

# **ME 543 COMPUTATIONAL FLUID DYNAMICS**

By ----

**MASTER OF TECHNOLOGY  
COMPUTATIONAL MECHANICS  
INDIAN INSTITUTE OF TECHNOLOGY, GUWAHATI**



**DATE OF SUBMISSION : 29-Nov-2020**

**COURSE INSTRUCTOR : Professor Anoop Kumar Das**

```

1:  /**first order upwind scheme**/
2:  #include<stdio.h>
3:  #include<stdlib.h>
4:  #include<conio.h>
5:  #include<math.h>
6:  #define nx 200
7:  #define m 200
8:  #define L 1
9:
10: int main()
11: {
12:     int count = 0,i;
13:     double dx,x[m],u_old[m],u_new[m],lamda = 0.2;
14:     FILE *fp,*fp1;
15:     fp = fopen("final .dat","w");
16:     fp1= fopen("initial.dat ","w");
17:     /**Grid generation**/
18:     dx = L/(nx-1);
19:     for(i=1;i<=nx;i++)
20:     {
21:         x[i] = (i-1)*dx;
22:     }
23:     /**initialize**/
24:     for(i=1;i<=(nx+1)/2;i++)
25:     {
26:         u_old[i] = 0.0;
27:     }
28:     for(i=(nx+1)/2+1;i<=nx;i++)
29:     {
30:         u_old[i] = 1.0;
31:     }
32:     /**writing the initial solution to corresponding output file**/
33:     for(i=1;i<=nx;i++)
34:     {
35:         fprintf(fp,"\n%lf\t%lf",x[i],u_old[i]);
36:     }
37:     /**time marching**/
38:     while(count<=25)
39:     {
40:         for(i=2;i<=nx-1;i++)
41:         {
42:             // first order upwind scheme
43:             u_new[i] = u_old[i] - lamda*(u_old[i]-u_old[i-1]);
44:         }
45:         for(i=2;i<=nx-1;i++)
46:         {
47:             u_old[i] = u_new[i];
48:         }
49:         printf("\n%d",count);
50:         count++;
51:     }
52:     /**writng the final solution to the**/
53:     for(i=1;i<=nx;i++)
54:     {
55:         fprintf(fp1,"\n%lf\t%lf",x[i],u_old[i]);
56:     }
57:     fclose(fp);
58:     fclose(fp1);
59:     return (0);
60: }

```

[illegible][illegible]

[illegible]

[illegible]

[illegible]

[illegible][illegible]

[illegible]



[illegible]

[illegible]

[illegible][illegible]

[illegible]

[illegible]

[illegible]

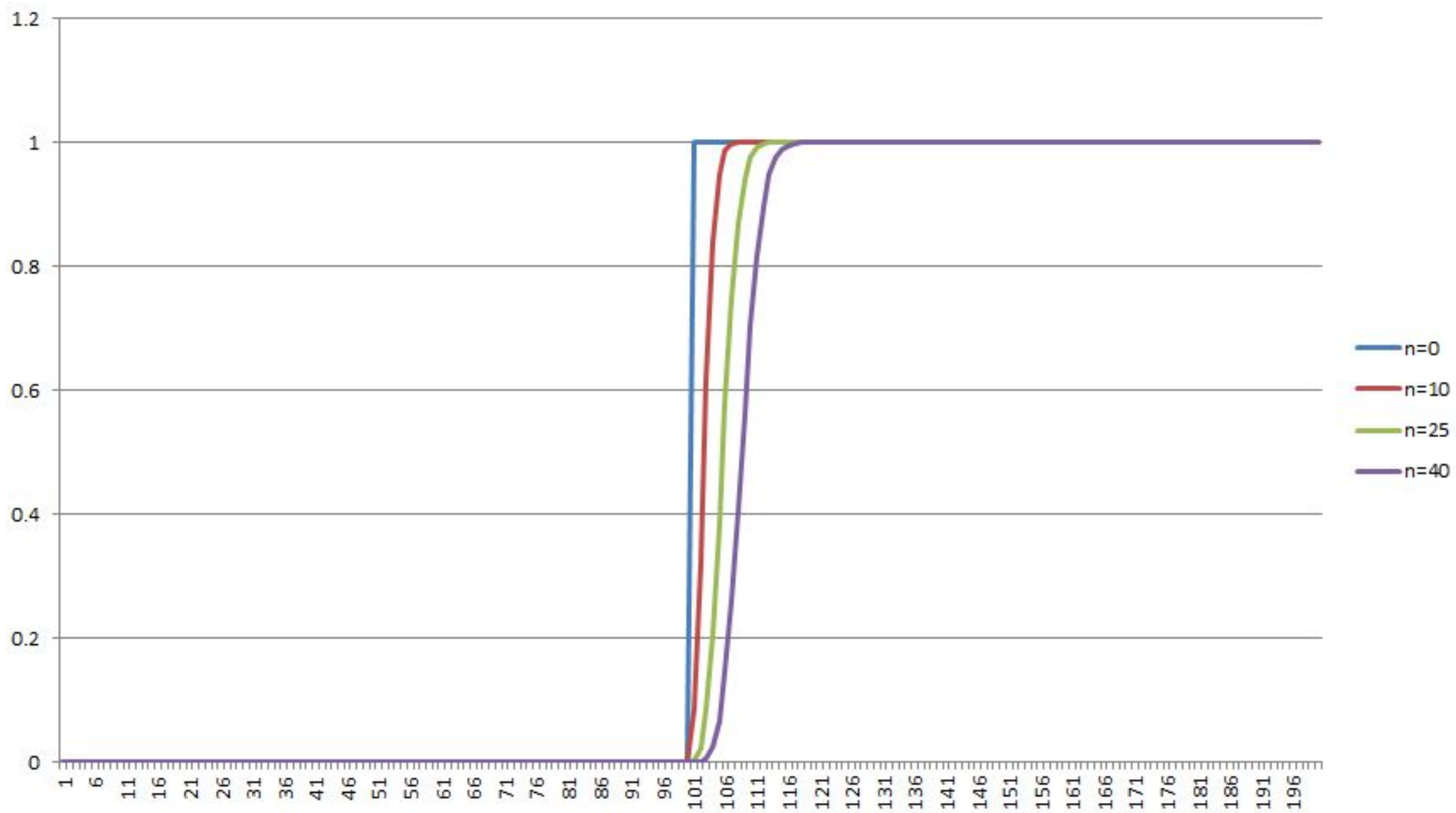
[illegible][illegible]

[illegible]



[illegible]

[illegible]



```

1: /**Lax Wendroff scheme **/
2: #include<stdio.h>
3: #include<stdlib.h>
4: #include<conio.h>
5: #include<math.h>
6: #define nx 200
7: #define m 200
8: #define L 1
9:
10: int main()
11: {
12:     int count = 0,i;
13:     double dx,x[m],u_old[m],u_new[m],lamda = 0.2;
14:     FILE *g,*p;
15:     g = fopen("final .dat","w");
16:     p= fopen("initial.dat ","w");
17:     /**Grid generation***/
18:     dx = L/(nx-1);
19:     for(i=1;i<=nx;i++)
20:     {
21:         x[i] = (i-1)*dx;
22:     }
23:     /**initialize**/
24:     for(i=1;i<=(nx+1)/2;i++)
25:     {
26:         u_old[i] = 0.0;
27:     }
28:     for(i=(nx+1)/2+1;i<=nx;i++)
29:     {
30:         u_old[i] = 1.0;
31:     }
32:     /**writing the initial solution to corresponding output file**/
33:     for(i=1;i<=nx;i++)
34:     {
35:         fprintf(g,"%lf\t%lf",x[i],u_old[i]);
36:     }
37:     /**time marching**/
38:     while(count<=25)
39:     {
40:         for(i=2;i<=nx-1;i++)
41:         {
42:             // Lax Wendroff scheme
43:             u_new[i] = u_old[i] - 0.5*lamda*(u_old[i+1] - u_old[i-1])
44:             + 0.5*lamda*lamda*(u_old[i+1] - 2*u_old[i] + u_old[i-1]);
45:         }
46:         for(i=2;i<=nx-1;i++)
47:         {
48:             u_old[i] = u_new[i];
49:         }
50:         printf("\n%d",count);
51:         count++;
52:     }
53:     /**writng the final solution to the**/
54:     for(i=1;i<=nx;i++)
55:     {
56:         fprintf(p,"%lf\t%lf",x[i],u_old[i]);
57:     }
58:     fclose(g);
59:     fclose(p);
60:     return (0);
}

```

[illegible][illegible]

[illegible]

[illegible]

[illegible]



[illegible][illegible]

[illegible]

[illegible]

[illegible]

[illegible][illegible]

0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000000
0.000000	-0.000000
0.000000	0.000000
0.000000	0.000000
0.000000	-0.000001
0.000000	-0.000001
0.000000	0.000005
0.000000	0.000013
0.000000	-0.000025
0.000000	-0.000100
0.000000	0.000056
0.000000	0.000596
0.000000	0.000389
0.000000	-0.002274
0.000000	-0.004657
0.000000	0.002332
0.000000	0.020258
0.000000	0.024457
0.000000	-0.020243
0.000000	-0.102882

[illegible]

[illegible]



[illegible][illegible]

[illegible]

[illegible]

[illegible]

