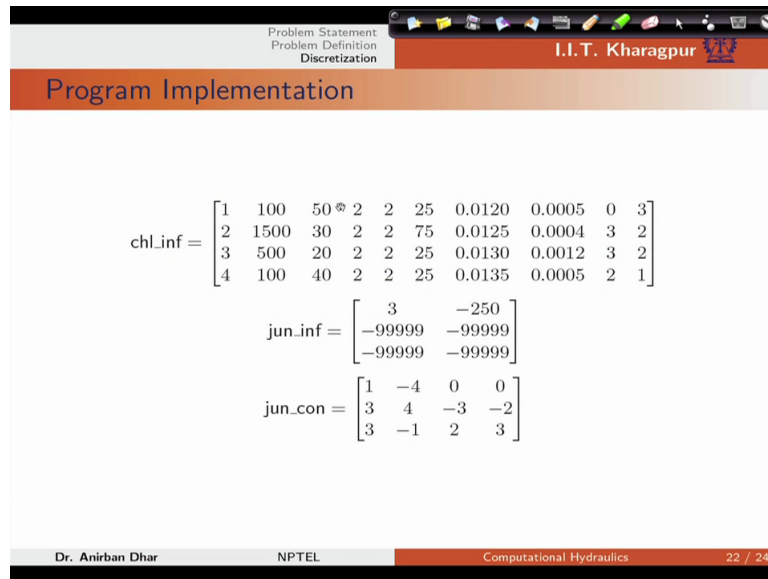


Computational Hydraulics
Professor Anirban Dhar
Department of Civil Engineering
Indian Institute of Technology Kharagpur
Lecture 41

Steady Channel Flow: Channel Network without Reverse Flow (Contd.)

Now if I open that scilab code for this one.

(Refer Slide Time: 00:26)



The screenshot shows a Scilab presentation slide titled "Program Implementation" from I.I.T. Kharagpur. It displays three matrices defined in Scilab code:

$$\text{chl_inf} = \begin{bmatrix} 1 & 100 & 50 & 2 & 2 & 25 & 0.0120 & 0.0005 & 0 & 3 \\ 2 & 1500 & 30 & 2 & 2 & 75 & 0.0125 & 0.0004 & 3 & 2 \\ 3 & 500 & 20 & 2 & 2 & 25 & 0.0130 & 0.0012 & 3 & 2 \\ 4 & 100 & 40 & 2 & 2 & 25 & 0.0135 & 0.0005 & 2 & 1 \end{bmatrix}$$
$$\text{jun_inf} = \begin{bmatrix} 3 & -250 \\ -99999 & -99999 \\ -99999 & -99999 \end{bmatrix}$$
$$\text{jun_con} = \begin{bmatrix} 1 & -4 & 0 & 0 \\ 3 & 4 & -3 & -2 \\ 3 & -1 & 2 & 3 \end{bmatrix}$$

The slide footer includes "Dr. Anirban Dhar", "NPTEL", "Computational Hydraulics", and "22 / 24".

So code is steady 1D channel network without reverse flow situation. So initial program it is starting with `clc clear`, so clear console clear all variables in this case. Next is defining A_v or area value. Next is d area, this is the dA by dy . Next is HR value that means this is hydraulic radius RH . Next is dR by dy . So these information are required for our problem.

(Refer Slide Time: 01:28)

```

1 clear
2 function Av=areav(y,B,m1,m2)
3 Av=B*y+(1/2)*(m1+m2)*y^2;
4 endfunction
5 function dAv=dareav(y,B,m1,m2)
6 dAv=B+(m1+m2)*y;
7 endfunction
8 function Rv=HRV(y,B,m1,m2)
9 Rv=(B*y^(1/2)*(m1+m2)*y^2)/(B+(sqrt(1+m1^2)+sqrt(1+m2^2))*y);
10 endfunction
11 function dRv=dHRV(y,B,m1,m2)
12 //dx=2*B^2+2*B*(m1+m2)*y+(m1+m2)*(sqrt(1+m1^2)+sqrt(1+m2^2))*y^2;
13 //dx=2*(B+(sqrt(1+m1^2)+sqrt(1+m2^2))*y)^2;----
14 Tw=B*(m1+m2)*y;
15 Fm=B*(sqrt(1+m1^2)+sqrt(1+m2^2))*y;
16 Rh=HRV(y,B,m1,m2);
17 dPdy=(sqrt(1+m1^2)+sqrt(1+m2^2));
18 dRv=(Tw/Fm)-(Rh/Fm)*dPdy;
19 endfunction
20
21 function Mlv=Mli(y1,Q1,y2,Q2,S0,delta_x,D1,D2,B,m1,m2)
22 Mlv=(y2-y1)-S0*delta_x*D1*(Q2^2*areav(y2,B,m1,m2)^(-2)-Q1^2*areav(y1,B,m1,m2)^(-2))+D2*(Q2^2*HRV(y2,B,m1,m2)^(-4/3)-Q1^2*HRV(y1,B,m1,m2)^(-4/3))*areav(y1,B,m1,m2)^(-2);
23 endfunction
24 function dMdyiv=dMdyi(y,Q,D1,D2,B,m1,m2)
25 term1=(2*Q^2/areav(y,B,m1,m2)^3)*dareav(y,B,m1,m2);
26 term2=2*Q^2*areav(y,B,m1,m2)^(-3)*HRV(y,B,m1,m2)^(-4/3)*dareav(y,B,m1,m2);
27 term3=(4/3)*Q^2*areav(y,B,m1,m2)^(-2)*HRV(y,B,m1,m2)^(-7/3)*dHRV(y,B,m1,m2);
28 dMdyiv=-D1*term1-D2*(term2+term3);
29 endfunction
30 function dMdyiplv=dMdyipl(y,Q,D1,D2,B,m1,m2)
31 term1=(2*Q^2/areav(y,B,m1,m2)^3)*dareav(y,B,m1,m2);
32 term2=2*Q^2*areav(y,B,m1,m2)^(-3)*HRV(y,B,m1,m2)^(-4/3)*dareav(y,B,m1,m2);
33 term3=(4/3)*Q^2*areav(y,B,m1,m2)^(-2)*HRV(y,B,m1,m2)^(-7/3)*dHRV(y,B,m1,m2);
34 dMdyiplv=-D1*term1-D2*(term2+term3);
35 endfunction
36 function dMdyiv=dMdyi(y,Q,D1,D2,B,m1,m2)
37 term1=(2*Q^2/areav(y,B,m1,m2)^3)*dareav(y,B,m1,m2);
38 term2=2*Q^2*areav(y,B,m1,m2)^(-3)*HRV(y,B,m1,m2)^(-4/3)*dareav(y,B,m1,m2);
39 term3=(4/3)*Q^2*areav(y,B,m1,m2)^(-2)*HRV(y,B,m1,m2)^(-7/3)*dHRV(y,B,m1,m2);
40 dMdyiv=-D1*term1-D2*(term2+term3);
41 endfunction
42 function dMdyiplv=dMdyipl(y,Q,D1,D2,B,m1,m2)
43 term1=(2*Q^2/areav(y,B,m1,m2)^3)*dareav(y,B,m1,m2);
44 term2=2*Q^2*areav(y,B,m1,m2)^(-3)*HRV(y,B,m1,m2)^(-4/3)*dareav(y,B,m1,m2);
45 term3=(4/3)*Q^2*areav(y,B,m1,m2)^(-2)*HRV(y,B,m1,m2)^(-7/3)*dHRV(y,B,m1,m2);
46 dMdyiplv=-D1*term1-D2*(term2+term3);
47 endfunction

```

Now next part is our $dM L i$ by this is MLV only so this is $M L i$ only. On the second one this $dM L i$ by $dy L i$. Next one this is the one. Next one is $dM L i$ divided by $dy L i$ plus 1. This is $dM L i$ by $dQ L i$ plus 1 and this is $dM L i$ by $dQ L i$, this value.

(Refer Slide Time: 02:45)

So we have defined all values there but for continuity the coefficients are constants, either plus 1 minus 1. So we will define it directly during the construction of the matrix. Now let us start with this channel reach thing. So channel reach starting is plus, ending is minus, flow depth is 1, flow discharge is 2, g is 9 point 81, g is global, yd is 3, Qd is 250, epsilon max this

is required for iteration, 1 into 10 to the power minus 6. Now junction number this is 3 and channel number this is 4.

So this is channel information matrix, junction information matrix, this is y_d minus Q_d . Minus Q_d is minus 250. This is junction connectivity and for each junction we need information for alpha. So for each channel we need (in) information about alpha. So in this case (fo) four alpha values are there.

(Refer Slide Time: 03:59)

```

5 endfunction
67 -----
68 // Channel Reach: Start + End -
69 // Flow Depth Condition: 1
70 // Flow Rate (Discharge) Condition: 2
71 -----
72 // Given Data
73 g=9.81; //m/s^2
74 global('g')
75 yd=3; //m
76 Qd=250; //m^3/s
77 eps_max=1e-6;
78 -----
79 junn=3;
80 chln=4;
81 -----
82 //----- Chl# | Length | Width | m1 | m2 | Segment | n | SO | JN1 | JN2 -----
83 chl_inf=[1 100 50 2 2 25 0.0120 0.0005 0 3
84          2 1500 30 2 2 75 0.0125 0.0004 3 2
85          3 500 20 2 2 25 0.0130 0.0012 3 2
86          4 100 40 2 2 25 0.0135 0.0005 2 1];
87 -----
88 jun_inf=[yd -Qd
89          -99999 -99999
90          -99999 -99999];
91 //0: Not Connected
92 //Positive Sign: 1st section of the 1-th channel rech is connected
93 //Negative Sign: Nl+1-th section if the 1-th channel rech is connected
94 jun_con=[1 -4 0 0
95          3 4 -3 -2
96          3 -1 2 3];
97 -----
98 alpha=[1 1 1 1];
99 //Derived Information
  
```

Now I can extract information from this channel information matrix. What is this second column? Second column is L_x , third column is B , fourth column is m_1 , fifth column is m_2 , sixth column is Δx , seventh column is n and S not is the eight column.

(Refer Slide Time: 04:35)

```

57 eps_max=1e-6;
58 //-----
59 junn=3;
60 chln=4;
61
62 ch1_inf=
63     100 50 2 2 25 0.0120 0.0005 0 3
64     1500 30 2 2 75 0.0125 0.0004 3 2
65     500 20 2 2 25 0.0130 0.0012 3 2
66     100 40 2 2 25 0.0135 0.0005 2 1];
67
68 jun_inf=[yd -qd
69         -99999 -99999
70         -99999 -99999];
71 //0: Not Connected
72 //Positive Sign: 1st section of the 1-th channel reach is connected
73 //Negative Sign: N+1-th section of the 1-th channel reach is connected
74 jun_con=[ -4 0 0
75           3 4 -3 -2
76           3 -1 2 3];
77
78 alpha=[1 1 1 1];
79
80 //Derived Information
81 Lx=ch1_inf(1:chln,2);
82 B=ch1_inf(1:chln,4);
83 m1=ch1_inf(1:chln,4);
84 m2=ch1_inf(1:chln,5);
85 delta_x=ch1_inf(1:chln,6);
86 n=ch1_inf(1:chln,7);
87 S0=ch1_inf(1:chln,8);
88
89 //Calculated
90 mnode=Lx./delta_x+1;
91

```

After extracting this thing we can calculate this mnode values. Mnode is Lx divided by delta x by point plus 1. We can add because it should be number of segments plus 1. Now we can define arbitrary values. This is yv, yv we means that y value. Qv means Q value. Yd into one sum mnode. Sum mnode considers all nodes for all channel lengths or channel reaches. Qv, this is for all channel reaches.

Gv, this is general variable. So in this one we will have both y and Q. So structure of gv is y1 Q1, y2 Q2, like that. So this is the general structure.

(Refer Slide Time: 05:54)

```

81 B=ch1_inf(1:chln,4);
82 m1=ch1_inf(1:chln,4);
83 m2=ch1_inf(1:chln,5);
84 delta_x=ch1_inf(1:chln,6);
85 n=ch1_inf(1:chln,7);
86 S0=ch1_inf(1:chln,8);
87
88 //Calculated
89 mnode=Lx./delta_x+1;
90
91 //-----Problem Dependent Parameters-----
92 yv=yd*ones(sum(mnode),1);
93 Qv=qd*ones(sum(mnode),1);
94
95 //General variable with y and Q
96 gv=zeros(2*sum(mnode),1);
97
98 //General Identification Matrix
99 idv=0;
100 for l=1:chln
101     for i=1:mnode(l)
102         idv=idv+1;
103         gid(l,i)=idv;
104     end
105 end
106
107 //Initial Value
108 for l=1:chln
109     for i=1:mnode(l)
110         gv(2*gid(l,i)-1)=yv(gid(l,i)); //y
111         gv(2*gid(l,i))=Qv(gid(l,i)); //Q
112     end
113 end
114

```

Now we need to have general identification information. What is this gid or general ID? General ID is the ID which is required for construction of our Jacobian matrix. For any

channel let us say this is channel number 1. We are starting with 1, $N+1$. So this is 1 comma, this is 1 comma $N+1$. For channel reach 1 we are starting from 1 to $N+1$. But what will be the global ID of this one? So what I have done, I have defined it starting from 1 to N .

(Refer Slide Time: 06:49)

```

81 h=ch1_inf(1:chn,7);
82 m1=ch1_inf(1:chn,4);
83 m2=ch1_inf(1:chn,5);
84 delta_x=ch1_inf(1:chn,6);
85 n=ch1_inf(1:chn,7);
86 S0=ch1_inf(1:chn,8);
87
88 //Calculated
89 mnode=Lx./delta_x+1;
90
91 //-----Problem Dependent Parameters-----
92 yv=yd*ones(sum(mnode),1);
93 Qv=Qd*ones(sum(mnode),1);
94
95 //General variable with y and Q
96 gv=zeros(2*sum(mnode),1);
97
98 //General Identification Matrix
99 idv=0;
100 for l=1:chn
101     for i=1:mnode(l)
102         idv=idv+1;
103         gid(l,i)=idv;
104     end
105 end
106
107 //Initial Value
108 for l=1:chn
109     for i=1:mnode(l)
110         gv(2*gid(l,i)-1)=yv(gid(l,i)); //y
111         gv(2*gid(l,i))=Qv(gid(l,i)); //Q
112     end
113 end
114
  
```

So next one is 2 1, this is 2 N2 plus 1. This is 3 1, 3 N3 plus 1. This is 4 1 or 4 N4 plus 1. So in this case I have defined this gid starting from this point. So my starting section is 1, then 2, and up to N1 plus 1 this will be my N1 plus 1. So for first one this is okay. Next one onwards this section which is the first section of the next channel reach this is N1 plus 2. That means we are adding from our previous thing. So this will be N1 plus 1 plus N2 plus 1, last one.

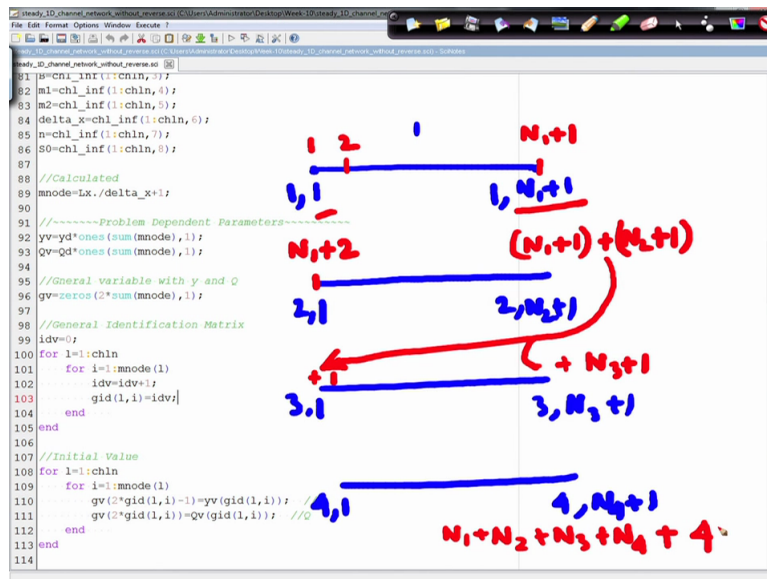
(Refer Slide Time: 08:06)

```

81 h=ch1_inf(1:chn,7);
82 m1=ch1_inf(1:chn,4);
83 m2=ch1_inf(1:chn,5);
84 delta_x=ch1_inf(1:chn,6);
85 n=ch1_inf(1:chn,7);
86 S0=ch1_inf(1:chn,8);
87
88 //Calculated
89 mnode=Lx./delta_x+1;
90
91 //-----Problem Dependent Parameters-----
92 yv=yd*ones(sum(mnode),1);
93 Qv=Qd*ones(sum(mnode),1);
94
95 //General variable with y and Q
96 gv=zeros(2*sum(mnode),1);
97
98 //General Identification Matrix
99 idv=0;
100 for l=1:chn
101     for i=1:mnode(l)
102         idv=idv+1;
103         gid(l,i)=idv;
104     end
105 end
106
107 //Initial Value
108 for l=1:chn
109     for i=1:mnode(l)
110         gv(2*gid(l,i)-1)=yv(gid(l,i)); //y
111         gv(2*gid(l,i))=Qv(gid(l,i)); //Q
112     end
113 end
114
  
```

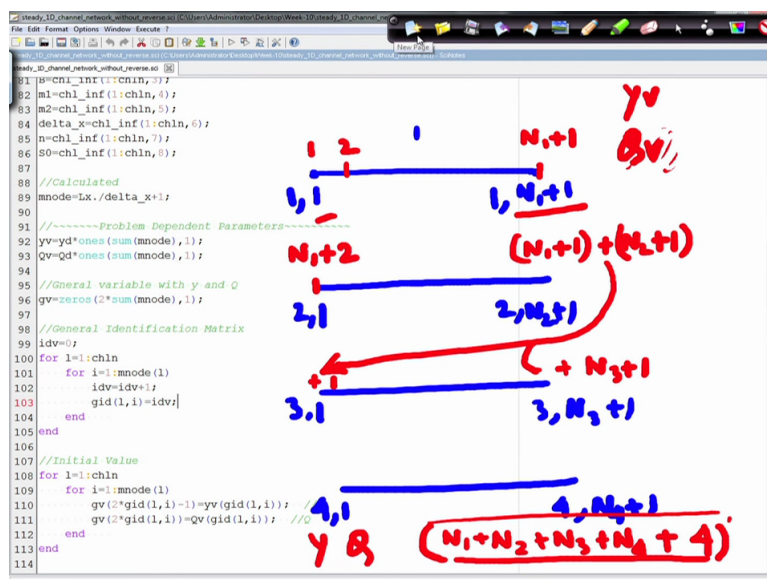
So same thing here this will be this one plus 1 plus this one plus N3 plus 1. Like that we can define a general number. So 1 to N1 plus 1 this is a local number. So we will have a general number starting from 1 up to N1 plus N2 plus N3 plus N4 plus 4.

(Refer Slide Time: 08:52)



So for this one we need this many y and Q values. So yv and Qv these two values are defined for these many variables.

(Refer Slide Time: 09:24)



Now for this one we can transfer the initial values. So global ID is gid. Gid if I multiply 2 into any gid L i. Gid L i means this is the global ID for Lth channel and ith section. So if I multiply 2 and subtract 1 this will give me the location for y. This is with gid L i. And (glob) this is general variable 2 into gid L i, this will give me this Q value for gid L i. So like that I can transfer the initial values here.

(Refer Slide Time: 10:48)

```

100 for i=1:CHIN
101     for l=1:mnode(1)
102         idv=idv+1;
103         gid(1,i)=idv;
104     end
105 end
106
107 //Initial Value
108 for l=1:chln
109     for i=1:mnode(1)
110         gv(2*gid(1,i)-1)=yv(gid(1,i)); //y
111         gv(2*gid(1,i))=Qv(gid(1,i)); //Q
112     end
113 end
114
115 //Derived Values
116 for l=1:chln
117     D1(l)=alpha(1)/(2*g);
118     D2(l)=(1/2)*n(1)^2*delta_x(1);
119 end
120 //
121 //
122
123 //Jacobian Matrix Size
124 A=zeros(2*sum(mnode),2*sum(mnode));
125 r=zeros(2*sum(mnode),1);
126
127 count = 0;
128 rmse=1;
129 //Space Loop
130 while rmse > eps_max
131     rmse=0;
132     eqn=0; //Equation Number
133

```

Handwritten equations in red:

$$gv(2 \times gid(1,i) - 1) = yv(gid(1,i))$$

$$gv(2 \times gid(1,i)) = Qv(gid(1,i))$$

Now after transferring these initial values I can calculate this D1 and D2. These values are required for calculation of the elements of Jacobian matrix. Next is construction of Jacobian matrix. So this is A and r these two are Jacobian matrix at right hand side and we are starting our calculation with counting, this count is zero. Rmse is 1. So rmse greater than epsilon max which is 1 into 10 to the power minus 6, rmse is equal to zero.

(Refer Slide Time: 11:30)

```

118     D2(l)=(1/2)*n(1)^2*delta_x(1);
119 end
120 //
121 //
122
123 //Jacobian Matrix Size
124 A=zeros(2*sum(mnode),2*sum(mnode));
125 r=zeros(2*sum(mnode),1);
126
127 count = 0;
128 rmse=1;
129 //Space Loop
130 while rmse > eps_max
131     rmse=0;
132     eqn=0; //Equation Number
133
134     //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
135     for l=1:chln
136         for i=1:mnode(1)-1
137             //Continuity
138             eqn=eqn+1;
139             A(eqn,2*gid(1,i)-1)=0; //y1
140             A(eqn,2*gid(1,i))=-1; //Q1
141             A(eqn,2*gid(1,i+1)-1)=0; //y1pl
142             A(eqn,2*gid(1,i+1))=1; //Q1pl
143             r(eqn)=0;
144             //Momentum
145             eqn=eqn+1;
146             A(eqn,2*gid(1,i)-1)=dMdy1(yv(gid(1,i)),Qv(gid(1,i)),D1(l),D2(l),B(l),m1(l),m2(l)); //y1
147             A(eqn,2*gid(1,i))=dMQ1(yv(gid(1,i)),Qv(gid(1,i)),D1(l),D2(l),B(l),m1(l),m2(l)); //Q1
148             A(eqn,2*gid(1,i+1)-1)=dMdy1pl(yv(gid(1,i+1)),Qv(gid(1,i+1)),D1(l),D2(l),B(l),m1(l),m2(l)); //y1pl
149             A(eqn,2*gid(1,i+1))=dMQ1pl(yv(gid(1,i+1)),Qv(gid(1,i+1)),D1(l),D2(l),B(l),m1(l),m2(l)); //Q1pl

```

After entering into this loop I am considering rmse is equal to zero and I am starting with equation number 0. Now when I am starting the calculation here so I am considering that for

each channel segment, so channel segment will be one number less because it is for NL and mnode is equivalent to NL plus 1.

(Refer Slide Time: 12:10)

```

130 while rmse > eps_max
131     rmse=0;
132     eqn=0; //Equation Number
133
134     //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
135     for l=1:chnl
136         for i=1:mnode(l)-1
137             //Continuity
138             eqn=eqn+1;
139             A(eqn,2*gid(l,i)-1)=0; //yi
140             A(eqn,2*gid(l,i))=-1; //qi
141             A(eqn,2*gid(l,i+1)-1)=0; //yipl
142             A(eqn,2*gid(l,i+1))=1; //qip1
143             r(eqn)=0;
144             //Momentum
145             eqn=eqn+1;
146             A(eqn,2*gid(l,i)-1)=dmdy1(yv(gid(l,i)),Qv(gid(l,i)),D1(l),D2(l),B(l),m1(l),m2(l)); //yi
147             A(eqn,2*gid(l,i))=dmdqi(yv(gid(l,i)),Qv(gid(l,i)),D1(l),D2(l),B(l),m1(l),m2(l)); //qi
148             A(eqn,2*gid(l,i+1)-1)=dmdyipl(yv(gid(l,i+1)),Qv(gid(l,i+1)),D1(l),D2(l),B(l),m1(l),m2(l)); //yipl
149             A(eqn,2*gid(l,i+1))=dmdqip1(yv(gid(l,i+1)),Qv(gid(l,i+1)),D1(l),D2(l),B(l),m1(l),m2(l)); //qip1
150             r(eqn)=-Mll(yv(gid(l,i)),Qv(gid(l,i)),yv(gid(l,i+1)),Qv(gid(l,i+1)),s0(l),delta_x(l),D1(l),D2(l),
151             B(l),m1(l),m2(l));
152         end
153     end
154     // Junction Continuity Condition JUN=3
155     for j=1:junn
156         eqn=eqn+1;
157         if(jun_inf(j,2) <> -99999) then
158             r(eqn)=jun_inf(j,2);
159         else
160             r(eqn)=0;

```

So we are considering mnode minus 1. So channel 1 to chnl channel number and 1 to mnode minus 1 for first continuity equation I am adding eqn equals to eqn plus 1. That means equation number, I am increasing the equation number. This is for momentum. So first continuity next momentum like that for all segments I can write these equations.

(Refer Slide Time: 12:42)

```

130 while rmse > eps_max
131     rmse=0;
132     eqn=0; //Equation Number
133
134     //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
135     for l=1:chnl
136         for i=1:mnode(l)-1
137             //Continuity
138             eqn=eqn+1;
139             A(eqn,2*gid(l,i)-1)=0; //yi
140             A(eqn,2*gid(l,i))=-1; //qi
141             A(eqn,2*gid(l,i+1)-1)=0; //yipl
142             A(eqn,2*gid(l,i+1))=1; //qip1
143             r(eqn)=0;
144             //Momentum
145             eqn=eqn+1;
146             A(eqn,2*gid(l,i)-1)=dmdy1(yv(gid(l,i)),Qv(gid(l,i)),D1(l),D2(l),B(l),m1(l),m2(l)); //yi
147             A(eqn,2*gid(l,i))=dmdqi(yv(gid(l,i)),Qv(gid(l,i)),D1(l),D2(l),B(l),m1(l),m2(l)); //qi
148             A(eqn,2*gid(l,i+1)-1)=dmdyipl(yv(gid(l,i+1)),Qv(gid(l,i+1)),D1(l),D2(l),B(l),m1(l),m2(l)); //yipl
149             A(eqn,2*gid(l,i+1))=dmdqip1(yv(gid(l,i+1)),Qv(gid(l,i+1)),D1(l),D2(l),B(l),m1(l),m2(l)); //qip1
150             r(eqn)=-Mll(yv(gid(l,i)),Qv(gid(l,i)),yv(gid(l,i+1)),Qv(gid(l,i+1)),s0(l),delta_x(l),D1(l),D2(l),
151             B(l),m1(l),m2(l));
152         end
153     end
154     // Junction Continuity Condition JUN=3
155     for j=1:junn
156         eqn=eqn+1;
157         if(jun_inf(j,2) <> -99999) then
158             r(eqn)=jun_inf(j,2);
159         else
160             r(eqn)=0;

```

Next part is junction continuity. What is this junction continuity? J equals to junction number, 1 to junction number. Eqn equals to equation 1 because we will have one continuity equation each for this junctions. This is junction information j2. What is this J2? 1 is related to y and 2 that is related to Q. So if this is not equal to minus 999 five 9s then we should start with r eqn equals to junction information j2. That means we can directly specify that Q value there.

(Refer Slide Time: 13:36)

```

150 r(eqn)=-M1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),S0(1),delta_x(1),D1(1),D2(1),
151 B(1),m1(1),m2(1));
152 end
153 end
154 // Junction Continuity Condition JUN=3
155 for j=1:junn
156   eqn=eqn+1;
157   if(jun_inf(j,2) <> -99999) then
158     r(eqn)=jun_inf(j,2);
159   else
160     r(eqn)=0;
161   end
162 // jun_con=1 -4 0 0
163 // jun_con=3 4 -3 -2
164 // jun_con=3 -1 2 3;
165   for l=1:jun_con(j,1)
166     if(abs(jun_con(j,l+1)) > eps_max) then
167       if(jun_con(j,l+1) > 0) then
168         jn_node=1;
169         A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=-1;
170         r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,l+1)),jn_node));
171       end
172       if(jun_con(j,l+1) < 0) then
173         jn_node=mnode(abs(jun_con(j,l+1)));
174         A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=1;
175         r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,l+1)),jn_node));
176       end
177     end
178   end
179   r(eqn)=-r(eqn);
180 end
181
182

```

Whatever amount is added or extracted there or else we can start from r eqn equals to zero because we are not adding any quantity there. So next part is starting from L 1 to junction continuity j1. So first column provides us information about the number of channel sections connected to a particular junction.

So starting from 1 to junction condition or junction conditions j1 we have to check whether this junction condition jL plus 1 that means the next one if it is absolute value is greater than epsilon max because if it is zero that means we do not have any connectivity there. If it is greater than zero or small number then only we should proceed. If this junction continuity is greater than zero that means we are starting from j node is starting node from 1.

(Refer Slide Time: 15:02)

```

161     end
162     //----- jun_con=(1 -4 0 0
163     //----- 3 4 -3 -2
164     //----- 3 -1 2 3);
165     for l=1:jun_con(j,1)
166         if(abs(jun_con(j,1+l)) > eps_max) then
167             if(jun_con(j,1+l) > 0) then
168                 jn_node=jn_node+1;
169                 A(eqn,2*gid(abs(jun_con(j,1+l)),jn_node))=-1;
170                 r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,1+l)),jn_node));
171             end
172             if(jun_con(j,1+l) < 0) then
173                 jn_node=mnode(abs(jun_con(j,1+l)));
174                 A(eqn,2*gid(abs(jun_con(j,1+l)),jn_node))=1;
175                 r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,1+l)),jn_node));
176             end
177         end
178     end
179     r(eqn)=-r(eqn);
180 end
181
182
183
184
185 //Junction Energy Condition
186 for j=1:junn
187     //----- jun_con=(1 -4 0 0
188     //----- 3 4 -3 -2
189     //----- 3 -1 2 3);
190     if(jun_inf(j,1) <> -99999) then
191         eqn=eqn+1;
192     end
193     if(jun_con(j,2) > 0) then jn_node=1; end
194     if(jun_con(j,2) < 0) then jn_node=mnode(abs(jun_con(j,2))); end

```

So we can write this equation for this L. L is again absolute value of junction continuity jL plus 1 and $jnode$. This is minus 1 because at the starting node it should be minus. And this r eqn equals to r eqn minus this Q value because we are considering that Q value is positive here. Now no reversal of flow.

(Refer Slide Time: 15:32)

```

161     end
162     //----- jun_con=(1 -4 0 0
163     //----- 3 4 -3 -2
164     //----- 3 -1 2 3);
165     for l=1:jun_con(j,1)
166         if(abs(jun_con(j,1+l)) > eps_max) then
167             if(jun_con(j,1+l) > 0) then
168                 jn_node=jn_node+1;
169                 A(eqn,2*gid(abs(jun_con(j,1+l)),jn_node))=-1;
170                 r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,1+l)),jn_node));
171             end
172             if(jun_con(j,1+l) < 0) then
173                 jn_node=mnode(abs(jun_con(j,1+l)));
174                 A(eqn,2*gid(abs(jun_con(j,1+l)),jn_node))=1;
175                 r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,1+l)),jn_node));
176             end
177         end
178     end
179     r(eqn)=-r(eqn);
180 end
181
182
183
184
185 //Junction Energy Condition
186 for j=1:junn
187     //----- jun_con=(1 -4 0 0
188     //----- 3 4 -3 -2
189     //----- 3 -1 2 3);
190     if(jun_inf(j,1) <> -99999) then
191         eqn=eqn+1;
192     end
193     if(jun_con(j,2) > 0) then jn_node=1; end
194     if(jun_con(j,2) < 0) then jn_node=mnode(abs(jun_con(j,2))); end

```

So this is directly subtracted from r eqn or junction continuity less than zero. That means this is terminating section. Then jn node equal to (m) $mnode$ equal to or absolute value of junction continuity jL plus 1. This will provide us information about the channel reach.

(Refer Slide Time: 16:00)

```

161     end
162     //----- jun_con=1 -4 0 0
163     //----- 3 4 -3 -2
164     //----- 3 -1 2 3};
165     for l=1:jun_con(j,1)
166         if(abs(jun_con(j,1+l)) > eps_max) then
167             if(jun_con(j,1+l) > 0) then
168                 jn_node=l;
169                 A(eqn,2*gid(abs(jun_con(j,1+l)),jn_node))=-1;
170                 r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,1+l)),jn_node));
171             end
172             if(jun_con(j,1+l) < 0) then
173                 jn_node=mnode(abs(jun_con(j,1+l)));
174                 A(eqn,2*gid(abs(jun_con(j,1+l)),jn_node))=1;
175                 r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,1+l)),jn_node));
176             end
177         end
178     end
179     r(eqn)=-r(eqn);
180 end
181
182
183
184
185 //Junction Energy Condition
186 for j=1:junn
187     //----- jun_con=1 -4 0 0
188     //----- 3 4 -3 -2
189     //----- 3 -1 2 3};
190     if(jun_inf(j,1) <> -99999) then
191         eqn=eqn+1;
192
193         if(jun_con(j,2) > 0) then jn_node1=1; end
194         if(jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
195         A(eqn,2*gid(abs(jun_con(j,2)),jn_node1))=1;
196         r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1))-jun_inf(j,1);
197     end
198     r(eqn)=r(eqn);
199     if(jun_con(j,1) > 1) then
200         for l=1:jun_con(j,1)-1
201             eqn=eqn+1;
202             if(jun_con(j,2) > 0) then jn_node1=1; end
203             if(jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
204             A(eqn,2*gid(abs(jun_con(j,2)),jn_node1))=1;
205             r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1));
206         end
207         if(jun_con(j,1+2) > 0) then jn_node2=1; end
208         if(jun_con(j,1+2) < 0) then jn_node2=mnode(abs(jun_con(j,1+2))); end
209         A(eqn,2*gid(abs(jun_con(j,1+2)),jn_node2))=1;
210         r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
211     end
212     r(eqn)=-r(eqn);
213 end

```

Then I can write this code there and finally r eqn should be minus r eqn because minus of the function value should be there. So with this our next part is junction condition or energy condition. What should be there? For junction 1 to junction number and if junction information j_1 which is not equal to minus 999 so we should start or increasing our equation number and first node is the value which is there in the second column.

And we can directly provide the information there that value in the second column minus the value which is directly provided at the downstream junction point or upstream junction point depending on the junction condition. This is r eqn equals to minus r eqn.

(Refer Slide Time: 17:12)

```

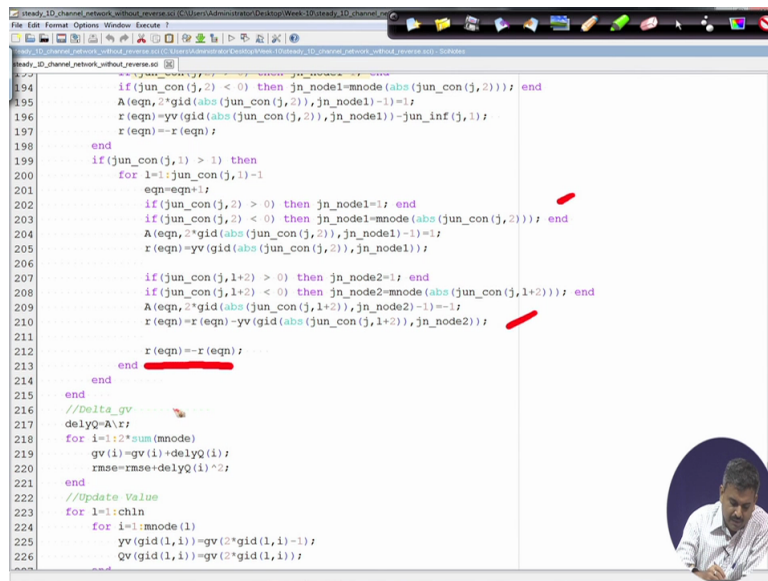
180     end
181
182
183
184
185 //Junction Energy Condition
186 for j=1:junn
187     //----- jun_con=1 -4 0 0
188     //----- 3 4 -3 -2
189     //----- 3 -1 2 3};
190     if(jun_inf(j,1) <> -99999) then
191         eqn=eqn+1;
192
193         if(jun_con(j,2) > 0) then jn_node1=1; end
194         if(jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
195         A(eqn,2*gid(abs(jun_con(j,2)),jn_node1))=1;
196         r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1))-jun_inf(j,1);
197     end
198     r(eqn)=r(eqn);
199     if(jun_con(j,1) > 1) then
200         for l=1:jun_con(j,1)-1
201             eqn=eqn+1;
202             if(jun_con(j,2) > 0) then jn_node1=1; end
203             if(jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
204             A(eqn,2*gid(abs(jun_con(j,2)),jn_node1))=1;
205             r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1));
206         end
207         if(jun_con(j,1+2) > 0) then jn_node2=1; end
208         if(jun_con(j,1+2) < 0) then jn_node2=mnode(abs(jun_con(j,1+2))); end
209         A(eqn,2*gid(abs(jun_con(j,1+2)),jn_node2))=1;
210         r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
211     end
212     r(eqn)=-r(eqn);
213 end

```

Now if this is the condition which will be there with 1 section or 1 channel. Now if you have multiple channels then we can utilise this junction connectivity 1. This is j_1 which is greater than 1. If it is 1 then obviously this will be calculated with the first condition.

If connectivity is greater than 1, this connectivity number then L is equal to 1 to junction connectivity 1 minus 1 we can identify the starting and terminating nodes and for that one we can specify the derivatives or elements of the Jacobian matrix. And finally r eqn equals to minus r eqn there.

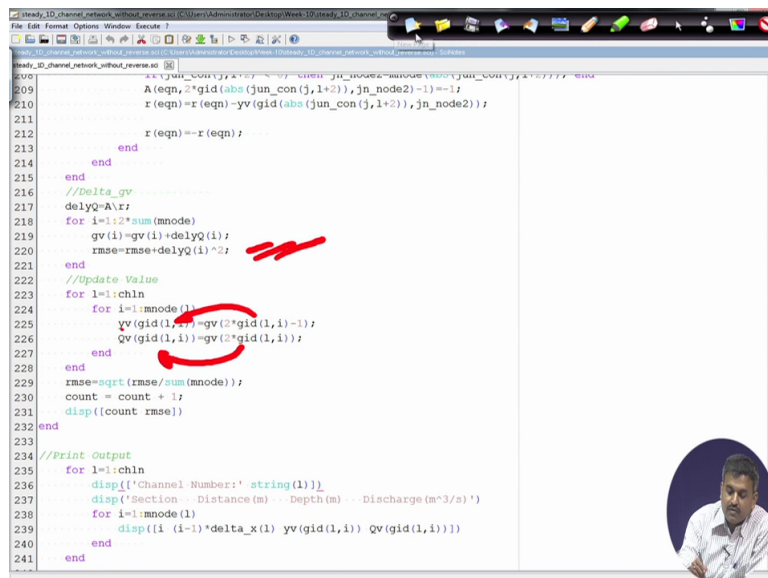
(Refer Slide Time: 18:20)



```
194     if (jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
195     A(eqn,2*gid(abs(jun_con(j,2)),jn_node1)-1)=1;
196     r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1))-jun_inf(j,1);
197     r(eqn)=-r(eqn);
198     end
199     if (jun_con(j,1) > 1) then
200     for l=1:jun_con(j,1)-1
201     eqn=eqn+1;
202     if (jun_con(j,2) > 0) then jn_node1=1; end
203     if (jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
204     A(eqn,2*gid(abs(jun_con(j,2)),jn_node1)-1)=1;
205     r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1));
206     r(eqn)=-r(eqn);
207     if (jun_con(j,1+2) > 0) then jn_node2=1; end
208     if (jun_con(j,1+2) < 0) then jn_node2=mnode(abs(jun_con(j,1+2))); end
209     A(eqn,2*gid(abs(jun_con(j,1+2)),jn_node2)-1)=1;
210     r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
211     r(eqn)=-r(eqn);
212     end
213     end
214     end
215     //Delta gv
216     delyQ=A\r;
217     for i=1:2*sum(mnode)
218     gv(i)=gv(i)+delyQ(i);
219     rmse=rmse+delyQ(i)^2;
220     end
221     //Update Value
222     for l=1:chln
223     for i=1:mnode(l)
224     yv(gid(l,i))=gv(2*gid(l,i)-1);
225     Qv(gid(l,i))=gv(2*gid(l,i));
226     end
```

And finally we can calculate this delta yQ. After calculation of this delta yQ we can update gv value and simultaneously we can also calculate this rmse value. And after calculation of gv again we need to transfer these values directly for yv and Qv because during calculation we are directly utilising this yv and Q. From gv to yv Qv this mapping we can directly use and we can directly transfer these values.

(Refer Slide Time: 19:07)



```
209     A(eqn,2*gid(abs(jun_con(j,1+2)),jn_node2)-1)=1;
210     r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
211     r(eqn)=-r(eqn);
212     end
213     end
214     end
215     //Delta gv
216     delyQ=A\r;
217     for i=1:2*sum(mnode)
218     gv(i)=gv(i)+delyQ(i);
219     rmse=rmse+delyQ(i)^2;
220     end
221     //Update Value
222     for l=1:chln
223     for i=1:mnode(l)
224     yv(gid(l,i))=gv(2*gid(l,i)-1);
225     Qv(gid(l,i))=gv(2*gid(l,i));
226     end
227     end
228     rmse=sqrt(rmse/sum(mnode));
229     count = count + 1;
230     disp([count rmse])
231 end
232 //Print output
233 for l=1:chln
234     disp(['Channel Number: ' string(l)])
235     disp(['Section -- Distance (m) -- Depth (m) -- Discharge (m^3/s)'])
236     for i=1:mnode(l)
237         disp([i (i-1)*delta_x(1) yv(gid(l,i)) Qv(gid(l,i))])
238     end
239 end
240 end
241 end
```

And we can stop based on the convergence criteria. And final is the output. This is for 1 to channel number, display the channel number and display section, distance, depth, discharge, this for i equals to 1 to mnode L. That means we are considering all sections. I i minus del x,

this will provide us the distance at which the section is there. Next one is depth, this will provide us the discharge.

(Refer Slide Time: 19:54)

```

211
212         r(eqn)=-r(eqn);
213     end
214 end
215 end
216 //Delta_gv
217 delyQ=A*r;
218 for i=1:2*sum(mnode)
219     gv(i)=gv(i)+delyQ(i);
220     rmse=rmse+delyQ(i)^2;
221 end
222 //Update Value
223 for l=1:chln
224     for i=1:mnode(l)
225         yv(gid(l,i))-gv(2*gid(l,i)-1);
226         Qv(gid(l,i))-gv(2*gid(l,i));
227     end
228 end
229 rmse=sqrt(rmse/sum(mnode));
230 count = count + 1;
231 disp([count rmse]);
232 end
233 //Print Output
234 for l=1:chln
235     disp(['Channel Number: ' string(l)])
236     disp(['Section -- Distance (m) -- Depth (m) -- Discharge (m^3/s)'])
237     for i=1:mnode(l)
238         disp([i (i-1)*delta_x(l) yv(gid(l,i)) Qv(gid(l,i))])
239     end
240 end
241
242
243
244

```

So if I run this one, now after iteration what I am getting? This is for channel number 1, this is section 1, 2, 3, 4, 5. This is distance zero, 25, 50, 75, 100. These are depth values and this is discharge. Obviously this discharge we have not directly provided at distance zero which is our entry point or inlet point.

(Refer Slide Time: 20:35)

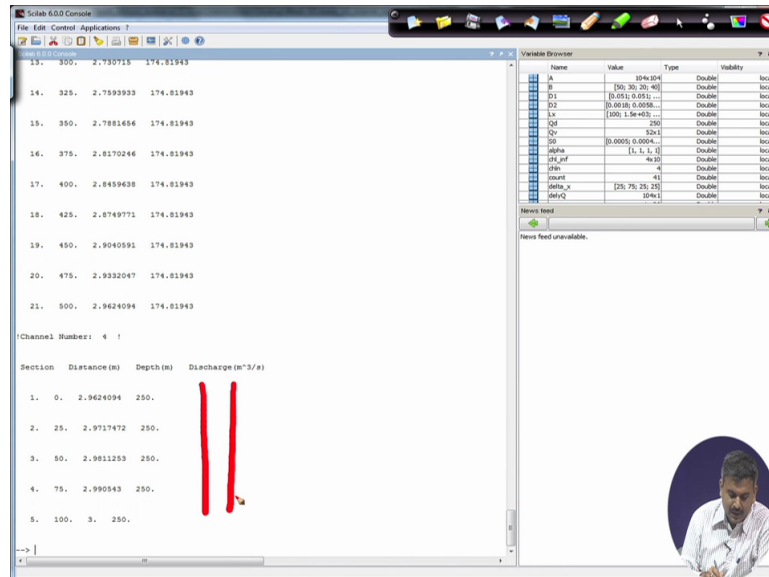
```

--> //Print Output
-->
--> for l=1:chln
-->     disp(['Channel Number: ' string(l)])
-->     disp(['Section -- Distance(m) -- Depth(m) -- Discharge (m^3/s)'])
-->     for i=1:mnode(l)
-->         disp([i (i-1)*delta_x(l) yv(gid(l,i)) Qv(gid(l,i))])
-->     end
-->
Channel Number: 1 !
Section Distance(m) Depth(m) Discharge (m^3/s)
1. 0. 2.3606113 250.
2. 25. 2.3698457 250.
3. 50. 2.3791341 250.
4. 75. 2.3884758 250.
5. 100. 2.3978697 250.
Channel Number: 2 !
Section Distance(m) Depth(m) Discharge (m^3/s)
1. 0. 2.3978697 75.180566
2. 75. 2.4253933 75.180566
3. 150. 2.4529207 75.180566
4. 225. 2.4805949 75.180566

```

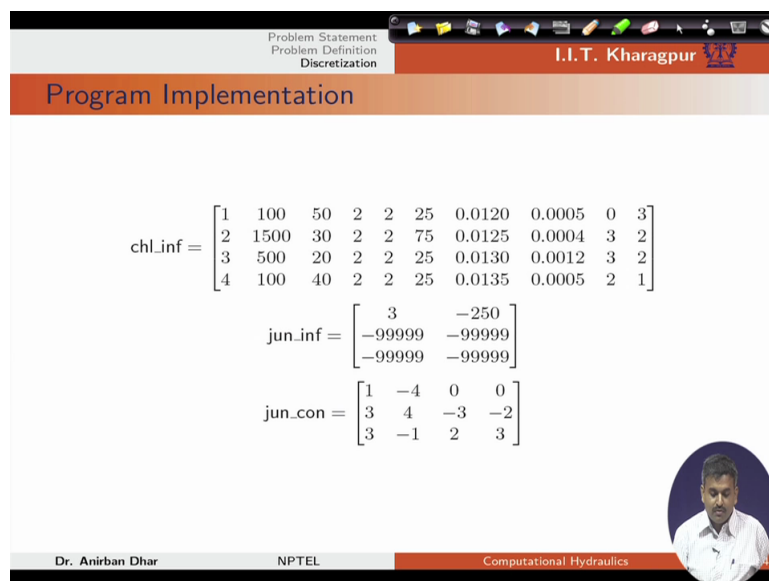

So indirectly we are getting this 250 metre cube per second value. Similarly for other sections, channel number 2 section starting from 1 to 21 and channel 3 starting from 1 to 21 and channel flow starting from 1 to 5, we are getting these values.

(Refer Slide Time: 21:23)



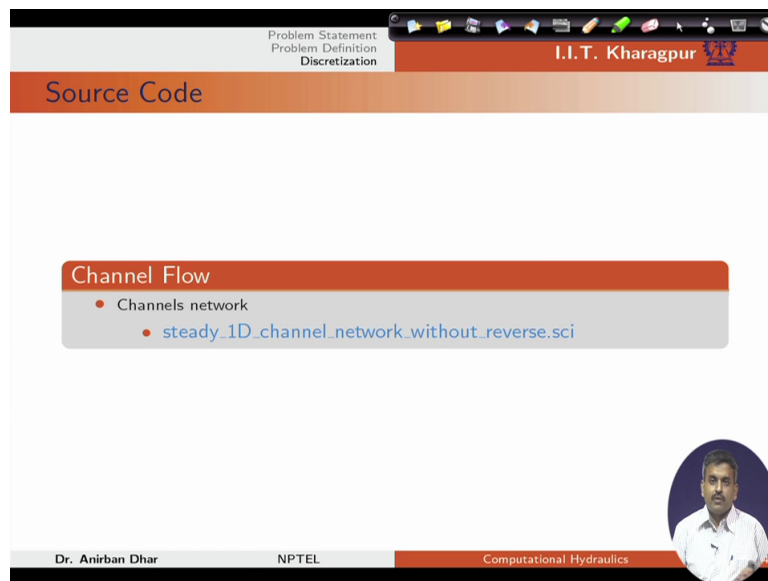
Now these values we have got without providing the initial guess by satisfying the continuity equation. Now so these values are directly coming from the iteration process with arbitrary initial values.

(Refer Slide Time: 21:49)



Now this code is steady 1D channel network without reverse.

(Refer Slide Time: 21:59)



The screenshot shows a presentation slide with a dark blue header and footer. The header contains the text "Problem Statement", "Problem Definition", and "Discretization" on the left, and "I.I.T. Kharagpur" with a logo on the right. The main content area has a dark blue bar at the top with the text "Source Code". Below this, there is a white box with a dark blue header "Channel Flow" and a list of items: "Channels network" and "steady_1D_channel_network_without_reverse.sci". The footer contains "Dr. Anirban Dhar", "NPTEL", and "Computational Hydraulics" on the left, and a circular portrait of a man on the right.

And in the next lecture we will be talking about the steady 1D channel network with reverse flow situation. Thank you.