

Computational Hydraulics
Professor Anirban Dhar
Department of Civil Engineering
Indian Institute of Technology Kharagpur
Lecture 45
Unsteady 1D Channel Flow (Contd.)

So in this case let us start this program with clc which is clear console, clear, clearing all variables. First one is area calculation. Area as a function of y, b, m1. B, m1, m2 these values are fixed values for any section. Only varying thing is y. This is dA by dy. Third one this is hR, hydraulic radius. And fourth function this is dRh divided by dy. Fifth function this is CL i N N plus 1.

(Refer Slide Time: 01:22)

```

1 clc
2 clear
3 //-----
4 function AV=area(y,B,m1,m2)
5     AV=B*y+(1/2)*(m1+m2)*y^2;
6 endfunction
7 function dAV=darea(y,B,m1,m2)
8     dAV=(m1+m2)*y;
9 endfunction
10 function RV=RV(y,B,m1,m2)
11     RV=(B*y+(1/2)*(m1+m2)*y^2)/(B+(sqrt(1+m1^2)+sqrt(1+m2^2))*y);
12 endfunction
13 function dRV=dRV(y,B,m1,m2)
14     Tw=(m1+m2)*y;
15     Fm=(sqrt(1+m1^2)+sqrt(1+m2^2))*y;
16     Rh=RV(y,B,m1,m2);
17     dRdy=(sqrt(1+m1^2)+sqrt(1+m2^2));
18     dRV=(Tw/Fm)-(Rh/Fm)*dRdy;
19 endfunction
20 //-----
21 function CLi=CLi(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha1,alpha2,B,m1,m2,nu)
22     term1=area(y2,B,m1,m2)-area(y20,B,m1,m2);
23     term2=area(y1,B,m1,m2)-area(y10,B,m1,m2);
24     term3=Q2-Q1;
25     term4=Q20-Q10;
26     CLi=(psi/delta_t)*term1+((1-psi)/delta_t)*term2+(theta/delta_x)*term3+((1-theta)/delta_x)*term4;
27 endfunction
28 //-----
29 function dcdy=dcdy(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha1,alpha2,B,m1,m2,nu)
30     dcdy=((1-psi)/delta_t)*darea(y1,B,m1,m2);
31 endfunction
32 //-----
33 function dcdydy=dcdydy(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha1,alpha2,B,m1,m2,nu)

```

Handwritten annotations in red on the code include:
- $A(y, B, m_1, m_2)$ next to the area function.
- $\frac{dA}{dy}$ next to the darea function.
- R_h next to the RV function.
- $\frac{dR_h}{dy}$ next to the dRV function.
- CL_i next to the CLi function.

Then comes our function which is dc or dc by dy i. This is dc by dy i plus 1 and next one is dc by dQ, this is i. Next one is dc by dQ i plus 1. So we have defined CL i without specifying this L i here I have written this. This is dc L i, y i, dc y i plus 1, dc dQ i, dc dQ i plus 1.

(Refer Slide Time: 02:44)

```

20
21
22 function C1iv=C1i(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
23     term1=arcxav(y2,B,m1,m2)-arcxav(y20,B,m1,m2);
24     term2=arcxav(y1,B,m1,m2)-arcxav(y10,B,m1,m2);
25     term3=Q2-Q1;
26     term4=Q20-Q10;
27
28     C1iv=(psi/delta_t)*term1+((1-psi)/delta_t)*term2+(theta/delta_x)*term3+((1-theta)/delta_x)*term4;
29 endfunction
30
31 function dCdyiv=dCdyi(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
32     dCdyiv=((1-psi)/delta_t)*darcxav(y1,B,m1,m2);
33 endfunction
34
35 function dCdyipiv=dCdyipi(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
36     dCdyipiv=(psi/delta_t)*darcxav(y2,B,m1,m2);
37 endfunction
38
39 function dCQ1iv=dCQ1i(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
40     dCQ1iv=-theta/delta_x;
41 endfunction
42
43 function dCQ1ipiv=dCQ1ipi(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
44     dCQ1ipiv=theta/delta_x;
45 endfunction
46
47
48
49
50
51 function M1iv=M1i(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
52     Av1=arcxav(y1,B,m1,m2);
53     Av2=arcxav(y2,B,m1,m2);
54     Av10=arcxav(y10,B,m1,m2);
55     Av20=arcxav(y20,B,m1,m2);
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Now within this again we need to specify this ML i. This is ML i N N plus 1. Next is specification of dM by dy i. This is dm by dy i plus 1.

(Refer Slide Time: 03:21)

```

69
70
71 function dMdyiv=dMdyi(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
72     Av1=arcxav(y1,B,m1,m2);
73     Rh1=HRV(y1,B,m1,m2);
74     dAv1=darcxav(y1,B,m1,m2);
75     dRh1=dHRV(y1,B,m1,m2);
76
77     term1=(Q1/Av1^2)*dAv1;
78     term2=(Q1^2/Av1^3)*dAv1;
79     term3=theta*g/delta_x;
80     term4=2*Q1*abs(Q1)*dAv1*Rh1^(-4/3)*Av1^(-3);
81     term42=(4/3)*Q1*abs(Q1)*dRh1*Rh1^(-7/3)*Av1^(-2);
82
83     dMdyiv=((1-psi)/delta_t)*term1+(theta*alpha/delta_x)*term2-term3-theta*(1-psi)*g*nm^2*(term4+term42);
84 endfunction
85
86 function dMdyipiv=dMdyipi(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
87     Av2=arcxav(y2,B,m1,m2);
88     Rh2=HRV(y2,B,m1,m2);
89     dAv2=darcxav(y2,B,m1,m2);
90     dRh2=dHRV(y2,B,m1,m2);
91
92     term1=(Q2/Av2^2)*dAv2;
93     term2=(Q2^2/Av2^3)*dAv2;
94     term3=theta*g/delta_x;
95     term4=2*Q2*abs(Q2)*dAv2*Rh2^(-4/3)*Av2^(-3);
96     term42=(4/3)*Q2*abs(Q2)*dRh2*Rh2^(-7/3)*Av2^(-2);
97
98     dMdyipiv=(psi/delta_t)*term1-(theta*alpha2/delta_x)*term2+term3-theta*psi*g*nm^2*(term4+term42);
99 endfunction
100
101 function dMQ1iv=dMQ1i(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
102     Av1=arcxav(y1,B,m1,m2);
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

After this calculation interestingly please note here that we have used this Q abs terms here, Q abs terms here.

(Refer Slide Time: 03:38)

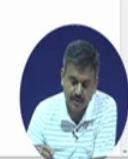
```
2 Av1=areaav(y1,B,m1,m2);
3 Rh1=HRV(y1,B,m1,m2);
4 dAv1=dareaav(y1,B,m1,m2);
5 dRh1=dHRV(y1,B,m1,m2);
6
7 term1=(Q1/Av1^2)*dAv1;
8 term2=(Q1^2/Av1^3)*dAv1;
9 term3=theta*Q1^2/delta_x;
10 term4=2*Q1*abs(Q1)*dRh1*(-4/3)*Av1^(-3);
11 term42=(4/3)*Q1*abs(Q1)*dRh1*Rh1^(-7/3)*Av1^(-2);
12
13 ddyipiv=((1-psi)/delta_t)*term1+(theta*alpha/delta_x)*term2-term3-theta*(1-psi)*g*nm^2*(term4+term42);
14 endfunction
04
1 function ddyipiv=ddyipiv(y1,Q1,Q2,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,
2 nm)
3 Av2=areaav(y2,B,m1,m2);
4 Rh2=HRV(y2,B,m1,m2);
5 dAv2=dareaav(y2,B,m1,m2);
6 dRh2=dHRV(y2,B,m1,m2);
7
8 term1=(Q2/Av2^2)*dAv2;
9 term2=(Q2^2/Av2^3)*dAv2;
10 term3=theta*Q2^2/delta_x;
11 term4=2*Q2*abs(Q2)*dAv2*(-4/3)*Av2^(-3);
12 term42=(4/3)*Q2*abs(Q2)*dRh2*Rh2^(-7/3)*Av2^(-2);
13
14 ddyipiv=(psi/delta_t)*term1-(theta*alpha2/delta_x)*term2+term3-theta*psi*g*nm^2*(term4+term42);
15 endfunction
99
1 function dmqiv=dmqiv(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
2 Av1=areaav(y1,B,m1,m2);
3 Rh1=HRV(y1,B,m1,m2);
4 term1=Av1^(-1);
5 term2=Q1/Av1^2;
6 term3=abs(Q1)*Rh1^(-4/3)*Av1^(-2);
7
8 dmqiv=((1-psi)/delta_t)*term1-(theta*alpha/delta_x)*term2+2*theta*(1-psi)*g*nm^2*term3;
9 endfunction
1
1 function dmqipiv=dmqipiv(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,
2 nm)
3 Av2=areaav(y2,B,m1,m2);
4 Rh2=HRV(y2,B,m1,m2);
5 term1=Av2^(-1);
6 term2=Q2/Av2^2;
7 term3=abs(Q2)*Rh2^(-4/3)*Av2^(-2);
8
9 dmqipiv=(psi/delta_t)*term1+(theta*alpha2/delta_x)*term2+term3-theta*psi*g*nm^2*term3;
10 endfunction
110
1 function bndv=bndcon(typ,jnum,tv)
2 if(typ==1 & jnum==3) then
3 bndv=1.4;
4 end
5 if(typ==2 & jnum==1) then
6 if(tv < 2000) then
7 bndv=50+(100/2000)*tv
8 end
9 end
```

These two are Q this is dM by dQ i and this is dM by dQ i plus 1.

(Refer Slide Time: 04:00)

```
11 term42=(4/3)*Q2*abs(Q2)*dRh2*Rh2^(-7/3)*Av2^(-2);
12
13 ddyipiv=(psi/delta_t)*term1-(theta*alpha2/delta_x)*term2+term3-theta*psi*g*nm^2*(term4+term42);
14 endfunction
99
1 function dmqiv=dmqiv(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,nm)
2 Av1=areaav(y1,B,m1,m2);
3 Rh1=HRV(y1,B,m1,m2);
4 term1=Av1^(-1);
5 term2=Q1/Av1^2;
6 term3=abs(Q1)*Rh1^(-4/3)*Av1^(-2);
7
8 dmqiv=((1-psi)/delta_t)*term1-(theta*alpha/delta_x)*term2+2*theta*(1-psi)*g*nm^2*term3;
9 endfunction
1
1 function dmqipiv=dmqipiv(y1,Q1,y2,Q2,y10,Q10,y20,Q20,zv1,zv2,theta,psi,delta_t,delta_x,alpha,alpha2,B,m1,m2,
2 nm)
3 Av2=areaav(y2,B,m1,m2);
4 Rh2=HRV(y2,B,m1,m2);
5 term1=Av2^(-1);
6 term2=Q2/Av2^2;
7 term3=abs(Q2)*Rh2^(-4/3)*Av2^(-2);
8
9 dmqipiv=(psi/delta_t)*term1+(theta*alpha2/delta_x)*term2+term3-theta*psi*g*nm^2*term3;
10 endfunction
110
1 function bndv=bndcon(typ,jnum,tv)
2 if(typ==1 & jnum==3) then
3 bndv=1.4;
4 end
5 if(typ==2 & jnum==1) then
6 if(tv < 2000) then
7 bndv=50+(100/2000)*tv
8 end
9 end
```

मल
जु
हिति



Now after specifying this we need to specify the boundary conditions. So what is this boundary condition function? This is boundary value function. This is boundary condition bndcon. This is type. If type is 1 so we will specify the depth. If type is 2 then we will specify discharge. Then junction number, depending on the junction number we can specify and tv or time value is required in this case.

(Refer Slide Time: 04:34)

```
8      dtdqpi=(psi/delta_t)*term1+(theta*alpha2/delta_x)*term2+2*theta*psi*g*nm**2*term3;
9  endfunction
118 function bndcon(typ,jnum,tv)
119     if (typ==1 & jnum==3) then
120         bndv=1.43;
121     end
122     if (typ==2 & jnum==1) then
123         if (tv < 2000) then
124             bndv=50+(100/2000)*tv
125         end
126         if (tv >= 2000) then
127             bndv=150-(100/2000)*(tv-2000)
128         end
129         if (tv > 4000) then
130             bndv=50;
131         end
132     end
133     if (typ==2 & jnum==2) then
134         if (tv < 2000) then
135             bndv=50+(100/2000)*tv
136         end
137         if (tv >= 2000) then
138             bndv=150-(100/2000)*(tv-2000)
139         end
140         if (tv > 4000) then
141             bndv=50;
142         end
143     end
144 endfunction
146 -----
147 // Channel Reach: Start + End -
148 // Flow Depth Condition: 1
149 // Flow Rate (Discharge) Condition: 2
150 -----
151 // Given Data
152 g=9.81 //m/s^2
```

So type 1 junction number 3 we have 1 point 43 which is specified flow depth at the end point. Then it is type 2 and junction number 1. This tv is less than 2000 then starting from 50, 50 plus 100 divided by 100 because it is ranging from 50 to 150. So 100 divided by 2000 into tv. This tv is greater than equal to 2000 then 150 minus which is the maximum value, minus 100 divided by 2000 into tv minus 2000. And tv is greater than 4000 then this is fixed value 50 metres cube per second.

(Refer Slide Time: 05:30)

```
2     if (typ==1 & jnum==3) then
3         bndv=1.43;
4     end
5     if (typ==2 & jnum==1) then
6         if (tv < 2000) then
7             bndv=50+(100/2000)*tv
8         end
9         if (tv >= 2000) then
10            bndv=150-(100/2000)*(tv-2000)
11        end
12        if (tv > 4000) then
13            bndv=50;
14        end
15    end
16    if (typ==2 & jnum==2) then
17        if (tv < 2000) then
18            bndv=50+(100/2000)*tv
19        end
20        if (tv >= 2000) then
21            bndv=150-(100/2000)*(tv-2000)
22        end
23        if (tv > 4000) then
24            bndv=50;
25        end
26    end
27 endfunction
146 -----
147 // Channel Reach: Start + End -
148 // Flow Depth Condition: 1
149 // Flow Rate (Discharge) Condition: 2
150 -----
151 // Given Data
152 g=9.81 //m/s^2
```

The same thing is specified for junction number 2 which is again boundary junction. So with our initial information let us say this is our g value, eps max, t max, delta t. Delta t is 250

seconds and theta is point 5, psi is point 5. So theta equals to point 5 and psi equals to point 5 and junction number is 4. Boundary junction, we have three boundary junctions. Channel number is 3.

(Refer Slide Time: 06:16)

```

endfunction
146
147 -----
148 // Channel Reach: Start + End -
149 // Flow Depth Condition: 1
150 // Flow Rate (Discharge) Condition: 2
151 -----
152 // Given Data
153 g=9.81; //m/s^2
154 global('g');
155 eps_max=1e-6;
156 t_max=25001; //s
157 delta_t=250; //s
158 theta=0.5;
159 psi=0.5;
160 -----
161 junn=4;
162 bjn=3;
163 chin=3;
164 //
165 // Ch# | Length | Width | m1 | m2 | Segment | n | S0 | JN1 | JN2
166 chi_inf=[1 5000 50 0 0 500 0.025 0.0002 1 4
167          2 5000 50 0 0 500 0.025 0.0002 2 4
168          3 5000 100 0 0 500 0.025 0.0002 4 3];
169
170 jun_inf=[-99999 2 2
171          -99999 2 2
172          1 -99999 0
173          -99999 -99999 1];
174 //0: Not Connected
175 //Positive Sign: 1st section of the l-th channel reach is connected
176 //Negative Sign: Ni+1-th section of the l-th channel reach is connected
177 jun_con=[1 1 0 0
178          1 2 0 0
179          1 -3 0 0
180          3 3 -1 -2];
181
182 alpha=ones(chin,1);
183
184 //Derived Information
185 Lx=chi_inf(1:chin,2);
186 B=chi_inf(1:chin,3);
187 m1=chi_inf(1:chin,4);
188 m2=chi_inf(1:chin,5);
189 delta_x=chi_inf(1:chin,6);
190 nm=chi_inf(1:chin,7);
191 S0=chi_inf(1:chin,8);
192
193 //Calculated
194 mode=Lx./delta_x;
195
196 // values
197 for l=1:chin
198     if (jun_inf(chi_inf(l,9),3) > jun_inf(chi_inf(l,10),3)) then

```

Now this is our channel information matrix, junction information matrix and then we have this junction connectivity matrix. Alpha for each channel we will consider that alpha is constant and equals to 1. So this value is 1.

(Refer Slide Time: 06:42)

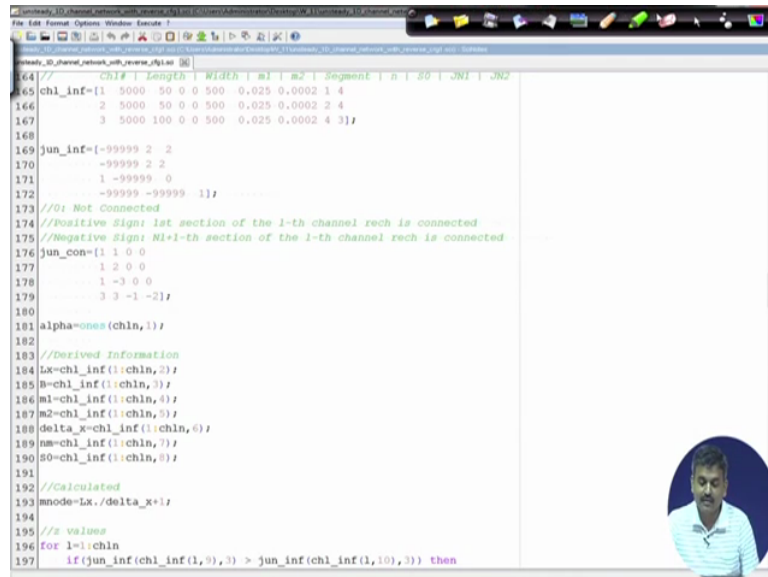
```

164 //
165 // Ch# | Length | Width | m1 | m2 | Segment | n | S0 | JN1 | JN2
166 chi_inf=[1 5000 50 0 0 500 0.025 0.0002 1 4
167          2 5000 50 0 0 500 0.025 0.0002 2 4
168          3 5000 100 0 0 500 0.025 0.0002 4 3];
169
170 jun_inf=[-99999 2 2
171          -99999 2 2
172          1 -99999 0
173          -99999 -99999 1];
174 //0: Not Connected
175 //Positive Sign: 1st section of the l-th channel reach is connected
176 //Negative Sign: Ni+1-th section of the l-th channel reach is connected
177 jun_con=[1 1 0 0
178          1 2 0 0
179          1 -3 0 0
180          3 3 -1 -2];
181
182 alpha=ones(chin,1);
183
184 //Derived Information
185 Lx=chi_inf(1:chin,2);
186 B=chi_inf(1:chin,3);
187 m1=chi_inf(1:chin,4);
188 m2=chi_inf(1:chin,5);
189 delta_x=chi_inf(1:chin,6);
190 nm=chi_inf(1:chin,7);
191 S0=chi_inf(1:chin,8);
192
193 //Calculated
194 mode=Lx./delta_x;
195
196 // values
197 for l=1:chin
198     if (jun_inf(chi_inf(l,9),3) > jun_inf(chi_inf(l,10),3)) then

```

Now we can transfer these values Lx, B, m1, m2, delta x, nm and S not from this channel information matrix directly and mnode we can calculate Lx by delta x plus 1. That will give us mnode or total number of sections for a particular channel.

(Refer Slide Time: 07:09)



```

164 // Channel Information Matrix
165 ch1_inf=[ 1 5000 50 0 0 500 0.025 0.0002 1 4
166           2 5000 50 0 0 500 0.025 0.0002 2 4
167           3 5000 100 0 0 500 0.025 0.0002 4 3];
168
169 jun_inf=[-99999 2 2
170           +99999 2 2
171           1 -99999 0
172           -99999 -99999 1];
173 //0: Not Connected
174 //Positive Sign: 1st section of the l-th channel reach is connected
175 //Negative Sign: Ni+1-th section of the l-th channel reach is connected
176 jun_con=[ 1 1 0 0
177           1 2 0 0
178           1 -3 0 0
179           3 3 -1 -2];
180
181 alpha=ones(ch1n,1);
182
183 //Derived Information
184 Lx=ch1_inf(1:ch1n,2);
185 B=ch1_inf(1:ch1n,3);
186 m1=ch1_inf(1:ch1n,4);
187 m2=ch1_inf(1:ch1n,5);
188 delta_x=ch1_inf(1:ch1n,6);
189 nm=ch1_inf(1:ch1n,7);
190 S=ch1_inf(1:ch1n,8);
191
192 //Calculated
193 mnode=Lx./delta_x+1;
194
195 // values
196 for l=1:ch1n
197     if (jun_inf(ch1_inf(l,9),3) > jun_inf(ch1_inf(l,10),3)) then

```

Now z values, we need to calculate these z values as per our previous calculation thing. That if connectivity this one junction information this channel information L 9. L 9 will give us the starting junction. This is 3, (cha) this junction information is giving the elevation at third column. So starting elevation is more than the ending elevation then this factor is negative because we are starting from that section. Or this factor is positive if this one is less than the ending or elevation of the end section.

(Refer Slide Time: 08:13)


```

294
295 // values
296 for l=1:chln
297     if (jun_inf(chl_inf(1,9),3) <= jun_inf(chl_inf(1,10),3)) then
298         fact=1
299     else
300         fact=1;
301     end
302     zv(1,l)=jun_inf(chl_inf(1,9),3);
303     for i=2:mnode(l)
304         zv(1,i)=zv(1,i-1)+fact*80(l)*delta_x(1);
305     end
306 end
307
308
309 //-----Problem Dependent Parameters-----
310 yv=zeros(sum(mnode),1);
311 Qv=zeros(sum(mnode),1);
312 yO=zeros(sum(mnode),1);
313 QO=zeros(sum(mnode),1);
314
315 //General Identification Matrix
316 idv=0;
317 for l=1:chln
318     for i=1:mnode(l)
319         idv=idv+1;
320         gid(1,i)=idv;
321     end
322 end
323
324 //Initial Value
325 for l=1:chln

```

So we need to specify Qv which is Q value at future time level. Yv, Qv, yO or yOld, QOld and we have this general identification matrix gid which we have utilised for our steady state case also.

(Refer Slide Time: 08:42)

```

207
208 //-----Problem Dependent Parameters-----
209
210 yv=zeros(sum(mnode),1);
211 Qv=zeros(sum(mnode),1);
212 yO=zeros(sum(mnode),1);
213 QO=zeros(sum(mnode),1);
214
215 //General Identification Matrix
216 idv=0;
217 for l=1:chln
218     for i=1:mnode(l)
219         idv=idv+1;
220         gid(1,i)=idv;
221     end
222 end
223
224 //Initial Value
225 for l=1:chln
226     for i=1:mnode(l)
227         if (l==1 | l==2) then
228             yO(gid(1,i))=1.4300;
229             QO(gid(1,i))=50.00;
230         else
231             yO(gid(1,i))=1.4300;
232             QO(gid(1,i))=100.00;
233         end
234     end
235 end
236
237 //Old Time Level
238 yv=yO;
239 Qv=QO;
240

```

Now in this case we need to specify initial values. This is 50 and 1 point 43 for L that means channel 1 and 2. And this is 100 and 1 point 43 for this channel 3.

(Refer Slide Time: 09:06)

```
215 //GENERAL IDENTIFICATION MATRIX
216 idv=0;
217 for l=1:chn
218     for i=1:mnode(l)
219         idv=idv+1;
220         gid(l,i)=idv;
221     end
222 end
223
224
225 //Initial Value
226 for l=1:chn
227     for i=1:mnode(l)
228         if (l==1 | l==2) then
229             y0(gid(l,i))=1.4300;
230             Q0(gid(l,i))=50.00;
231         else
232             y0(gid(l,i))=1.4300;
233             Q0(gid(l,i))=100.00;
234         end
235     end
236 end
237
238 //Old Time Level
239 yv=y0;
240 Qv=Q0;
241
242 //General variable with y and Q
243 qv=zeros(2*sum(mnode),1);
244
245 //Initial Value
246 for l=1:chn
247     for i=1:mnode(l)
248         qv(2*gid(l,i)-1)=yv(gid(l,i)); //y
```



Old time level values we are taking this as guess value. We are starting this. Now we have defined this general variable and we can transfer these values to the general variable. After that we can start our time loop. This is our time loop, this is our space loop.

(Refer Slide Time: 09:36)

```
242 //General variable with y and Q
243 gv=zeros(2*sum(mnode),1);
244 -----
245 //Initial Value
246 for l=1:chln
247     for i=1:mnode(l)
248         gv(2*gid(l,i)-1)=yv(gid(l,i)); //y
249         gv(2*gid(l,i))=Qv(gid(l,i)); //Q
250     end
251 end
252
253 //
254 //
255 tv=0; //Time Level Zero
256 tcount=0;
257 while t<=tmax
258     tv=tv+delta_t;
259
260     //Jacobian Matrix Size
261     A=zeros(2*sum(mnode),2*sum(mnode));
262     r=zeros(2*sum(mnode),1);
263     disp(['Time in Seconds:' string(tv)])
264     count = 0;
265     rmse=1;
266     //Space Loop
267     while rmse > eps_max
268         rmse=0;
269         eqn=0; //Equation Number
270
271         //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
272         for l=1:chln
273             for i=1:mnode(l)-1
274                 //Continuity
275                 eqn=eqn+1;
```

So after starting at tv equals t zero and t count this is time counter equals to zero, after entering we can increase this time value tv equals to tv plus delta t and we can specify over Jacobian matrix A and right hand vector which is r. Time display, time in seconds, count equals to zero, rmse equals to 1 to enter into this space loop.

(Refer Slide Time: 10:08)

```
242 //General variable with y and Q
243 gv=zeros(2*sum(mnode),1);
244 -----
245 //Initial Value
246 for l=1:chln
247     for i=1:mnode(l)
248         gv(2*gid(l,i)-1)=yv(gid(l,i)); //y
249         gv(2*gid(l,i))=Qv(gid(l,i)); //Q
250     end
251 end
252
253 //
254 //
255 tv=0; //Time Level Zero
256 tcount=0;
257 while t<=tmax
258     tv=tv+delta_t;
259
260     //Jacobian Matrix Size
261     A=zeros(2*sum(mnode),2*sum(mnode));
262     r=zeros(2*sum(mnode),1);
263     disp(['Time in Seconds:' string(tv)])
264     count = 0;
265     rmse=1;
266     //Space Loop
267     while rmse > eps_max
268         rmse=0;
269         eqn=0; //Equation Number
270
271         //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
272         for l=1:chln
273             for i=1:mnode(l)-1
274                 //Continuity
275                 eqn=eqn+1;
```

After entering into this space loop rmse greater than epsilon max, rmse is equal to zero, equation number equals to zero. Then we can start adding equation number when we will be writing the equations for continuity.

(Refer Slide Time: 10:32)

```
258 tv=tv+delta_t;
259
260 //Jacobian Matrix Size
261 A=zeros(2*sum(mnode),2*sum(mnode));
262 r=zeros(2*sum(mnode),1);
263 disp(['Time in Seconds: ' string(tv)])
264 count = 0;
265 rmse=1;
266 //space loop
267 while rmse > eps_max
268     rmse=0;
269     eqn=0; //Equation Number
270
271     //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
272     for l=1:chln
273         for i=1:mnode(l)-1
274             //Continuity
275             eqn=eqn+1;
276             A(eqn,'gid(1,i)-1')=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
277             Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),
278             B(1),m(1),m2(1),nm(1)); //y1
279             A(eqn,'gid(1,i)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
280             Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
281             (1),m(1),m2(1),nm(1)); //y1
282             A(eqn,'gid(1,i)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
283             Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
284             (1),m(1),m2(1),nm(1)); //y1
285             A(eqn,'gid(1,i+1)-1')=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
286             Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
287             (1),m(1),m2(1),nm(1)); //y1
288             A(eqn,'gid(1,i+1)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
289             Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
290             (1),m(1),m2(1),nm(1)); //y1
291             r(eqn)=-CL1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(
292             gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m(1),m2(1),
293             nm(1));
```

So this is for continuity. Obviously too many (thi) input variables are required for dc dy calculation. So this is first one, this is second one, this is for y i, Q i, y i plus 1, Q i plus 1 and this is minus CL i.

(Refer Slide Time: 10:59)

```
269 eqn=0; //equation number
270
271 //Equations Corresponding to Segments (2N1+2N2+2N3+2N4)
272 for l=1:chln
273     for i=1:mnode(l)-1
274         //continuity
275         eqn=eqn+1;
276         A(eqn,'gid(1,i)-1')=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
277         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),
278         B(1),m(1),m2(1),nm(1)); //y1
279         A(eqn,'gid(1,i)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
280         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
281         (1),m(1),m2(1),nm(1)); //y1
282         A(eqn,'gid(1,i)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
283         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
284         (1),m(1),m2(1),nm(1)); //y1
285         A(eqn,'gid(1,i+1)-1')=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
286         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
287         (1),m(1),m2(1),nm(1)); //y1
288         A(eqn,'gid(1,i+1)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
289         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
290         (1),m(1),m2(1),nm(1)); //y1
291         r(eqn)=-CL1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(
292         gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m(1),m2(1),
293         nm(1));
294
295         //Momentum
296         eqn=eqn+1;
297         A(eqn,'gid(1,i)-1')=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
298         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),
299         B(1),m(1),m2(1),nm(1)); //y1
300         A(eqn,'gid(1,i)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
301         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
302         (1),m(1),m2(1),nm(1)); //y1
303         A(eqn,'gid(1,i)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
304         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
305         (1),m(1),m2(1),nm(1)); //y1
306         A(eqn,'gid(1,i+1)-1')=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
307         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
308         (1),m(1),m2(1),nm(1)); //y1
309         A(eqn,'gid(1,i+1)=dcdy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),
310         Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B
311         (1),m(1),m2(1),nm(1)); //y1
312         r(eqn)=-CL2(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(
313         gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m(1),m2(1),
314         nm(1));
```

This is for momentum, for momentum we have this dM by dy which is for y i, Q i, y i plus 1, Q i plus 1, this is minus ML i.

(Refer Slide Time: 11:23)

```
280 r(eqn)=-C11(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1));
281
282 //Momentum
283 eqn=eqn+1;
284 A(eqn,2*gid(1,i)-1)=dmxy1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1)); //y1
285 A(eqn,2*gid(1,i))=dmxy2(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1)); //y2
286 A(eqn,2*gid(1,i)+1)=dmxy3(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1)); //y3
287 A(eqn,2*gid(1,i+1)-1)=dmxip1(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1)); //yip1
288 r(eqn)=-M11(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1));
289 end
290 end
291 //
292 // Junction Continuity Condition -JUN=3
293 for j=1:junn
294 dcond=0; // Initially zero, -1 if discharge condition is there
295 if(jun_inf(j,2) == 2) then
296 eqn=eqn+1;
297 r(eqn)=bndcon(j,tv);
298 dcond=1;
299 else
```

So after getting this 2 into N1 plus N2 plus N3 number of equations we should try to get the junction conditions. So first one is for d condition which is our condition in this case. Initial is zero if discharge condition zero equals to 1 if discharge condition is there. That means if discharge condition is there for boundary nodes then only we should utilise this otherwise we should omit it. If in junction information j2 that means column 2 if it is equals to 2 that means our discharge is specified.

We should calculate the discharge from boundary condition bndcon function and I will just change this d condition or dcon value to 1.

(Refer Slide Time: 12:35)

```
288 r(eqn)=-M11(yv(gid(1,i)),Qv(gid(1,i)),yv(gid(1,i+1)),Qv(gid(1,i+1)),yo(gid(1,i)),Qo(gid(1,i)),yo(gid(1,i+1)),Qo(gid(1,i+1)),zv(1,i),zv(1,i+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),nm(1));
289 end
290 end
291 //
292 // Junction Continuity Condition -JUN=3
293 for j=1:junn
294 dcond=0; // Initially zero, -1 if discharge condition is there
295 if(jun_inf(j,2) == 2) then
296 eqn=eqn+1;
297 r(eqn)=bndcon(j,tv);
298 dcond=1;
299 else
300 if(j>bjn) then //For all internal Junctions
301 eqn=eqn+1;
302 r(eqn)=r;
303 dcond=1;
304 end
305 end
306
307 if(dcond==1) then
308 for l=1:jun_con(j,l)
309 if(abs(jun_con(j,l+1)) > eps_max) then
310 if(jun_con(j,l+1) > 0) then
311 jn_node=l;
312 A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=-1;
313 r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,l+1)),jn_node));
314 end
315 if(jun_con(j,l+1) < 0) then
316 jn_node=mnode(abs(jun_con(j,l+1)));
317 A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=1;
318 r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,l+1)),jn_node));
319 end
320 end
```



If junction number is greater than bjn that means boundary junction then this r eqn that is equation number that should be zero because we do not have inflow condition from somewhere. So we will have only the variables. So we should start with the zero value.

(Refer Slide Time: 12:54)

```

288 r(eqn)=-M1(yv(gid(1,1)),qv(gid(1,1)),yv(gid(1,1+1)),Qv(gid(1,1+1)),y0(gid(1,1)),Q0(gid(1,1)),y0(
gid(1,1+1)),Q0(gid(1,1+1)),zv(1,1),zv(1,1+1)),theta,psi,delta_t,delta_x(1),alpha(1),alpha(1),B(1),m1(1),m2(1),
nm(1));
289 end
290 end
291 // Junction Continuity Condition -JUN=3
292 for j=1:junn
293 dcond=0 // Initially zero, -1 if discharge condition is there
294 if(jun_inf(j,2) == 2) then
295 eqn=eqn+1;
296 r(eqn)=Endcon(2,j,tv);
297 dcond=1;
298 else
299 if(j>bjn) then //For all internal Junctions
300 eqn=eqn+1;
301 r(eqn)=0;
302 dcond=1;
303 end
304 end
305
306 if(dcond==1) then
307 for l=1:jun_con(j,1)
308 if(abs(jun_con(j,l+1)) > eps_max) then
309 if(jun_con(j,l+1) > 0) then
310 jn_node=1;
311 A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=-1;
312 r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,l+1)),jn_node));
313 end
314 if(jun_con(j,l+1) < 0) then
315 jn_node=-node(abs(jun_con(j,l+1)));
316 A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=1;
317 r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,l+1)),jn_node));
318 end
319 end
320 end
321 r(eqn)=-r(eqn);
322 end
323 end
324
325 //Junction Energy Condition
326 for j=1:junn
327 if(jun_inf(j,1) == 1) then
328 eqn=eqn+1;
329
330 if(jun_con(1,2) > 0) then jn_node=1; end

```

Now if this dcon or discharge condition equals to 1 then only we should iterate here. Now if junction condition or (connec) junction connectivity depending on whether it is positive or negative we can identify this starting and end nodes and we can add with r eqn equals to r eqn plus Q values or subtract this one Qv.

(Refer Slide Time: 13:32)

```

300 if(j>bjn) then //For all internal Junctions
301 eqn=eqn+1;
302 r(eqn)=0;
303 dcond=1;
304 end
305 end
306
307 if(dcond==1) then
308 for l=1:jun_con(j,1)
309 if(abs(jun_con(j,l+1)) > eps_max) then
310 if(jun_con(j,l+1) > 0) then
311 jn_node=1;
312 A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=-1;
313 r(eqn)=r(eqn)-Qv(gid(abs(jun_con(j,l+1)),jn_node));
314 end
315 if(jun_con(j,l+1) < 0) then
316 jn_node=-node(abs(jun_con(j,l+1)));
317 A(eqn,2*gid(abs(jun_con(j,l+1)),jn_node))=1;
318 r(eqn)=r(eqn)+Qv(gid(abs(jun_con(j,l+1)),jn_node));
319 end
320 end
321 end
322 r(eqn)=-r(eqn);
323 end
324
325 //Junction Energy Condition
326 for j=1:junn
327 if(jun_inf(j,1) == 1) then
328 eqn=eqn+1;
329
330 if(jun_con(1,2) > 0) then jn_node=1; end

```

For each junction node or internal junction node or boundary junction node we will have one discharge equation or discharge continuity equation. But if we have flow depth condition specified at the end section of 3 then we do not need any discharge condition there. Then we should omit that point. For junction energy condition if junction information 1 that is first column equals to 1 somewhere so then we should specify this boundary condition value directly and we should directly specify that value into this yv.

(Refer Slide Time: 14:39)

```

325
326 //Junction Energy Condition
327 for j=1:junn
328     if(jun_inf(j)) == 1 then
329         eqn=eqn+1;
330
331
332         if(jun_con(j,2) > 0) then jn_node1=jn; end
333         if(jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
334         A(eqn,2)*gid(abs(jun_con(j,2)),jn_node1)-bndcon(j,tv);
335         r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1))-bndcon(j,tv);
336
337         r(eqn)=r(eqn);
338         disp(['Energy Equation Number' string(eqn) string(r(eqn))])
339     end
340     if(jun_con(j,1) > 0) then
341         for l=1:jun_con(j,1)-1
342             eqn=eqn+1;
343
344             if(jun_con(j,2) > 0) then jn_node1=jn; end
345             if(jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
346             A(eqn,2)*gid(abs(jun_con(j,2)),jn_node1)-bndcon(j,tv);
347             r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1));
348
349             if(jun_con(j,1+2) > 0) then jn_node2=jn; end
350             if(jun_con(j,1+2) < 0) then jn_node2=mnode(abs(jun_con(j,1+2))); end
351             A(eqn,2)*gid(abs(jun_con(j,1+2)),jn_node2)-bndcon(j,tv);
352             r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
353
354             r(eqn)=r(eqn);
355         end
356     end
357 end
358
  
```

Now in this case if this condition is not satisfied and junction connectivity this one first column we have more than one entry that means let us say for internal junction we have three nodes available or three channels available. Then we should write at least two energy conditions for that one. So that is why I have written L equals junction connectivity minus 1. That means 1 to 2. That means at least two conditions we should add. We have three connected channels so we will have two conditions.

(Refer Slide Time: 15:28)

```
325 //Junction Energy Condition
326 for j=1:junn
327     if (jun_inf(j,1) == 1) then
328         eqn=eqn+1;
329
330         if (jun_con(j,2) > 0) then jn_node1=1; end
331         if (jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
332         A(eqn,2*gid(abs(jun_con(j,2)),jn_node1)-1)=1;
333         r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1))-bndcon(1,j,tv);
334
335         r(eqn)=-r(eqn);
336         disp(['Energy Equation Number' string(eqn) string(r(eqn))]);
337     end
338     if (jun_con(j,1) > 1) then
339         for l=1:jun_npn(j,1)-1
340             eqn=eqn+1;
341
342             if (jun_con(j,2) > 0) then jn_node1=1; end
343             if (jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
344             A(eqn,2*gid(abs(jun_con(j,2)),jn_node1)-1)=1;
345             r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1));
346
347             if (jun_con(j,1+2) > 0) then jn_node2=1; end
348             if (jun_con(j,1+2) < 0) then jn_node2=mnode(abs(jun_con(j,1+2))); end
349             A(eqn,2*gid(abs(jun_con(j,1+2)),jn_node2)-1)=1;
350             r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
351
352             r(eqn)=-r(eqn);
353         end
354     end
355 end
356 end
357 end
358 end
```

Now after writing this finally what we have to do? We have to calculate this Δy_Q and we have to add this Δy_Q with y_Q . y_Q is general variable in our case.

(Refer Slide Time: 15:45)

```
343     eqn=eqn+1;
344
345     if (jun_con(j,2) > 0) then jn_node1=1; end
346     if (jun_con(j,2) < 0) then jn_node1=mnode(abs(jun_con(j,2))); end
347     A(eqn,2*gid(abs(jun_con(j,2)),jn_node1)-1)=1;
348     r(eqn)=yv(gid(abs(jun_con(j,2)),jn_node1));
349
350     if (jun_con(j,1+2) > 0) then jn_node2=1; end
351     if (jun_con(j,1+2) < 0) then jn_node2=mnode(abs(jun_con(j,1+2))); end
352     A(eqn,2*gid(abs(jun_con(j,1+2)),jn_node2)-1)=1;
353     r(eqn)=r(eqn)-yv(gid(abs(jun_con(j,1+2)),jn_node2));
354
355     r(eqn)=-r(eqn);
356 end
357 end
358 end
359
360 //b1 y Q
361 delyQ=A\r;
362
363 for l=1:sum(mnode)
364     yv(l)=yv(l)+delyQ(l);
365     rmsr=rmsr+(yv(l)-yv(l))^2;
366 end
367 //update value
368 for l=1:chn
369     for i=1:mnode(l)
370         yv(gid(l,i))=yv(2*gid(l,i)-1);
371         Qv(gid(l,i))=qv(2*gid(l,i));
372     end
373 end
374
375 rmsr=sqrt(rmsr/sum(mnode));
376 count = count + 1;
```

Now after adding this we need to update these values because updated values should be transferred to y_V and Q_V so that we can utilise these values for next iteration. And we should also calculate this $rmse$ because if $rmse$ is less than ϵ_{max} we should terminate this one.

(Refer Slide Time: 16:13)

```

350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383

```

So after termination of this space loop we should update our time loop value because this new time value will be old time value for our case for the next time loop iteration. So update value for time n which is again specifying the value n plus 1 to nth level. And I have stored these values which is yv at 31 which is equivalent to x is equal to 4000 and I can utilise this value for plotting so that I can get the desired plot for this one.

If you have 100 as del x then yv and Qv should be calculated at 143. If it is del x is 43 then yv and Qv should be collected at 283 to get the information about x is equal to 4000.

(Refer Slide Time: 17:39)

```

367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

```

n ← n+1

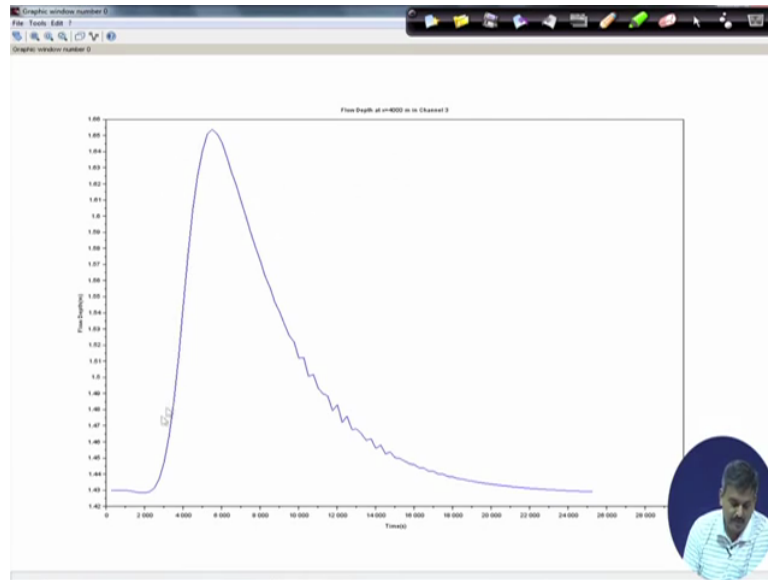
```

389
390
391
392
393
394
395
396
397
398
399
400

```

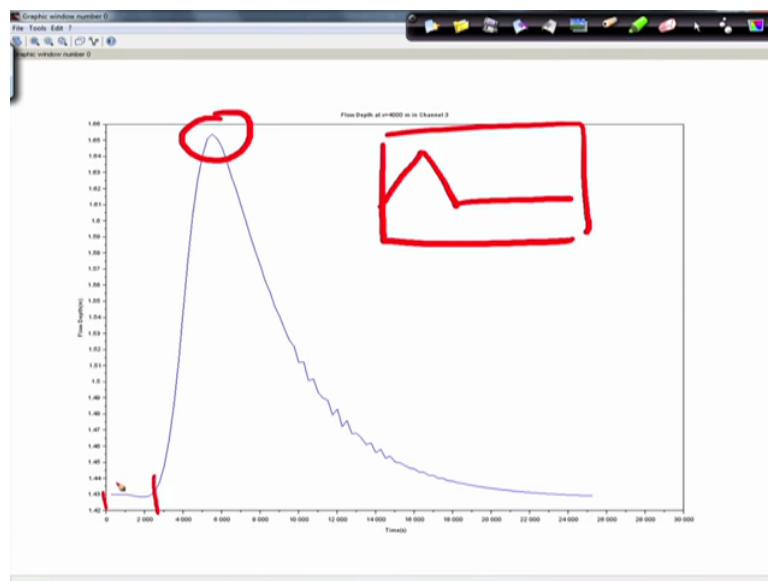
Now at that point if I run this one so time in seconds, so time is increasing so after each convergence in the space loop the time is increasing here. Now in this case we can see that two plots are there. One is for flow depth.

(Refer Slide Time: 18:16)



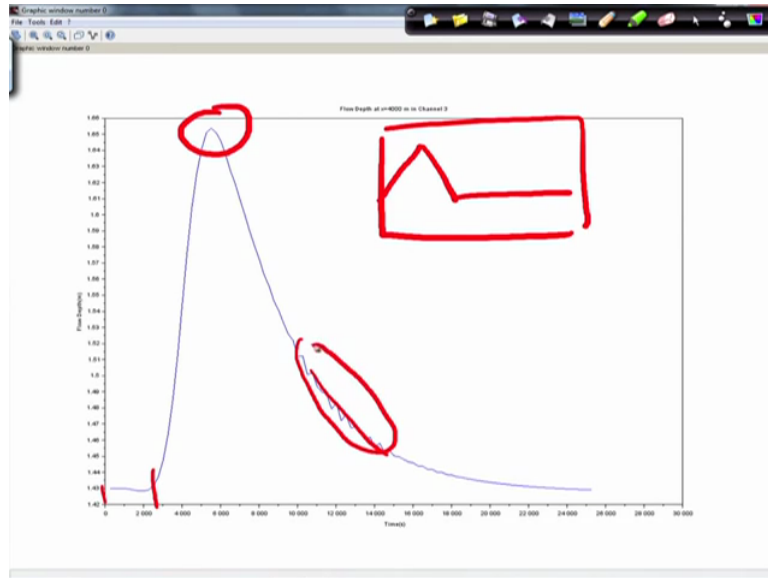
So what flow depth we are getting? Starting from 1 point 43 here this flow depth is reaching up to 1 point 65 and above. But it is below 1 point 66. And again there is decrease because we have considered one inflow discharge at upstream. So obviously there will be increase in the depth initially with a lag, this much is the lag.

(Refer Slide Time: 18:53)



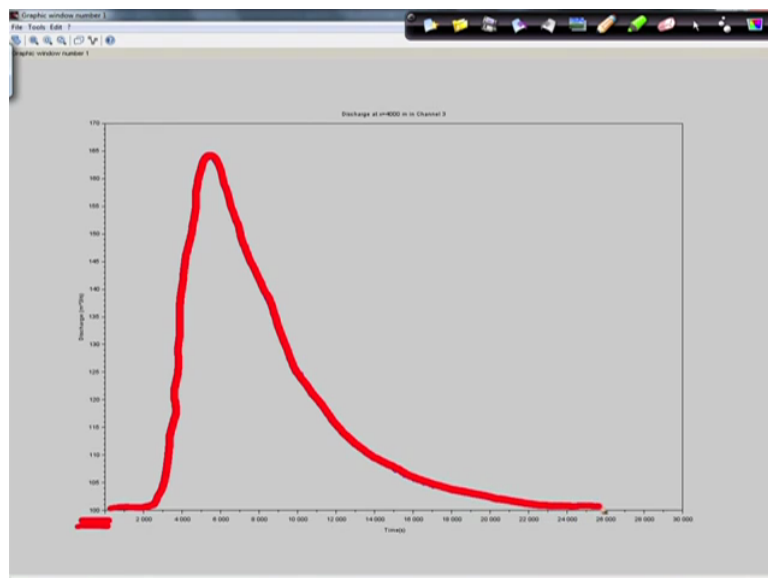
And finally there will be decrease in the depth level. But in this case we can see that some variations are there in depth.

(Refer Slide Time: 19:05)



We can change theta and psi values to get different values here for depth. In this case this is a discharge plot. It is starting from 100 because we have specified 100 metre cube in this case. So this is our plot that we have got.

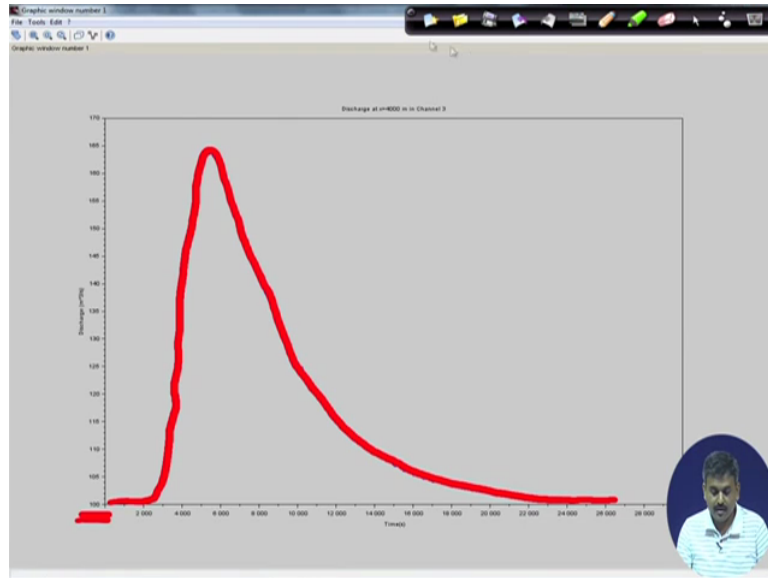
(Refer Slide Time: 19:41)



So interestingly it is reaching up to 165 metre cube per second. And again this discharge values are decreasing here. Initially there is rise in the discharge obviously because of the

inflow condition, the upstream junction nodes at 1 and 2. But finally discharge values are decreasing again reaching to that 100 metre cube per second value.

(Refer Slide Time: 20:18)



So this is all about our unsteady channel flow problem. Now you can utilise this source code unsteady 1D channel network with reverse cfg1 because we have used only configuration 1 in this case. And try to simulate the same problem with different theta and psi values. In this case I have utilised theta equals to point 5 and psi equals to point 5 but you can change the values of theta and psi and check the stability of the problem.

(Refer Slide Time: 21:14)

Problem Statement
Problem Definition
Discretization
References

I.I.T. Kharagpur

List of Source Code

Channel Flow with Reverse Flow

- Channel Network with Configuration 1
 - `unsteady_1D_channel_network_with_reverse_cfg1.sci`

$\theta = 0.5, \psi = 0.5$

Dr. Anirban Dhar NPTEL Computational Hydraulics 25 / 27

So obviously as per literature this problem this theta and psi if I start from zero to 1 in this case, if this is my psi and this is theta so obviously this part is unconditionally stable part.

(Refer Slide Time: 21:42)

The screenshot shows a presentation slide with the following elements:

- Navigation menu: Problem Statement, Problem Definition, Discretization, References.
- Header: I.I.T. Kharagpur.
- Title: List of Source Code.
- Diagram: A 2x2 grid representing a channel network. The top-right cell is shaded with diagonal lines. Red handwritten annotations include a circled θ at the top-left corner and a circled ψ at the bottom-right corner.
- Section: Channel Flow with Reverse Flow.
- List of source code files:
 - Channel Network with Configuration 1
 - `unsteady_1D_channel_network_with_reverse_cfg1.sci`
- Handwritten in red: $\theta \quad \psi$
 $\theta = 0.5, \psi = 0.5$
- Footer: Dr. Anirban Dhar, NPTEL, Computational Hydraulics, and a small circular portrait of the speaker.

That means if both the values are more than point 5 then I have unconditionally stable situation. But check what is the solution if I change the values or I decrease one value with respect to another? Thank you.