2 g y_{n+1}^3}\right)^{-1} \left[\left(\frac{2n^2 Q^2}{B^2 y_{n+1}^3}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{4}{3}} + \left(\frac{4n^2 Q^2}{3B^2 y_{n+1}^2}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{7}{3}} \left(\frac{B}{B + 2y_{n+1}} - \frac{2B y_{n+1}}{(B + 2y_{n+1})^2}\right) \right] - \left(\frac{3Q^2}{B^2 g y_{n+1}^4}\right) \left(1 - \frac{Q^2}{B^2 g y_{n+1}^3}\right)^{-2} \left[S_0 - \left(\frac{n^2 Q^2}{B^2 y_{n+1}^2}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{4}{3}} \right] \right]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So initial was $y_{n+1} - \Delta x \psi(x_{n+1}, y_{n+1})$. Obviously if we differentiate this one with respect to y_{n+1} which is the unknown quantity here, this is y_{n+1} and minus y_n . So initially this quantity is not significant here because if you differentiate it with respect to y_{n+1} this will be zero, this is 1. From the second term we are getting this big expression for rectangular channels.

(Refer Slide Time: 22:25)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Backward Euler Method

From Newton-Raphson method,

$$y_{n+1}^{(p)} = y_{n+1}^{(p-1)} - \frac{F(y_{n+1}^{(p-1)})}{F'(y_{n+1}^{(p-1)})}$$

where

$\frac{dF}{dy_{n+1}} = \frac{d}{dy_{n+1}} [y_{n+1} - \Delta x \psi(x_{n+1}, y_{n+1}) - y_n]$

$$F'(y_{n+1}) = 1 - \Delta x \left[\left(1 - \frac{Q^2}{B^2 g y_{n+1}^3}\right)^{-1} \left[\left(\frac{2n^2 Q^2}{B^2 y_{n+1}^3}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{4}{3}} + \left(\frac{4n^2 Q^2}{3B^2 y_{n+1}^2}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{7}{3}} \left(\frac{B}{B + 2y_{n+1}} - \frac{2B y_{n+1}}{(B + 2y_{n+1})^2}\right) \right] - \left(\frac{3Q^2}{B^2 g y_{n+1}^4}\right) \left(1 - \frac{Q^2}{B^2 g y_{n+1}^3}\right)^{-2} \left[S_0 - \left(\frac{n^2 Q^2}{B^2 y_{n+1}^2}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{4}{3}} \right] \right]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 12 / 21

Now if we write the code for this one we can get the solution out of this. So in this case if we start this scilab so we have GVF implicit. In GVF implicit, backward Euler. Now given data from our problems statement Q is 20 metre cube per second, S not is point 0008, n is point

015, bed width for rectangular channel that is 15 metres, G is 9 point 81, Lx is 200, y is point 8 metres.

(Refer Slide Time: 23:38)

```

1 clear
2 clc
3 // Given Data
4 Q=20; //m^3/s
5 S0=0.0008;
6 n=0.015;
7 B=15; //m
8 g=9.81 //m/s^2
9 Lx=200; //m
10 y0=0.8; //m
11 mnode=201;
12 global('Q','S0','n','B','g')
13 -----
14 //Critical Depth Calculation
15 yc=(Q^2/(g*B^2))^(1/3);
16 disp(yc)
17
18
19 //Normal Depth Calculation
20 function Gval=Gfunc(y)
21     Gval=(S0^(1/2)*B^(5/3)/n)*(y^(5/3)/(B+2*y)^(2/3))-Q
22 endfunction
23
24 function Gpval=Gderi(y)
25     Gpval=(S0^(1/2)*B^(5/3)/(3*n))*(y^(2/3)/(B+2*y)^(5/3))*(5*B+6*y)
26 endfunction
27
28 //Newton-Raphson
29 eps_max=1e-6;
30 aerror=1;
31 yn=y0;
32 while abs(aerror) > eps_max
33     aerror=(Gfunc(yn)/Gderi(yn));
34     yn=yn-aerror;
35 end
36 disp(yn)
37 -----
38 //Main dydx calculation
39 function dydx = psl(x,y)
40     A=y*B;
41     F_y=B+2*y;
42     R_y=A_y/P_y;
43     SF=(n^2*Q^2)/((R_y)^(4/3)*A_y^2);

```

Now with this let us say that mnode or nodes in the m or l x direction that is 201 and these values are global, Q, S not, n, B and g. In this case first step is critical depth calculation. Yc is Q square g B square and whole to the power 1 3rd. We can get the yc value. Then normal depth calculation Gval and G prime val. So G and G prime, these two values are required.

(Refer Slide Time: 24:19)

```

10 y0=0.8; //m
11 mnode=201;
12 global('Q','S0','n','B','g')
13 -----
14 //Critical Depth Calculation
15 yc=(Q^2/(g*B^2))^(1/3);
16 disp(yc)
17
18
19 //Normal Depth Calculation
20 function Gval=Gfunc(y)
21     Gval=(S0^(1/2)*B^(5/3)/n)*(y^(5/3)/(B+2*y)^(2/3))-Q
22 endfunction
23
24 function Gpval=Gderi(y)
25     Gpval=(S0^(1/2)*B^(5/3)/(3*n))*(y^(2/3)/(B+2*y)^(5/3))*(5*B+6*y)
26 endfunction
27
28 //Newton-Raphson
29 eps_max=1e-6;
30 aerror=1;
31 yn=y0;
32 while abs(aerror) > eps_max
33     aerror=(Gfunc(yn)/Gderi(yn));
34     yn=yn-aerror;
35 end
36 disp(yn)
37 -----
38 //Main dydx calculation
39 function dydx = psl(x,y)
40     A=y*B;
41     F_y=B+2*y;
42     R_y=A_y/P_y;
43     SF=(n^2*Q^2)/((R_y)^(4/3)*A_y^2);

```

So using Newton Raphson here we can get the solution and we can get this y_n . So after getting this y_n and y_c we can comment on the flow condition.

(Refer Slide Time: 24:34)

```

10 y0=0.5; %m
11 mode=201;
12 global('Q','S0','n','B','g')
13 //-----
14 //Critical Depth Calculation
15 yc=(Q^2/(g*B^2))^(1/3);
16 disp(yc)
17
18
19 //Normal Depth Calculation
20 function opval=gfunc(y)
21     opval=(S0^(1/2)*B^(5/3)/n)*(y^(5/3)/(B+2*y)^(2/3))-Q
22 endfunction
23
24 function opval=gderi(y)
25     opval=(S0^(1/2)*B^(5/3)/(3*n))*(y^(2/3)/(B+2*y)^(5/3))*(5*B+6*y)
26 endfunction
27
28 //Newton-Raphson
29 eps_max=1e-6;
30 aerror=1;
31 yn=yc;
32 while abs(aerror) > eps_max
33     aerror=(gfunc(yn)/gderi(yn));
34     yn=yn-aerror;
35 end
36 disp(yn)
37 //-----
38 //Main dydx calculation
39 function dydx = psi(x,y)
40     A_y=B*y;
41     P_y=B+2*y;
42     R_y=A_y/P_y;
43     Sf=(n^2*Q^2)/((R_y)^(4/3)*A_y^2);

```

So obviously if your y_n is greater than y_c we have mild slope condition. And if y_n is less than y_c we have steep slope condition.

(Refer Slide Time: 24:56)

```

32 while abs(aerror) > eps_max
33     aerror=(gfunc(yn)/gderi(yn));
34     yn=yn-aerror;
35 end
36 disp(yn)
37 //-----
38 //Main dydx calculation
39 function dydx = psi(x,y)
40     A_y=B*y;
41     P_y=B+2*y;
42     R_y=A_y/P_y;
43     Sf=(n^2*Q^2)/((R_y)^(4/3)*A_y^2);
44     Frs=(Q^2*B)/(g*(B*y)^3);
45     dydx=(S0-Sf)/(1-Frs);
46 endfunction
47
48 function Fval=Ffunc(y,x,delta_x,yp)
49     Fval=y-delta_x*psi(x,y)-yp
50 endfunction
51
52 function Fpval=Fderi(y,delta_x)
53     trm1=(1-Q^2/(B^2*g*y^3))^(1/3);
54     trm2=(2*n^2*Q^2)/(B^2*y^3);
55     trm3=(B*y/(B+2*y))^(4/3);
56     trm4=(4*n^2*Q^2)/(3*B^2*y^2);
57     trm5=(B*y/(B+2*y))^(7/3);
58     trm6=(B/(B+2*y))-2*B*y/(B+2*y)^2;
59     trm7=(3*Q^2)/(B^2*g*y^4);
60     trm8=(1-Q^2/(B^2*g*y^3))^(2/3);
61     trm9=S0;
62     trm10=(n^2*Q^2)/(B^2*y^2);
63     trm11=(B*y/(B+2*y))^(4/3);
64     Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
65 endfunction

```

Now first case we have dy by dx equals to $\psi(x,y)$, our general governing equation in this case.

(Refer Slide Time: 25:12)

```
backward_euler.sl (C:\Users\Administrator\Desktop\the_day\gpl\backward_euler.sl) - SciNotes
File Edit Format Options Window Execute ?
backward_euler.sl (C:\Users\Administrator\Desktop\the_day\gpl\backward_euler.sl) - SciNotes
backward_euler.sl [
32 while ~err (while) ~ eps_max
33   aerror=(Gfunc(yn)/Gderi(yn));
34   yn=yn-aerror;
35 end
36 disp(yn)
37 -----
38 //Main dydx calculation
39 1 function dydx = psi(x,y)
40 2 A_y=B*y;
41 3 P_y=B+2*y;
42 4 R_y=A_y/P_y;
43 5 Sf=(n^2*Q^2)/(R_y)^(4/3)*A_y^2;
44 6 Frs=(Q^2*B)/(g*(B*y)^3);
45 7 dydx=(S0-Sf)/(1-Frs);
46 endfunction
47
48 1 function Fval=Ffunc(y,x,delta_x,yp)
49 2   Fval-y-delta_x*psi(x,y)-yp
50 endfunction
51
52 1 function Fpval=Fderi(y,delta_x)
53 2   trm1=(1-Q^2/(B^2*g*y^3))^(-1);
54 3   trm2=(2*n^2*Q^2)/(B^2*y^3);
55 4   trm3=(B*y/(B+2*y))^(4/3);
56 5   trm4=(4*n^2*Q^2)/(3*B^2*y^2);
57 6   trm5=(B*y/(B+2*y))^(7/3);
58 7   trm6=((B/(B+2*y))-(2*B*y/(B+2*y)^2));
59 8   trm7=(3*Q^2)/(B^2*g*y^4);
60 9   trm8=(1-Q^2/(B^2*g*y^3))^(-2);
61 10  trm9=S0;
62 11  trm10=(n^2*Q^2)/(B^2*y^2);
63 12  trm11=(B*y/(B+2*y))^(4/3);
64 13  Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
65 endfunction
```

$\frac{dy}{dx} = \psi(x,y)$



So this is By, B plus 2y, R is Ay divided by Py. Now we can calculate this dy by dx.

(Refer Slide Time: 25:27)

```
backward_euler.sl (C:\Users\Administrator\Desktop\the_day\gpl\backward_euler.sl) - SciNotes
File Edit Format Options Window Execute ?
backward_euler.sl (C:\Users\Administrator\Desktop\the_day\gpl\backward_euler.sl) - SciNotes
backward_euler.sl [
32 while ~err (while) ~ eps_max
33   aerror=(Gfunc(yn)/Gderi(yn));
34   yn=yn-aerror;
35 end
36 disp(yn)
37 -----
38 //Main dydx calculation
39 1 function dydx = psi(x,y)
40 2 A_y=B*y;
41 3 P_y=B+2*y;
42 4 R_y=A_y/P_y;
43 5 Sf=(n^2*Q^2)/(R_y)^(4/3)*A_y^2;
44 6 Frs=(Q^2*B)/(g*(B*y)^3);
45 7 dydx=(S0-Sf)/(1-Frs);
46 endfunction
47
48 1 function Fval=Ffunc(y,x,delta_x,yp)
49 2   Fval-y-delta_x*psi(x,y)-yp
50 endfunction
51
52 1 function Fpval=Fderi(y,delta_x)
53 2   trm1=(1-Q^2/(B^2*g*y^3))^(-1);
54 3   trm2=(2*n^2*Q^2)/(B^2*y^3);
55 4   trm3=(B*y/(B+2*y))^(4/3);
56 5   trm4=(4*n^2*Q^2)/(3*B^2*y^2);
57 6   trm5=(B*y/(B+2*y))^(7/3);
58 7   trm6=((B/(B+2*y))-(2*B*y/(B+2*y)^2));
59 8   trm7=(3*Q^2)/(B^2*g*y^4);
60 9   trm8=(1-Q^2/(B^2*g*y^3))^(-2);
61 10  trm9=S0;
62 11  trm10=(n^2*Q^2)/(B^2*y^2);
63 12  trm11=(B*y/(B+2*y))^(4/3);
64 13  Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
65 endfunction
```



Next step is Fval because we need that function val. Y is actually yn plus 1. This is delta x into psi x comma y. And at this step we have this minus yp. Yp is previous time (val) level value.

(Refer Slide Time: 25:57)

```

backward_euler.m (C:\Users\Administrator\Desktop\the day\qcf\mpic\backward_euler.m) - SciNotes
File Edit Format Options Window Escape T
backward_euler.m (C:\Users\Administrator\Desktop\the day\qcf\mpic\backward_euler.m) - SciNotes
backward_euler.m
5 Sf=(n^2*Q^2)/( (R_y)^ (4/3)*A_y^2);
6 Frs=(Q^2*B)/(g*(B*y)^3);
7 dydx=(S0-Sf)/(1-Frs);
8 endfunction
47
1 function Fval=Ffunc(y,x,delta_x,yp)
2     Fval=y-delta_x*pai(x,y)-yp
3 endfunction
51
1 function Fpval=Fderi(y,delta_x)
2     trm1=(1-Q^2/(B^2*g*y^3))^(-1);
3     trm2=(2*n^2*Q^2)/(B^2*y^3);
4     trm3=(B*y/(B+2*y))^(-4/3);
5     trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6     trm5=(B*y/(B+2*y))^(-7/3);
7     trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8     trm7=(3*Q^2)/(B^2*g*y^4);
9     trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10    trm9=S0;
11    trm10=(n^2*Q^2)/(B^2*y^2);
12    trm11=(B*y/(B+2*y))^(-4/3);
13    Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75     //Newton-Raphson
76     eps_max=1e-6;

```

Now we need to calculate this term by term Fp value or derivative of F with respect to y plus 1. Now if we check the first term here so this is 1 minus Q square divided by B square g cube n plus 1.

(Refer Slide Time: 26:28)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Backward Euler Method

From Newton-Raphson method,

$$y_{n+1}^{(p)} = y_{n+1}^{(p-1)} - \frac{F(y_{n+1}^{(p-1)})}{F'(y_{n+1}^{(p-1)})}$$

where

$$F'(y_{n+1}) = 1 - \Delta x \left[\left(1 - \frac{Q^2}{B^2 g y_{n+1}^3}\right)^{-1} \left[\left(\frac{2n^2 Q^2}{B^2 y_{n+1}^3}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{4}{3}} + \left(\frac{4n^2 Q^2}{3B^2 y_{n+1}^3}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{7}{3}} \left(\frac{B}{B + 2y_{n+1}} - \frac{2B y_{n+1}}{(B + 2y_{n+1})^2}\right) \right] - \left(\frac{3Q^2}{B^2 g y_{n+1}^4}\right) \left(1 - \frac{Q^2}{B^2 g y_{n+1}^3}\right)^{-2} \left[S_0 - \left(\frac{n^2 Q^2}{B^2 y_{n+1}^2}\right) \left(\frac{B y_{n+1}}{B + 2y_{n+1}}\right)^{-\frac{4}{3}} \right] \right]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 12 / 21

So same thing I have written here to the power minus 1. Then second term 2n square Q square B square y cube which is the second term here.

(Refer Slide Time: 26:58)

```
backward_euler_so
5 Sf=(n^2*Q^2)/((R_y)^(4/3)*A_y^2);
6 Frs=(Q^2*B)/(g*(B*y)^3);
7 dydx=(S0-Sf)/(1-Frs);
8 endfunction
47
1 function Fval=Efunc(y,x,delta_x,yp)
2   Fval=y-delta_x*psi(x,y)-yp
3 endfunction
51
1 function Fpval=Ederi(y,delta_x)
2   trm1=(1-Q^2/(B^2*g*y^3))^(-1);
3   trm2=(2*n^2*Q^2)/(B^2*y^3);
4   trm3=(B*y/(B+2*y))^(4/3);
5   trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6   trm5=(B*y/(B+2*y))^(7/3);
7   trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8   trm7=(3*Q^2)/(B^2*g*y^4);
9   trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10  trm9=S0;
11  trm10=(n^2*Q^2)/(B^2*y^2);
12  trm11=(B*y/(B+2*y))^(4/3);
13  Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75   //Newton-Raphson
76   eps_max=1e-6;
```

Third term is $B y^n$ plus $1 B$ plus this $2 y^n$ plus 1 . So same thing if I write here this is the thing. Now like that I have written all the terms in this F prime calculation. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 terms here.

(Refer Slide Time: 27:35)

```
backward_euler_so
5 Sf=(n^2*Q^2)/((R_y)^(4/3)*A_y^2);
6 Frs=(Q^2*B)/(g*(B*y)^3);
7 dydx=(S0-Sf)/(1-Frs);
8 endfunction
47
1 function Fval=Efunc(y,x,delta_x,yp)
2   Fval=y-delta_x*psi(x,y)-yp
3 endfunction
51
1 function Fpval=Ederi(y,delta_x)
2   trm1=(1-Q^2/(B^2*g*y^3))^(-1);
3   trm2=(2*n^2*Q^2)/(B^2*y^3);
4   trm3=(B*y/(B+2*y))^(4/3);
5   trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6   trm5=(B*y/(B+2*y))^(7/3);
7   trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8   trm7=(3*Q^2)/(B^2*g*y^4);
9   trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10  trm9=S0;
11  trm10=(n^2*Q^2)/(B^2*y^2);
12  trm11=(B*y/(B+2*y))^(4/3);
13  Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75   //Newton-Raphson
76   eps_max=1e-6;
```

Now after getting these terms we can simply, this is term 1 into term 2 into term 3 plus term 4, term 5, term 6. So we can get this Fpval or F prime value from this derivative calculation.

(Refer Slide Time: 27:58)

```

5 Sf=(n^2*Q^2)/( (R_y)^ (4/3)*A_y^2);
6 Frs=(Q^2*B)/(g*(B*y)^3);
7 dydx=(S0-Sf)/(1-Frs);
8 endfunction
47
1 function Fval=Efunc(y,x,delta_x,yp)
2     Fval=y-delta_x*pai(x,y)-yp
3 endfunction
51
1 function Fpval=Ederi(y,delta_x)
2     trm1=(1-Q^2/(B^2*g*y^3))^( -1);
3     trm2=(2*n^2*Q^2)/(B^2*y^3);
4     trm3=(B*y/(B+2*y))^( -4/3);
5     trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6     trm5=(B*y/(B+2*y))^( -7/3);
7     trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8     trm7=(3*Q^2)/(B^2*g*y^4);
9     trm8=(1-Q^2/(B^2*g*y^3))^( -2);
10    trm9=S0;
11    trm10=(n^2*Q^2)/(B^2*y^2);
12    trm11=(B*y/(B+2*y))^( -4/3);
13    Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75     //Newton-Raphson
76     eps_max=1e-6;
77     aerror=1;
78     while abs(aerror) > eps_max
79         aerror=(Efunc(yv(i+1),xc(i+1),delta_x,yv(i))/Ederi(yv(i+1),delta_x));
80         yv(i+1)=yv(i)+aerror;
81     end
82 end
83
84 //Plots
85 plot(xc,yv,"-b")
86
87 set(gca(),"auto_clear","off")
88 plot(0 Lx,[y0 y1],"-m")
89

```

Now we need to use this one in our backward Euler method. So first of all we need to have this grid which is xc. Xc is running from zero to Lx. We have all total mnode, number of nodes which is 201 in our case. In this case Lx is 200 metres and delta x is Lx divided by mnode minus 1. So this is individual del x we can calculate. Now in this case yv is our variable. We need this y value.

(Refer Slide Time: 29:07)

```

5     trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6     trm5=(B*y/(B+2*y))^( -7/3);
7     trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8     trm7=(3*Q^2)/(B^2*g*y^4);
9     trm8=(1-Q^2/(B^2*g*y^3))^( -2);
10    trm9=S0;
11    trm10=(n^2*Q^2)/(B^2*y^2);
12    trm11=(B*y/(B+2*y))^( -4/3);
13    Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75     //Newton-Raphson
76     eps_max=1e-6;
77     aerror=1;
78     while abs(aerror) > eps_max
79         aerror=(Efunc(yv(i+1),xc(i+1),delta_x,yv(i))/Ederi(yv(i+1),delta_x));
80         yv(i+1)=yv(i)+aerror;
81     end
82 end
83
84 //Plots
85 plot(xc,yv,"-b")
86
87 set(gca(),"auto_clear","off")
88 plot(0 Lx,[y0 y1],"-m")
89

```

At the starting we have yv 1 equals to y not. Yv 1 which is specified at the first node, this is y not.

(Refer Slide Time: 29:21)

```

5 trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6 trm5=(B*y/(B+2*y))^(1/3);
7 trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8 trm7=(3*Q^2)/(B^2*g*y^4);
9 trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10 trm9=S0;
11 trm10=(n^2*Q^2)/(B^2*y^2);
12 trm11=(B*y/(B+2*y))^(4/3);
13 Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9+trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75 //Newton-Raphson
76 eps_max=1e-6;
77 aerror=1;
78 while abs(aerror) > eps_max
79 aerror=(Ffunc(yv(i+1),xc(i+1),delta_x,yv(i))/Eder1(yv(i+1),delta_x));
80 yv(i+1)=yv(i+1)-aerror;
81 end
82 end
83 end
84
85 //Plots
86 plot(xc,yv,"-b")
87
88 set(gca(),"auto_clear","off")
89 plot([0 Lx],[yn ynl,'-m')
  
```

Now we can start our calculation for i equals to 1 to m node minus 1. So epsilon max, this is the iteration parameter 1 into 10 to the power minus 6. Aerror, this is 1 or absolute error in this case. Now in this case we are starting from the guess value yv i plus 1 equals to yv i .

(Refer Slide Time: 29:57)

```

5 trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6 trm5=(B*y/(B+2*y))^(1/3);
7 trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8 trm7=(3*Q^2)/(B^2*g*y^4);
9 trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10 trm9=S0;
11 trm10=(n^2*Q^2)/(B^2*y^2);
12 trm11=(B*y/(B+2*y))^(4/3);
13 Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9+trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75 //Newton-Raphson
76 eps_max=1e-6;
77 aerror=1;
78 yv(i+1)=yv(i);
79 while abs(aerror) > eps_max
80 aerror=(Ffunc(yv(i+1),xc(i+1),delta_x,yv(i))/Eder1(yv(i+1),delta_x));
81 yv(i+1)=yv(i+1)-aerror;
82 end
83 end
84
85 //Plots
86 plot(xc,yv,"-b")
87
88 set(gca(),"auto_clear","off")
89 plot([0 Lx],[yn ynl,'-m')
  
```

Now we should iterate if absolute error is greater than epsilon max. So what is this absolute error? Absolute error is absolute value of aerror. Aerror is f func that means function value at y_n plus 1 for previous iteration divided by F prime y_n plus 1. That is also calculated based on the previous iteration. Now this is minus and this is y_n plus 1. So from this one we can get the solution.

(Refer Slide Time: 30:55)

```

5   trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6   trm5=(B*y/(B+2*y))^(1/3);
7   trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8   trm7=(3*Q^2)/(B^2*g*y^4);
9   trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10  trm9=S0;
11  trm10=(n^2*Q^2)/(B^2*y^2);
12  trm11=(B*y/(B+2*y))^(4/3);
13  Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
14 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75     //Newton-Raphson
76     eps_max=1e-6;
77     aerror=1;
78     yv(i+1)=yv(i);
79     while abs(aerror) > eps_max
80         aerror=(Ffunc(yv(i+1),xc(i+1),delta_x,yv(i))/Ederi(yv(i+1),delta_x));
81         yv(i+1)=yv(i+1)-aerror;
82     end
83 end
84
85 //Plots
86 plot(xc,yv,"-b")
87 set(gca(),"auto_clear","off")
88 plot([0 Lx],[yn ynl,'-m')
89

```

$$y_{n+1} = y_n - \frac{F(y_{n+1}^{(p-1)})}{F'(y_{n+1}^{(p-1)})}$$



So in this case we have this y_{n+1} which is coming from iteration P and this is y_{n+1} which is coming from iteration $P-1$ and minus F by F' and this was y_{n+1} , not y_n .

(Refer Slide Time: 31:41)

```

51 function Fpval=Ederi(y,delta_x)
52 trm1=(1-Q^2/(B^2*g*y^3))^(-1);
53 trm2=(2*n^2*Q^2)/(B^2*y^3);
54 trm3=(B*y/(B+2*y))^(4/3);
55 trm4=(4*n^2*Q^2)/(3*B^2*y^2);
56 trm5=(B*y/(B+2*y))^(1/3);
57 trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
58 trm7=(3*Q^2)/(B^2*g*y^4);
59 trm8=(1-Q^2/(B^2*g*y^3))^(-2);
60 trm9=S0;
61 trm10=(n^2*Q^2)/(B^2*y^2);
62 trm11=(B*y/(B+2*y))^(4/3);
63 Fpval=1-delta_x*(trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11));
64 endfunction
66 // Backward Euler Method
67 //Grid Data
68 xc=linspace(0,Lx,mnode);
69 delta_x=Lx/(mnode-1);
70 //Initialization
71 yv=zeros(mnode,1);
72
73 yv(1)=y0;
74 for i=1:mnode-1
75     //Newton-Raphson
76     eps_max=1e-6;
77     aerror=1;
78     yv(i+1)=yv(i);
79     while abs(aerror) > eps_max
80         aerror=(Ffunc(yv(i+1),xc(i+1),delta_x,yv(i))/Ederi(yv(i+1),delta_x));
81         yv(i+1)=yv(i+1)-aerror;
82     end
83 end
84

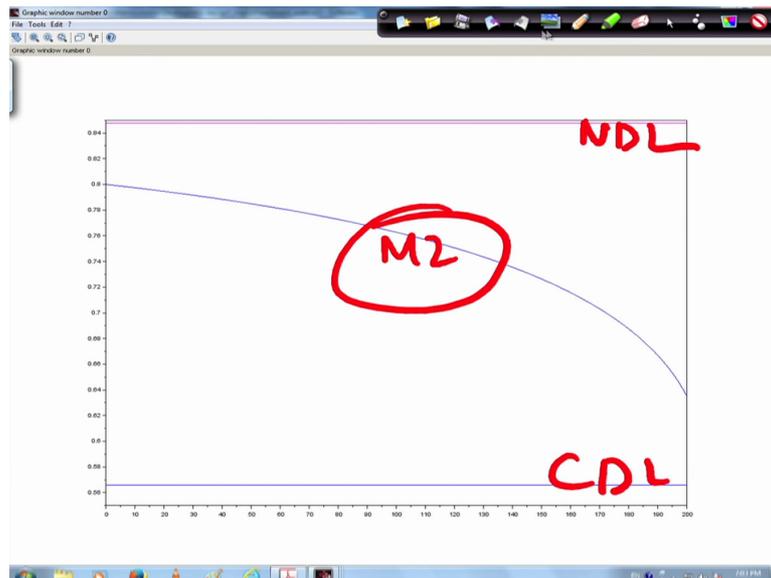
```

$$y_{n+1}^{(p)} = y_{n+1}^{(p-1)} - \frac{F}{F'}$$



So we can use this for calculation of Newton Raphson. Now this is minus error, this is x_c, y . Now if you run it, similar to our explicit method we are getting $M2$ curve in this case. This is NDL or normal depth line. This is CDL or critical depth line.

(Refer Slide Time: 32:18)



From our calculation point of view this y_n is point 8478 and y_c is point 5658. Now if we try to utilise the higher level methods or higher order methods in this case then if we have implicit Runge Kutta in this case, please remember that we have this K_i and i is there. That is why it is implicit.

(Refer Slide Time: 33:07)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

Increments can be written as

$$\begin{aligned}
 K_i &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right) \\
 &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^{i-1} c_{ij}^y K_j + c_{ii}^y K_i \right) \\
 &= \Delta x \Psi \left(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i \right)
 \end{aligned}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now what we can do, we can split it into explicit part plus some increment. This $C_{ii} y K_i$, this is the increment part. So we can write this quantity, this is again explicit in nature. So this is explicit, this is explicit, only implicit is this quantity. This is $C_{ii} y K_i$.

(Refer Slide Time: 33:44)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

Increments can be written as

$$\begin{aligned}
 K_i &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right) \\
 &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^{i-1} c_{ij}^y K_j + c_{ii}^y K_i \right) \\
 &= \Delta x \Psi \left(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i \right)
 \end{aligned}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now in this case we have this delta x which is only $C_i \Delta x$ and delta y which is our $\sum_{j=1}^{i-1} C_{ij} K_j$. This is explicit quantity.

(Refer Slide Time: 34:17)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

Increments can be written as

$$\begin{aligned}
 K_i &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right) \\
 &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^{i-1} c_{ij}^y K_j + c_{ii}^y K_i \right) \\
 &= \Delta x \Psi \left(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i \right)
 \end{aligned}$$

with $\delta_x = c_i^x \Delta x$ and $\delta_y = \sum_{j=1}^{i-1} c_{ij}^y K_j$

The the multivariate function $\Psi()$ can be expanded as

$$\Psi \left(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i \right) = \Psi \left(x_n + \delta_x, y_n + \delta_y \right) + c_{ii}^y \Psi' \left(x_n + \delta_x, y_n + \delta_y \right)$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now in this case we have this $\Psi(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i)$. So this is increment part. So we can expand it with Taylor series and this is a first order expansion. So, on the right hand side we have $x_n + \delta_x, y_n + \delta_y$ and $c_{ii}^y K_i$. This is $x_n + \delta_x, y_n + \delta_y$ and this quantity is multiplied there and we are differentiating with respect to y here.

(Refer Slide Time: 35:11)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

Increments can be written as

$$\begin{aligned}
 K_i &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^i c_{ij}^y K_j \right) \\
 &= \Delta x \Psi \left(x_n + c_i^x \Delta x, y_n + \sum_{j=1}^{i-1} c_{ij}^y K_j + c_{ii}^y K_i \right) \\
 &= \Delta x \Psi \left(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i \right)
 \end{aligned}$$

with $\delta_x = c_i^x \Delta x$ and $\delta_y = \sum_{j=1}^{i-1} c_{ij}^y K_j$

The the multivariate function $\Psi()$ can be expanded as

$$\Psi \left(x_n + \delta_x, y_n + \delta_y + c_{ii}^y K_i \right) = \Psi \left(x_n + \delta_x, y_n + \delta_y \right) + c_{ii}^y \Psi' \left(x_n + \delta_x, y_n + \delta_y \right) K_i$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

So in this case if we combine the terms so this is $K_i \delta_x$ into Ψ which is our known quantity and this K_i is unknown.

(Refer Slide Time: 35:36)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

By combining all the expressions

$$K_i = \Delta x \left[\Psi \left(x_n + \delta_x, y_n + \delta_y \right) + c_{ii}^y \Psi' \left(x_n + \delta_x, y_n + \delta_y \right) K_i \right]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 14 / 21

Now if I transfer this thing on the left hand side so I can write this as $1 - c_{ii}^y$ and y . This is $\delta_x \Psi$ plus prime into K_i in this case and this is $\delta_x \Psi(x_n + \delta_x, y_n + \delta_y)$. So I can transfer this on the right hand side. This is inverse term δ_x .

(Refer Slide Time: 36:28)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

By combining all the expressions

$$K_i = \Delta x [\Psi(x_n + \delta_x, y_n + \delta_y) + c_{ii}^y \Psi'(x_n + \delta_x, y_n + \delta_y) K_i]$$

In implicit compact form it can be written as

$$K_i = \Delta x [1 - c_{ii}^y \Delta x \Psi'(x_n + \delta_x, y_n + \delta_y)]^{-1} \Psi(x_n + \delta_x, y_n + \delta_y)$$

where

$$\Psi'(x, y) = \left(1 - \frac{Q^2}{B^2 g y^3}\right)^{-1} \left[\left(\frac{2n^2 Q^2}{B^2 y^3}\right) \left(\frac{By}{B+2y}\right)^{-\frac{4}{3}} + \left(\frac{4n^2 Q^2}{3B^2 y^2}\right) \left(\frac{By}{B+2y}\right)^{-\frac{7}{3}} \left(\frac{B}{B+2y} - \frac{2By}{(B+2y)^2}\right) \right] - \left(\frac{3Q^2}{B^2 g y^4}\right) \left(1 - \frac{Q^2}{B^2 g y^3}\right)^{-2} \left[S_0 - \left(\frac{n^2 Q^2}{B^2 y^2}\right) \left(\frac{By}{B+2y}\right)^{-\frac{4}{3}}\right]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 14 / 21

Now from this case we have this psi prime. Psi prime value we can calculate and we can directly get the Ki value here. So Ki value we can get in terms of known values because x_n , δx , y_n , δy and psi and psi prime, these quantities are known. And for a given Runge Kutta method C_{ii} , this quantity or this parameter is known and δx is fixed for a particular problem.

(Refer Slide Time: 37:14)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Implicit Runge-Kutta

By combining all the expressions

$$K_i = \Delta x [\Psi(x_n + \delta_x, y_n + \delta_y) + c_{ii}^y \Psi'(x_n + \delta_x, y_n + \delta_y) K_i]$$

In implicit compact form it can be written as

$$K_i = \Delta x [1 - c_{ii}^y \Delta x \Psi'(x_n + \delta_x, y_n + \delta_y)]^{-1} \Psi(x_n + \delta_x, y_n + \delta_y)$$

where

$$\Psi'(x, y) = \left(1 - \frac{Q^2}{B^2 g y^3}\right)^{-1} \left[\left(\frac{2n^2 Q^2}{B^2 y^3}\right) \left(\frac{By}{B+2y}\right)^{-\frac{4}{3}} + \left(\frac{4n^2 Q^2}{3B^2 y^2}\right) \left(\frac{By}{B+2y}\right)^{-\frac{7}{3}} \left(\frac{B}{B+2y} - \frac{2By}{(B+2y)^2}\right) \right] - \left(\frac{3Q^2}{B^2 g y^4}\right) \left(1 - \frac{Q^2}{B^2 g y^3}\right)^{-2} \left[S_0 - \left(\frac{n^2 Q^2}{B^2 y^2}\right) \left(\frac{By}{B+2y}\right)^{-\frac{4}{3}}\right]$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 14 / 21

Now in this case if we utilise it for second order RK method or RK2 then considering the Butcher tableau we have half, half. That means coefficient of x is half, coefficient of y is half.

So this is nothing but K_1 equals to Δx and we have $\psi(x_n + \frac{1}{2} \Delta x)$ plus half K_1 . And final thing is y_{n+1} equals to $y_n + K_1$.

(Refer Slide Time: 38:12)

The slide is titled "Second Order RK Method (RK2)". It contains the following content:

- Navigation menu: Problem Definition, General Structure, Backward Euler Method, Implicit Runge-Kutta Methods, References.
- Institution: I.I.T. Kharagpur.
- Text: "Considering Butcher Tableau as"
- Butcher Tableau:

$\frac{1}{2}$	$\frac{1}{2}$
$-\frac{1}{2}$	1
- Handwritten equations in red:

$$K_1 = \Delta x \psi(x_n + \frac{1}{2} \Delta x, y_n + \frac{1}{2} K_1)$$

$$y_{n+1} = y_n + K_1$$
- Footer: Dr. Anirban Dhar, NPTEL, Computational Hydraulics, 15 / 21.

So if we utilise this, this is second order method. Now what should be our approach? We can expand this and expansion is in terms of our known quantities. Now in this case we can expand this part using Taylor series as $\psi(x_n + \frac{1}{2} \Delta x)$. This quantity is y_n which is known, plus we can write ψ' and we can multiply our half into K_1 in this case. If we multiply again Δx outside and equate it with K_1 we can get this particular format.

So, on the right hand side we have all known quantities. ψ , ψ' , these are known quantities on the right hand side. So we can get this K_1 value.

(Refer Slide Time: 39:40)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Second Order RK Method (RK2)

Considering Butcher Tableau as

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

RK2

$$y_{n+1} = y_n + K_1$$

with $K_1 = \Delta x \Psi \left(x_n + \frac{1}{2} \Delta x, y_n + \frac{1}{2} K_1 \right)$

Order of RK2 method: $\mathcal{O}(\Delta x^2)$
Semi-Implicit Equation $K_1 = \Delta x \Psi \left(x_n + \frac{1}{2} \Delta x, y_n + \frac{1}{2} K_1 \right)$

$$K_1 = \Delta x \left[1 - \frac{1}{2} \Delta x \Psi' \left(x_n + \frac{1}{2} \Delta x, y_n \right) \right]^{-1} \Psi \left(x_n + \frac{1}{2} \Delta x, y_n \right)$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 15 / 21

Now if we see the code for this one, this is RK2 implicit. First portion is same. In this case we have RK2 method. We need grid data in this case also. Xx, delta x, yv, this is yv 1 equals to y not. And then we are calculating the K1. Now K1 in this case this is coming from this known quantity. This is 1 minus half delta x, psi or psi derivative, these are known quantities.

This is psi derivative, this is Psi value directly. And we are considering inverse of that value and we are adding this K1 with our previous time value to get the final plot.

(Refer Slide Time: 41:01)

```

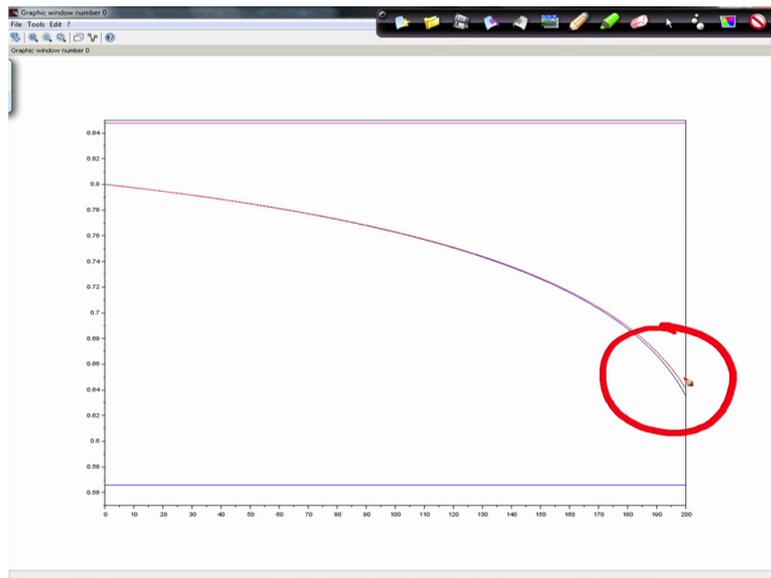
4 trm3=(B*y/(B+2*y))^(4/3);
5 trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6 trm5=(B*y/(B+2*y))^(7/3);
7 trm6=((B/(B+2*y))-(2*B*y/(B+2*y)^2));
8 trm7=(3*Q^2)/(B^2*g*y^4);
9 trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10 trm9=50;
11 trm10=(n^2*Q^2)/(B^2*y^2);
12 trm11=(B*y/(B+2*y))^(4/3);
13 pspival=trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11);
14 endfunction
62 // RK2 Method
63 //Grid Data
64 xc=linspace(0,Lx,mnode);
65 delta_x=Lx/(mnode-1);
66 //Initialization
67 yv=zeros(mnode,1);
68
69 yv(1)=y0;
70 for i=1:mnode-1
71     K1=delta_x*(1-(1/2)*delta_x*psider1(xc(i)+(1/2)*delta_x,yv(i)))^(-1)*psi(xc(i)+(1/2)*delta_x,yv(i));
72     yv(i+1)=yv(i)+K1;
73 end
74
75 //Plots
76 plot(xc,yv,"-r")
77
78 set(gca(),"auto_clear","off")
79 plot([0 Lx],[y0 y1],"-m")
80 set(gca(),"auto_clear","off")
81 plot([0 Lx],[yc yc],"-b")
82 xtitle("Backward Euler method", "X axis" "Flow Depth")
83 //hl=legend("Backward Method","Normal Depth Line","Critical Depth Line")
84

```

Now in this case we have RK2 method. Now let us try to plot this thing on top of our previous solution. So this is with red colour. Now if I compel this, so it is coming here. Now

it is interesting to see that red line which is RK2 method is somewhat on the upper side. That is it is giving higher value compared to the blue line which is our backward Euler method.

(Refer Slide Time: 41:51)



Now if we further increase the order of RK, second order RK approach we have this thing. Now considering the fourth order RK method which considers these many points that means we have two entries here and 2 by 2 matrix for C_y . This is for C_x , this is for C_y and this is for W_1 and W_2 .

(Refer Slide Time: 42:38)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Fourth Order RK Method (RK4)

Considering Butcher Tableau as

1	$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4} - \frac{\sqrt{3}}{6}$
2	$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{4} + \frac{\sqrt{3}}{6}$	$\frac{1}{4}$
		$\frac{1}{2}$	$\frac{1}{2}$

C_1 C_2 C_3 C_4 W_1 W_2 2×2

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now with this if I write this RK4, this is RK4 in this case. So RK4 is half-half because this is W_1 , this is W_2 and we have these terms. This K_1 is equal to Δx into C_1 which is coming from here, the first quantity. This C_2 is coming here, then C_{1-1} , this is the first one, C_{1-2} , then this quantity is C_{2-1} and last one this is C_{2-2} .

(Refer Slide Time: 43:32)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Fourth Order RK Method (RK4)

Considering Butcher Tableau as

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \end{array}$$

RK4

$$y_{n+1} = y_n + \frac{1}{5} K_1 + \frac{1}{2} K_2$$

with $K_1 = \Delta x \Psi(x_n + c_1^x \Delta x, y_n + c_{11}^y K_1 + c_{12}^y K_2)$
 $K_2 = \Delta x \Psi(x_n + c_2^x \Delta x, y_n + c_{21}^y K_1 + c_{22}^y K_2)$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now with this if we try our method or approach we can get the solution. This is fourth order RK. Now in this case we have K1 and K2. Now for K1 and K2 for both the cases we can expand it. Now expansion is in terms of both the terms. Only yn is known, K1 and K2 both are unknown. So this is our increment, increment. So psi prime, psi these are known quantities now. Now this is coming from Taylor series.

(Refer Slide Time: 44:17)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Fourth Order RK Method (RK4)

Expanded form of the increment equations can be written as

$$\begin{aligned} K_1 &= \Delta x \left[\Psi(x_n + c_1^x \Delta x, y_n) + (c_{11}^y K_1 + c_{12}^y K_2) \Psi'(x_n + c_1^x \Delta x, y_n) \right] \\ K_2 &= \Delta x \left[\Psi(x_n + c_2^x \Delta x, y_n) + (c_{21}^y K_1 + c_{22}^y K_2) \Psi'(x_n + c_2^x \Delta x, y_n) \right] \end{aligned}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Our next step is calculation of K1, K2. Now K1, K2 can be calculated by solving this system where we have 2 by 2 matrix which is the coefficient matrix. On right hand side also we have

known quantities. Now we can solve it using our standard approach and utilise it for the solution of fourth order RK.

(Refer Slide Time: 44:55)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Fourth Order RK Method (RK4)

Expanded form of the increment equations can be written as

$$K_1 = \Delta x [\Psi(x_n + c_1^x \Delta x, y_n) + (c_{11}^y K_1 + c_{12}^y K_2) \Psi'(x_n + c_1^x \Delta x, y_n)]$$

$$K_2 = \Delta x [\Psi(x_n + c_2^x \Delta x, y_n) + (c_{21}^y K_1 + c_{22}^y K_2) \Psi'(x_n + c_2^x \Delta x, y_n)]$$

By rearranging the expressions

$$\begin{bmatrix} 1 - \Delta x c_{11}^y \Psi'(x_n + c_1^x \Delta x, y_n) & -\Delta x c_{12}^y \Psi'(x_n + c_1^x \Delta x, y_n) \\ -\Delta x c_{21}^y \Psi'(x_n + c_2^x \Delta x, y_n) & 1 - \Delta x c_{22}^y \Psi'(x_n + c_2^x \Delta x, y_n) \end{bmatrix} \begin{Bmatrix} K_1 \\ K_2 \end{Bmatrix} = \begin{Bmatrix} \Delta x \Psi(x_n + c_1^x \Delta x, y_n) \\ \Delta x \Psi(x_n + c_2^x \Delta x, y_n) \end{Bmatrix}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics

Now if I open this fourth order RK, the first portion is same. Now in this case this is RK4 method. Now I have created this AK which is coefficient a or coefficient matrix for K, this is 2 by 2 zeros.

(Refer Slide Time: 45:26)

```

5 trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6 trm5=(B*y/(B+2*y))^(7/3);
7 trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8 trm7=(3*Q^2)/(B^2*g*y^4);
9 trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10 trm9=50;
11 trm10=(n^2*Q^2)/(B^2*y^2);
12 trm11=(B*y/(B+2*y))^(4/3);
13 psipval=trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11);
14 endfunction
62 // RK4 Method
63 //Grid Data
64 xc=linspace(0,Lx,mnode);
65 delta_x=Lx/(mnode-1);
66 //Initialization
67 yv=zeros(mnode,1);
68 AK=zeros(2,2);
69 %%Butcher Tableau
70 cx(1)=1/2+sqrt(3)/6;
71 cx(2)=1/2+sqrt(3)/6;
72 cy(1,1)=1/4;
73 cy(1,2)=1/4+sqrt(3)/6;
74 cy(2,1)=1/4+sqrt(3)/6;
75 cy(2,2)=1/4;
76 yv(1)=y0;
77 for i=1:mnode-1
78     AK(1,1)=1-delta_x*cy(1,1)*psideri(xc(i)+cx(1)*delta_x,yv(i));
79     AK(1,2)=-delta_x*cy(1,2)*psideri(xc(i)+cx(1)*delta_x,yv(i));
80     AK(2,1)=-delta_x*cy(2,1)*psideri(xc(i)+cx(2)*delta_x,yv(i));
81     AK(2,2)=1-delta_x*cy(2,2)*psideri(xc(i)+cx(2)*delta_x,yv(i));
82     RK(i)=delta_x*psi(xc(i)+cx(1)*delta_x,yv(i));
83     RK(i)=delta_x*psi(xc(i)+cx(2)*delta_x,yv(i));
84     KV=AKRK;
85     yv(i+1)=yv(i)+RK(i);

```

Now we have this cx1, cx2, cy 1-1, cy 1-2, cy 2-1, cy 2-2 and y1 equals to y not. These values are available.

(Refer Slide Time: 45:40)

```

5   trm4=(4*n^2*Q^2)/(3*B^2*y^2);
6   trm5=(B*y/(B+2*y))^(-7/3);
7   trm6=(B/(B+2*y))-(2*B*y/(B+2*y)^2);
8   trm7=(3*Q^2)/(B^2*g*y^4);
9   trm8=(1-Q^2/(B^2*g*y^3))^(-2);
10  trm9=80;
11  trm10=(n^2*Q^2)/(B^2*y^2);
12  trm11=(B*y/(B+2*y))^(-4/3);
13  psipval=trm1*(trm2*trm3+trm4*trm5*trm6)-trm7*trm8*(trm9-trm10*trm11);
14 endfunction
15 // RK Method
62 // Grid Data
63 xc=linspace(0,Lx,mnode);
64 delta_x=Lx/(mnode-1);
65 // Initialization
66 yv=zeros(mnode,1);
67 AK=zeros(2,2);
68 //Butcher Tableau
69 cx(1)=1/2+sqrt(3)/6;
70 cx(2)=1/2+sqrt(3)/6;
71 cy(1,1)=1/4;
72 cy(1,2)=1/4+sqrt(3)/6;
73 cy(2,1)=1/4+sqrt(3)/6;
74 cy(2,2)=1/4;
75 yv(1)=y0;
76 for i=1:mnode-1
77     AK(1,1)=1-delta_x*cy(1,1)*psideri(xc(i)+cx(1)*delta_x,yv(i));
78     AK(1,2)=-delta_x*cy(1,2)*psideri(xc(i)+cx(1)*delta_x,yv(i));
79     AK(2,1)=-delta_x*cy(2,1)*psideri(xc(i)+cx(2)*delta_x,yv(i));
80     AK(2,2)=1-delta_x*cy(2,2)*psideri(xc(i)+cx(2)*delta_x,yv(i));
81     rK(1)=delta_x*psi(xc(i)+cx(1)*delta_x,yv(i));
82     rK(2)=delta_x*psi(xc(i)+cx(2)*delta_x,yv(i));
83     Kv=AK*rK;
84     yv(i+1)=yv(i)+(1/2)*Kv(1)+(1/2)*Kv(2);
85 end
86 //Plots
87 plot(xc,yv,"-r")
88 set(gca(),"auto_clear","off")
89 plot(0 Lx,lyn ynl,"-m")
90 set(gca(),"auto_clear","off")
91 plot(0 Lx,lyc ycl,"-b")
92 xtitle("Backward Euler method", "X axis" "Flow-Depth")

```

Now within this calculation process starting from y equals to 1 to mnode minus 1, we can create that AK or coefficient matrix then calculate this K and RK which is on the right hand side. So AK is coefficient matrix. So in this case we can solve it using internal function this KV which is K value. K value we are directly getting from this backslash, this RK. Now from this one we have K1 and K2. So half of K1 and half of K2 if we add with the previous section value then we can get y i plus 1 in this case.

(Refer Slide Time: 46:36)

```

63 //Grid Data
64 xc=linspace(0,Lx,mnode);
65 delta_x=Lx/(mnode-1);
66 // Initialization
67 yv=zeros(mnode,1);
68 AK=zeros(2,2);
69 //Butcher Tableau
70 cx(1)=1/2+sqrt(3)/6;
71 cx(2)=1/2+sqrt(3)/6;
72 cy(1,1)=1/4;
73 cy(1,2)=1/4+sqrt(3)/6;
74 cy(2,1)=1/4+sqrt(3)/6;
75 cy(2,2)=1/4;
76 yv(1)=y0;
77 for i=1:mnode-1
78     AK(1,1)=1-delta_x*cy(1,1)*psideri(xc(i)+cx(1)*delta_x,yv(i));
79     AK(1,2)=-delta_x*cy(1,2)*psideri(xc(i)+cx(1)*delta_x,yv(i));
80     AK(2,1)=-delta_x*cy(2,1)*psideri(xc(i)+cx(2)*delta_x,yv(i));
81     AK(2,2)=1-delta_x*cy(2,2)*psideri(xc(i)+cx(2)*delta_x,yv(i));
82     rK(1)=delta_x*psi(xc(i)+cx(1)*delta_x,yv(i));
83     rK(2)=delta_x*psi(xc(i)+cx(2)*delta_x,yv(i));
84     Kv=AK\rK;
85     yv(i+1)=yv(i)+(1/2)*Kv(1)+(1/2)*Kv(2);
86 end
87 //Plots
88 plot(xc,yv,"-r")
89 set(gca(),"auto_clear","off")
90 plot(0 Lx,lyn ynl,"-m")
91 set(gca(),"auto_clear","off")
92 plot(0 Lx,lyc ycl,"-b")
93 xtitle("Backward Euler method", "X axis" "Flow-Depth")

```

$AKK_v = rK$
 $K_v = AK \setminus rK$

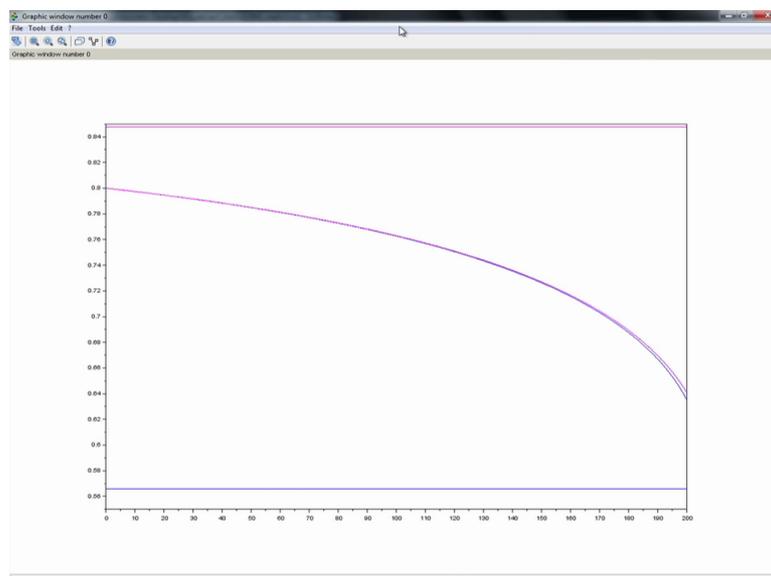
Similarly this approach is our RK4 approach. Now I can plot this one, this is again red one or magenta. I can change the colour in this case.

(Refer Slide Time: 46:46)

```
63 //Grid Data
64 xc=linspace(0,Lx,mnode);
65 delta_x=Lx/(mnode-1);
66 //Initialization
67 yv=zeros(mnode,1);
68 AK=zeros(2,2);
69 //Butcher Tableau
70 cx(1)=1/2+sqrt(3)/6;
71 cx(2)=1/2+sqrt(3)/6;
72 cy(1,1)=1/4;
73 cy(1,2)=1/4+sqrt(3)/6;
74 cy(2,1)=1/4+sqrt(3)/6;
75 cy(2,2)=1/4;
76 yv(1)=y0;
77 for i=1:mnode-1
78     AK(1,1)=1-delta_x*cy(1,1)*psideri(xc(i)+cx(1)*delta_x,yv(i));
79     AK(1,2)=-delta_x*cy(1,2)*psideri(xc(i)+cx(1)*delta_x,yv(i));
80     AK(2,1)=-delta_x*cy(2,1)*psideri(xc(i)+cx(2)*delta_x,yv(i));
81     AK(2,2)=1-delta_x*cy(2,2)*psideri(xc(i)+cx(2)*delta_x,yv(i));
82     Rk(1)= delta_x*psi(xc(i)+cx(1)*delta_x,yv(i));
83     Rk(2)= delta_x*psi(xc(i)+cx(2)*delta_x,yv(i));
84     Kv=AK*Rk;
85     yv(i+1)=yv(i)+(1/2)*Kv(1)+(1/2)*Kv(2);
86 end
87
88 //Plots
89 plot(xc,yv,"-r")
90
91 set(gca(),"auto_clear","off")
92 plot(0 Lx,lyn ynl,'-m')
93 set(gca(),"auto_clear","off")
94 plot(0 Lx,lyc ycl,'-b')
95 xtitle ("Backward Euler Method", "X axis" "Flow Depth")
```

And if I run it, again it is coinciding with our second order or it is near about value of our second order method. If we zoom it, obviously we will get some difference for this one.

(Refer Slide Time: 47:26)



Now in this case we have seen that using backward Euler second order RK and RK4 we can solve our GVF profile equation that is the first order initial value problem with implicit equations.

(Refer Slide Time: 47:55)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

Fourth Order RK Method (RK4)

Expanded form of the increment equations can be written as

$$K_1 = \Delta x [\Psi(x_n + c_1^x \Delta x, y_n) + (c_{11}^y K_1 + c_{12}^y K_2) \Psi'(x_n + c_1^x \Delta x, y_n)]$$
$$K_2 = \Delta x [\Psi(x_n + c_2^x \Delta x, y_n) + (c_{21}^y K_1 + c_{22}^y K_2) \Psi'(x_n + c_2^x \Delta x, y_n)]$$

By rearranging the expressions

$$\begin{bmatrix} 1 - \Delta x c_{11}^y \Psi'(x_n + c_1^x \Delta x, y_n) & -\Delta x c_{12}^y \Psi'(x_n + c_1^x \Delta x, y_n) \\ -\Delta x c_{21}^y \Psi'(x_n + c_2^x \Delta x, y_n) & 1 - \Delta x c_{22}^y \Psi'(x_n + c_2^x \Delta x, y_n) \end{bmatrix} \begin{Bmatrix} K_1 \\ K_2 \end{Bmatrix} = \begin{Bmatrix} \Delta x \Psi(x_n + c_1^x \Delta x, y_n) \\ \Delta x \Psi(x_n + c_2^x \Delta x, y_n) \end{Bmatrix}$$

Dr. Anirban Dhar NPTEL Computational Hydraulics 18 / 21

And we have discussed this backward Euler approach, RK2 approach, RK4 approach. Now you can utilise these codes for running and creating the GVF profiles. And you can change different values and get different kind of GVF profiles out of this.

(Refer Slide Time: 48:22)

Problem Definition
General Structure
Backward Euler Method
Implicit Runge-Kutta Methods
References

I.I.T. Kharagpur

List of Source Codes

Gradually Varied Flow-Implicit Approach

- Backward Euler approach
 - [backward_euler.sci](#)
- RK2 approach
 - [RK2_implicit.sci](#)
- RK4 approach
 - [RK4_implicit.sci](#)

Dr. Anirban Dhar NPTEL Computational Hydraulics 19 / 21

Thank you.