TABLE OF CONTENT

1. CODE FOR EXPLICIT SCHEME 2

1. TEMPERATURE CONTOUR & RESULT 4

1. CODE FOR IMPLICIT SCHEME 6

1. TEMPERATURE CONTOUR & RESULT 8

1. COMPARE THE RESULTS 10
2. CONCLUSION 11

1. CODE FOR EXPLICIT SCHEME

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int main()

{

int m,n;

double height,width;

printf("ENTER THE HEIGHT OF RECTANGLE PLATE:\n");

scanf("%lf",&height);

printf("ENTER THE WIDTH OF RECTANGLE PLATE:\n");

scanf("%lf",&width);

printf("\n");

printf("Number of grid along the x(width) direction:\n");

scanf("%d",&m);

printf("Number of grid along the y(height) direction:\n");

scanf("%d",&n);

int np=m\*n; // total number of points

double dx = 1.0/(double)m;

double dy = 1.0/(double)n;

double T[m+1][n+1],T\_prev[m+1][n+1],temp;

int i,j;

double gamma=1.0; //diffusion coefficient

double dt=1.0e-5; //time step;

double t=0.0;

double error=1.0;

double errortime=0.0;

double volume= dx\*dy;

double areae = dy, areaw = dy, distancex = dx;

double arean = dx, areas = dx, distancey = dy;

double as,aw,ap,ae,an,ap0,anb;

ae=(gamma\*areae)/distancex;

aw=(gamma\*areaw)/distancex;

an=(gamma\*arean)/distancey;

as=(gamma\*areas)/distancey;

ap0=volume/dt;

anb=ae+aw+an+as;

ap=ap0;

//Initialization and boundary conditions

for(j=0;j<=n;j++)

{

T[0][j]=0.0;

T[m][j]=0.0;

}

for(i=0;i<=m;i++)

{

T[i][0]=0.0;

T[i][n]=1.0;

}

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

T[i][j]=0.0;

}

}

int iterations=0,timestep=0;

//EXPLICIT

//Outer Time Loop

do

{

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

T\_prev[i][j]=T[i][j];

}

}

iterations =0;

while(error>0.00001)

{

error=0.0;

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

temp=T[i][j];

T[i][j]=((ap0-anb)\*T\_prev[i][j] + ae\*T[i+1][j] + aw\*T[i-1][j] + an\*T[i][j+1] + as\*T[i][j-1])/ap;

error=error+pow((T[i][j] - temp),2.0);

}

}

error=sqrt(error/np);

printf("Iteration-%d\tError=%e\n",iterations,error);

iterations++;

}

error=1.0;

errortime=0.0;

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

errortime=errortime+pow((T[i][j]-T\_prev[i][j]),2.0);

}

}

errortime=sqrt(errortime/np);

t=t+dt;

timestep++;

printf("Time=%lf\ttimestep-%d\tError=%e\n",t,timestep,errortime);

}while(errortime>1.0e-8);

//while(t<1.0);

FILE \*fp;

fp=fopen("temperature.dat","w");

fp=fopen("temperature.txt","w");

printf("i=%d\t\t j=%d\t\tT[i][j]\n",m+1,n+1);

for(j=0;j<=m;j++)

{

for(i=0;i<=m;i++)

{

fprintf(fp,"%lf\t%lf\t%lf\n",i\*dx,j\*dx,T[i][j]);

}

}

fclose(fp);

return 0;

}

1. TEMPERATURE CONTOUR

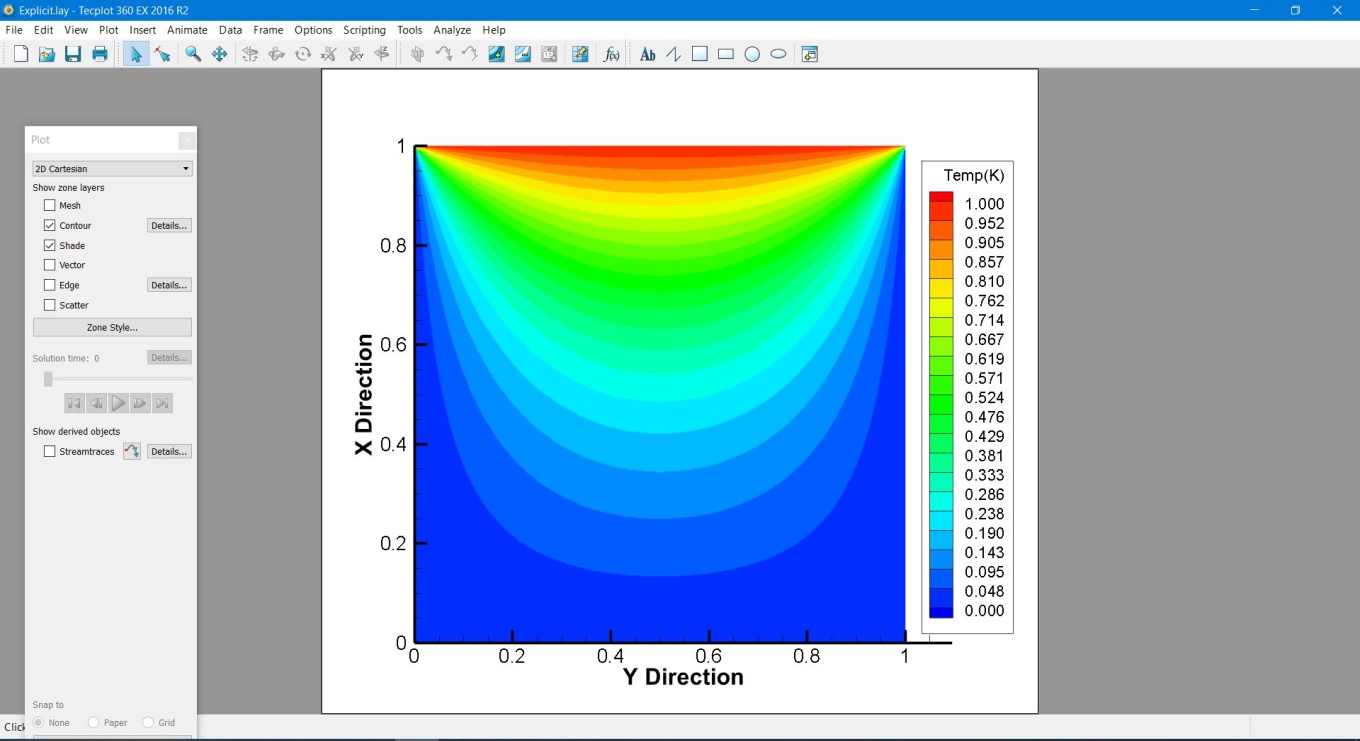


Fig a: Temp contour using Explicit method

RESULT AT EACH ITERATIONS



1. CODE FOR IMPLICIT METHOD

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int main()

{

int m,n;

double height,width;

printf("ENTER THE HEIGHT OF RECTANGLE PLATE:\n");

scanf("%lf",&height);

printf("ENTER THE WIDTH OF RECTANGLE PLATE:\n");

scanf("%lf",&width);

printf("\n");

printf("Number of grid along the x(width) direction:\n");

scanf("%d",&m);

printf("Number of grid along the y(height) direction:\n");

scanf("%d",&n);

int np=m\*n; // total number of points

double dx = 1.0/(double)m;

double dy = 1.0/(double)n;

double T[m+1][n+1],T\_prev[m+1][n+1],temp;

double bb, residual;

int i,j;

double gamma=1.0; //diffusion coefficient

double dt=1.0e-5; //time step;

double t=0.0;

double error=1.0;

double errortime=0.0;

double volume= dx\*dy;

double areae = dy, areaw = dy, distancex = dx;

double arean = dx, areas = dx, distancey = dy;

double as,aw,ap,ae,an,ap0,anb;

ae=(gamma\*areae)/distancex;

aw=(gamma\*areaw)/distancex;

an=(gamma\*arean)/distancey;

as=(gamma\*areas)/distancey;

ap0=volume/dt;

anb=ae+aw+an+as;

ap=ap0+anb;

//Initialization and boundary conditions

for(j=0;j<=n;j++)

{

T[0][j]=0.0;

T[m][j]=0.0;

}

for(i=0;i<=m;i++)

{

T[i][0]=0.0;

T[i][n]=1.0;

}

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

T[i][j]=0.0;

}

}

int iterations=0,timestep=0;

//Implicit

//outer Time Loop

do

{

//copy present to previous

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

T\_prev[i][j]=T[i][j];

}

}

// Inner loop

iterations =0;

while(error>0.00001)

{

error=0.0;

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

residual=ap0\*T\_prev[i][j] - ap\*T[i][j] + ae\*T[i+1][j] + aw\*T[i-1][j] + an\*T[i][j+1] + as\*T[i][j-1];

temp=T[i][j];

T[i][j]=T[i][j]+residual/ap;

error=error+pow((T[i][j]-temp),2.0);

}

}

error=sqrt(error/np);

printf("Iteration-%d\tError=%e\n",iterations,error);

iterations++;

}

error=1.0;

errortime=0.0;

for(i=1;i<m;i++)

{

for(j=1;j<n;j++)

{

errortime=errortime+pow((T[i][j]-T\_prev[i][j]),2.0);

}

}

errortime=sqrt(errortime/np);

t=t+dt;

timestep++;

printf("Time=%lf\ttimestep-%d\tError=%e\n",t,timestep,errortime);

}while(errortime>1.0e-8);

FILE \*fp;

fp=fopen("temperature\_Implicit.dat","w");

fprintf(fp,"ZONE\t I=%d\t J=%d\n",m+1,n+1);

for(j=0;j<=n;j++)

{

for(i=0;i<=m;i++)

{

fprintf(fp,"%lf\t%lf\t%lf\n",i\*dx,j\*dy,T[i][j]);

}

fprintf(fp,"\n");

return 0;

}

1. TEMPERATURE CONTOUR

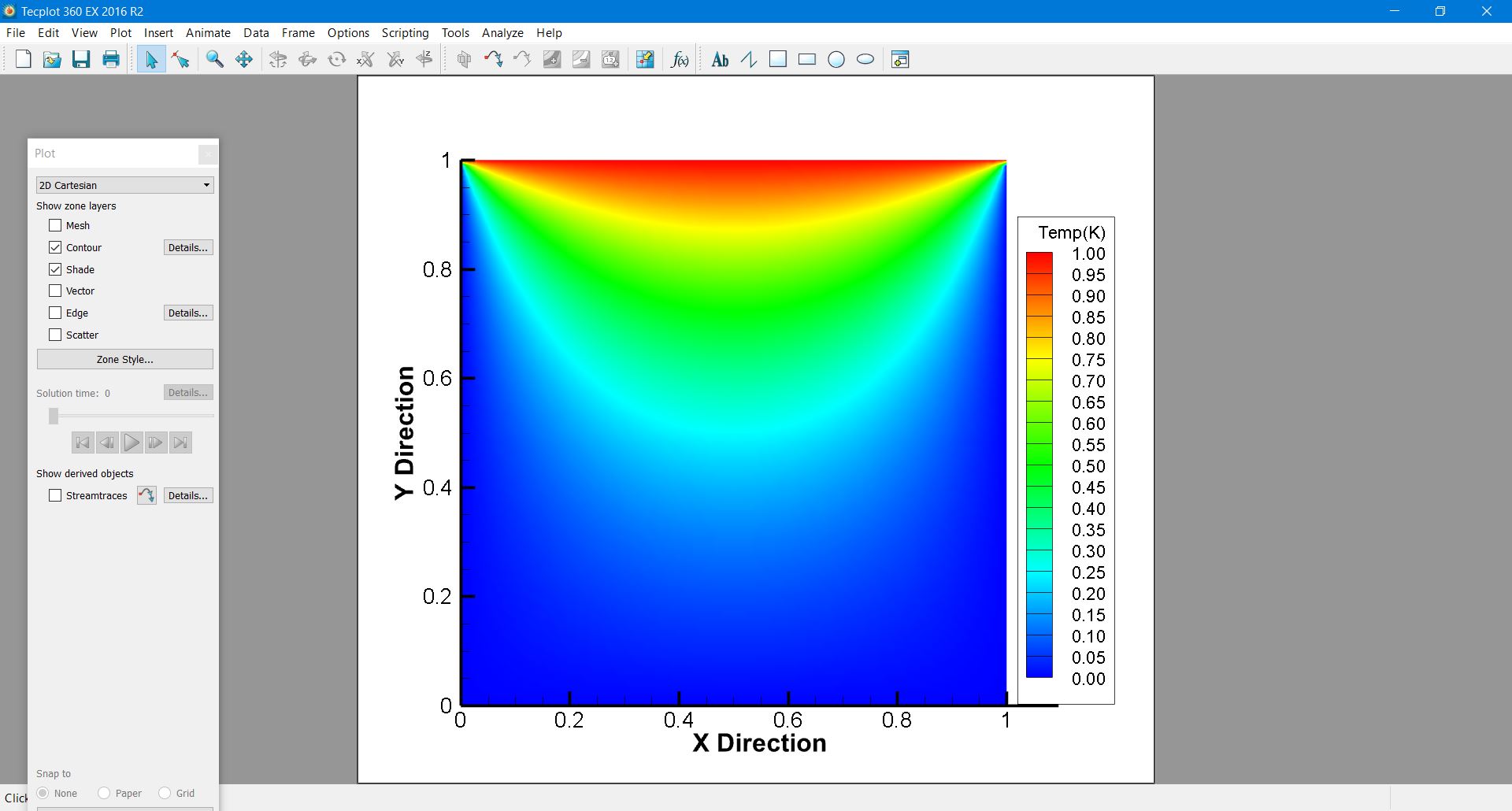
****

Fig b: Temp contour using Implicit method

RESULT AT EACH ITERATIONS



1. COMPARE THE RESULTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Time(sec) | | | |
| Δt | Grid 100 x 100 | | Grid 200 x 200 | |
| Explicit Solver | | | | |
| 10^-4 | | X(diverge) | | X(diverge) |
| 10^-5 | | **8.578 s** | | X(diverge) |
| 10^-6 | | 1 m 6.378 s | | 3 m 20.316 s |

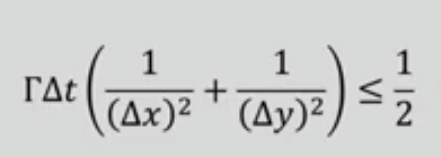
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Time(sec) | | | |
| Δt | Grid 100 x 100 | | Grid 200 x 200 | |
| Implicit Solver | | | | |
| 10^-6 | | 1 m 11.477 s | | 4 m 22.695 s |
| 10^-5 | | **13.097 s** | | 1 m 4.659 s |
| 10^-4 | | 4.426 s | | 37.102 s |
| 10^-3 | | 3.255 s | | 33.677 s |
| 10^-2 | | 2.991 s | | 31.760 s |
| 10^-1 | | 2.516 s | | 27.559 s |
| 1 | | 2.047 s | | 23.640 s |

1. CONCLUSION

In the Explicit solver, time step for the next iterations is restricted and acceptable time step from the stability criteria is Δt ≤ 2.5 x 10^-5.

Equation for Stability: Given data Δx = Δy = 0.01 and Γ = 1.

.



In the Implicit solver there is no restriction over selection of time step **as Implicit is unconditionally stable** so we can use higher time step. At time step 10^-5 we observed that implicit solver taking more time as we can see it involve comparatively more arithmetic operations. As we increase the time step solution will converge quickly.

**At Δt =1 , solution will converge in 2.047 s**. we cannot take Δt =1 for Explicit solver.

Select the Δt such a way that physics of the problem must be captured. (say, rain drop is falling on a pond so the phenomena happen in the range of millisecond and if we use Δt as 0.1 or 1 or 10 then we will not be able to captured the physics of the problem. Although the implicit solver converges but we may missed in between physics.

