Numerical simulations using solvers developed for computational fluid dynamics

A project report submitted in partial fulfilment of the requirements of the course ME670 (Advanced computational fluid dynamics)

By

Name: Prashil Bhaskarrao Thool

Roll No: 224103105

Department of Mechanical Engineering Indian Institute of Technology Guwahati Guwahati-781039

March 2023

Contents

Chapter 1

Time integration (4^{th} order Runge-kutta) method

Question-1:

Code for 4^{th} order Runge-Kutta method:
Consider transient 2D conduction problem governed by equation

$$\frac{\partial \theta}{\partial t} = \alpha \left[\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right]$$

Take a rectangular domain of width L and height W The boundary conditions are:

At x = 0: ∂θ/∂x = 0
At x = L: θ = 0
At y = 0: ∂θ/∂y = 0
At y = W: θ = 0
Initial condition: At t = 0: θ_i(x, y) = 1

Write a CFD code for finding the temperature variation. This problem has an analytical solution which is given by a series solution expressed as

$$\frac{\theta(x,t)}{\theta_i} = 4\left[\sum_{n=0}^{\infty} \frac{(-1)^n}{\lambda_n L} e^{-\alpha\lambda_n^2 t} \cos\lambda_n x\right]\left[\sum_{m=0}^{\infty} \frac{(-1)^m}{\lambda_m L} e^{-\alpha\lambda_m^2 t} \cos\lambda_m y\right]$$

where λ_n and λ_m are given by

$\lambda_n = (2n + 1)\frac{\pi}{2L}$ n = 0, 1, 2, 3 . . . .

$\lambda_m = (2m + 1)\frac{\pi}{2W}$ m = 0, 1, 2, 3 . . . .

Take $L = 2$, $W = 1$, $\theta i = 1$ and $\alpha = 1$. In your code you also write a subroutine to calculate temperature variation using the above analytical expression given by Eq. (1). Your results should contain at least the following:

(a) Compare the temperature contours obtained by the code and with those obtained by using the above analytical expression at $t = 0.1$(show the temperature contours figures side by side).

(b) Compare the temperature contours obtained by the code and with those obtained by using the above analytical expression at $t = 0.2$(show the temperature contours figures side by side).

(c) Compare the temperature contours obtained by the code and with those obtained by using the above analytical expression at $t = 1.0$(show the temperature contours figures side by side).

(d) Compare the temperature variation with y along mid-vertical plane $x = 0.5$ from the code and the above analytical expression for $t=0.1$. Plot both temperature profiles in the same figure.

(e) Compare the temperature variation with y along mid-horizontal plane $y = 0.5$ from the code and the above analytical expression for $t=0.5$. Plot both temperature profiles in the same figure.

(f) Temperature variation at point $(x, y) = (L/4, W/4)$ with respect to time till steady state is reached.

(a)

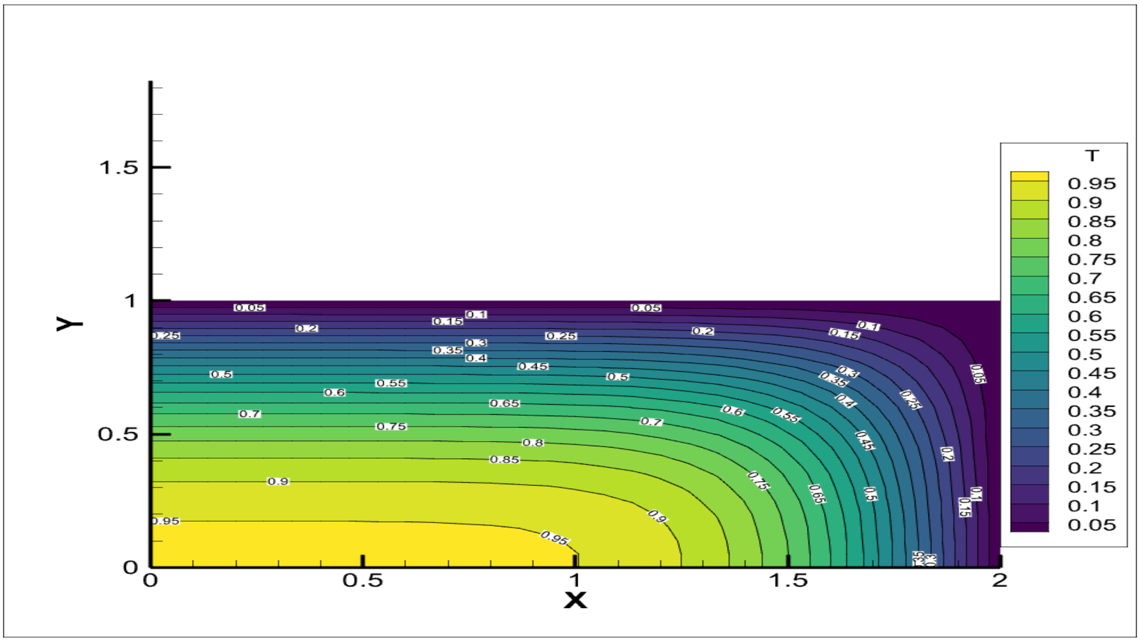

Fig 1: contour at t = 0.1

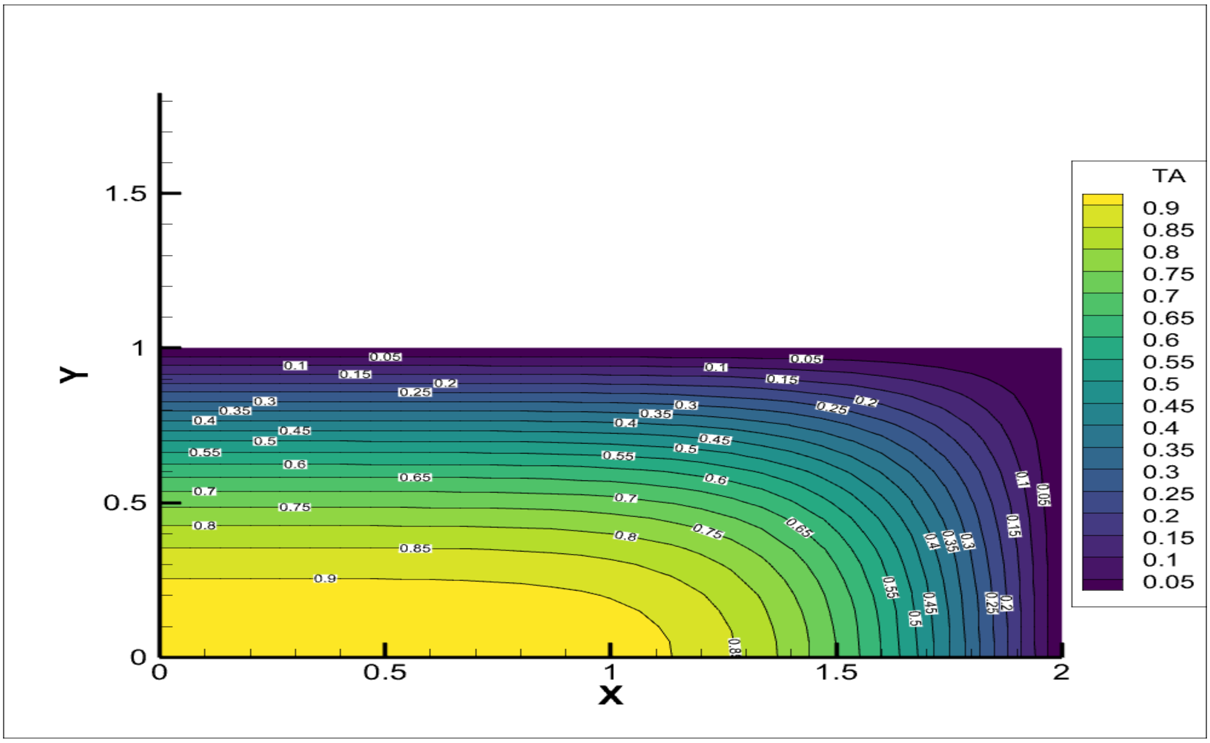

Fig 1.1: analytical at t = 0.1

(b)

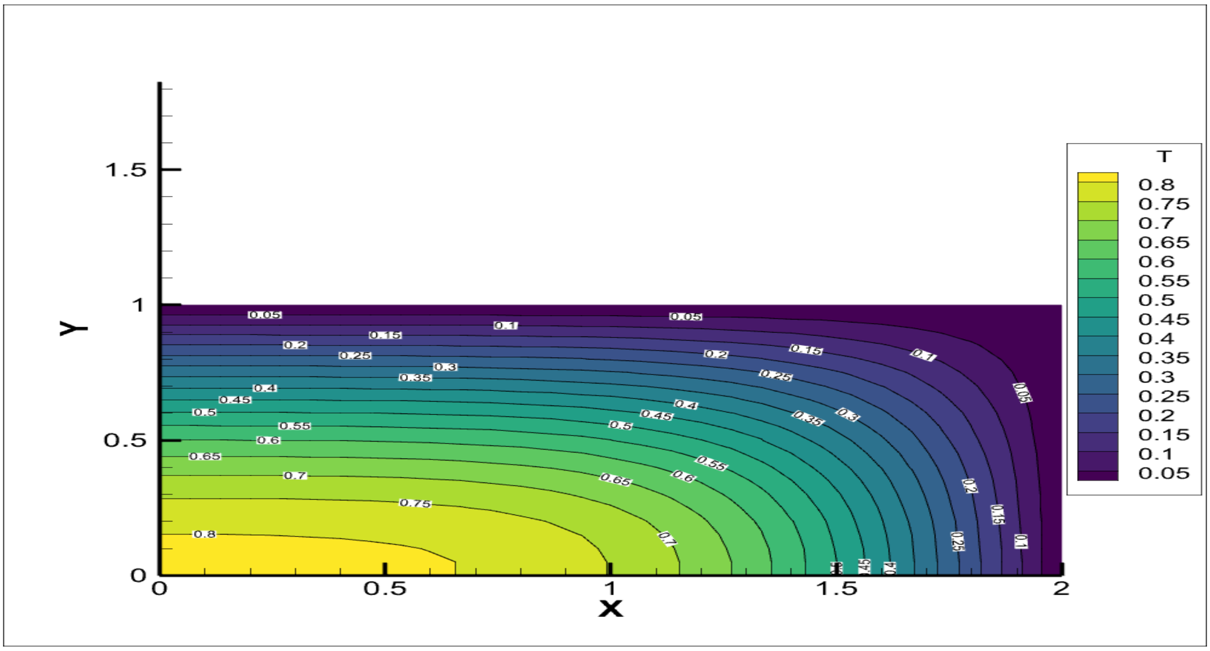

Fig 2: contour at t=0.2

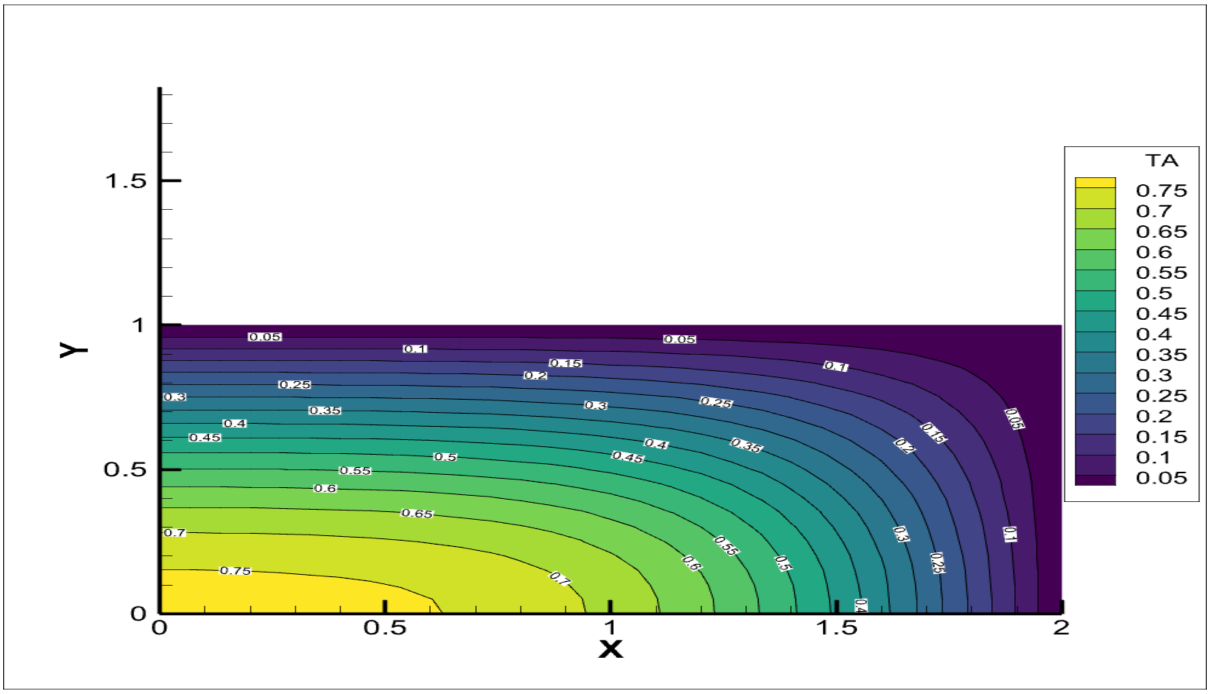

Fig 2.1: analytical at t=0.2

(c)

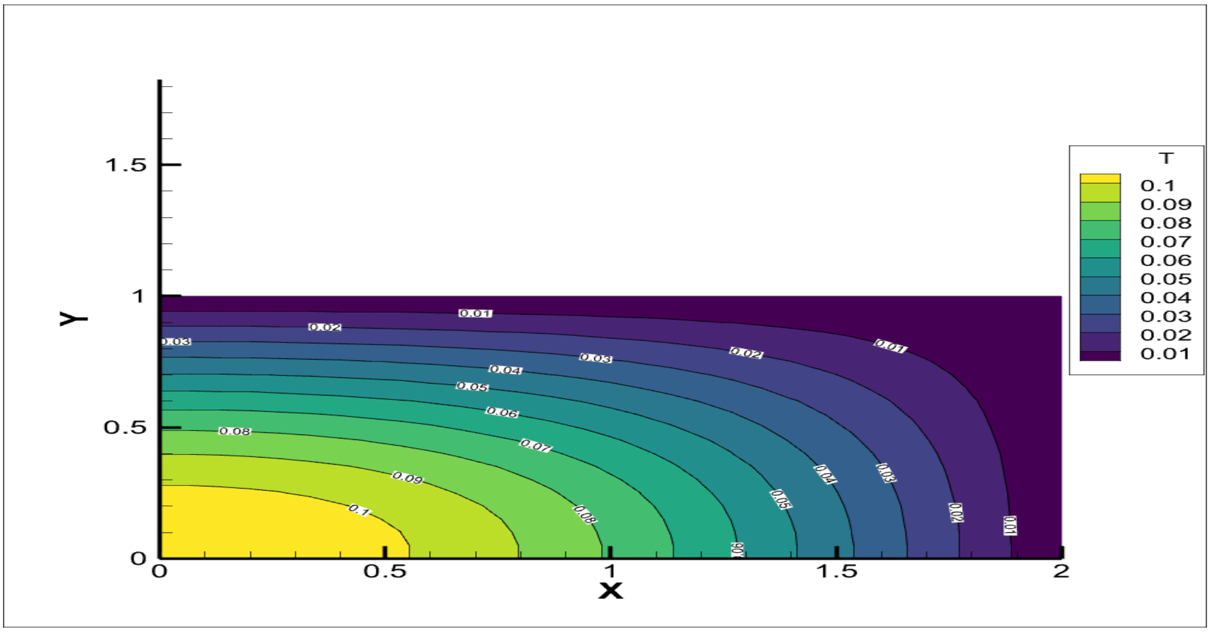

Fig 3: contour at t=1

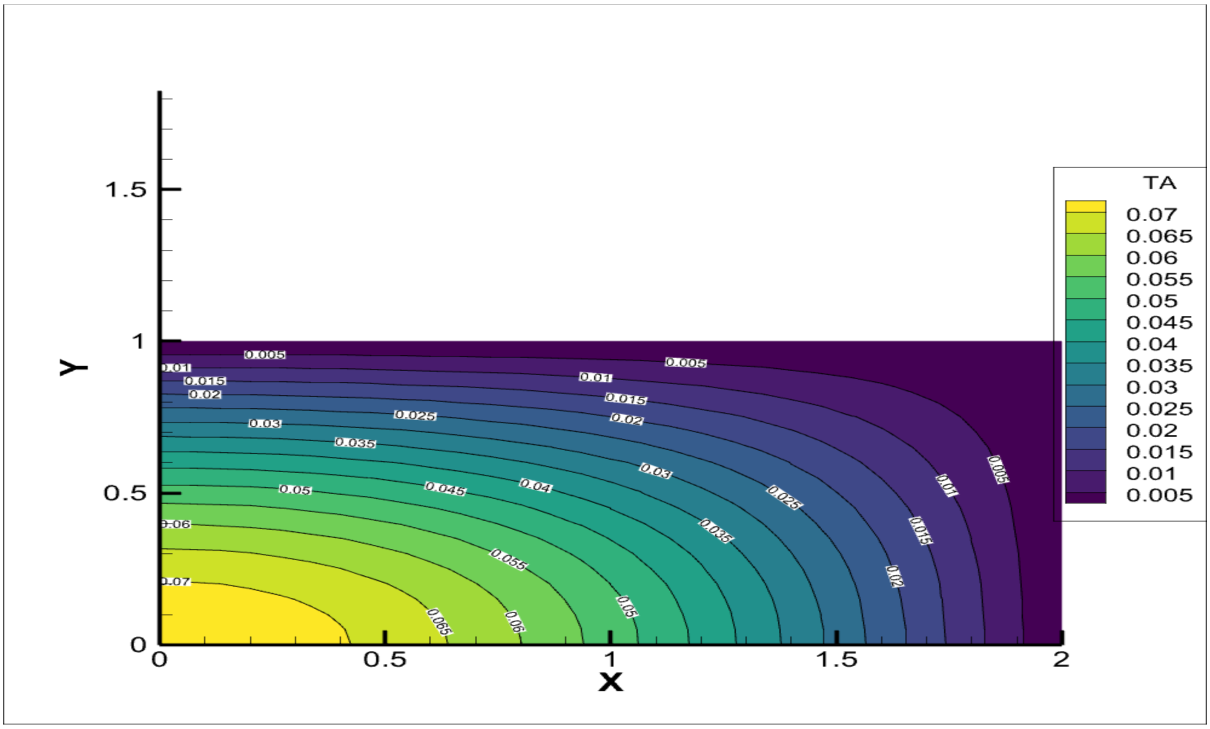

Fig 3.1: analytical at t=1

(d)

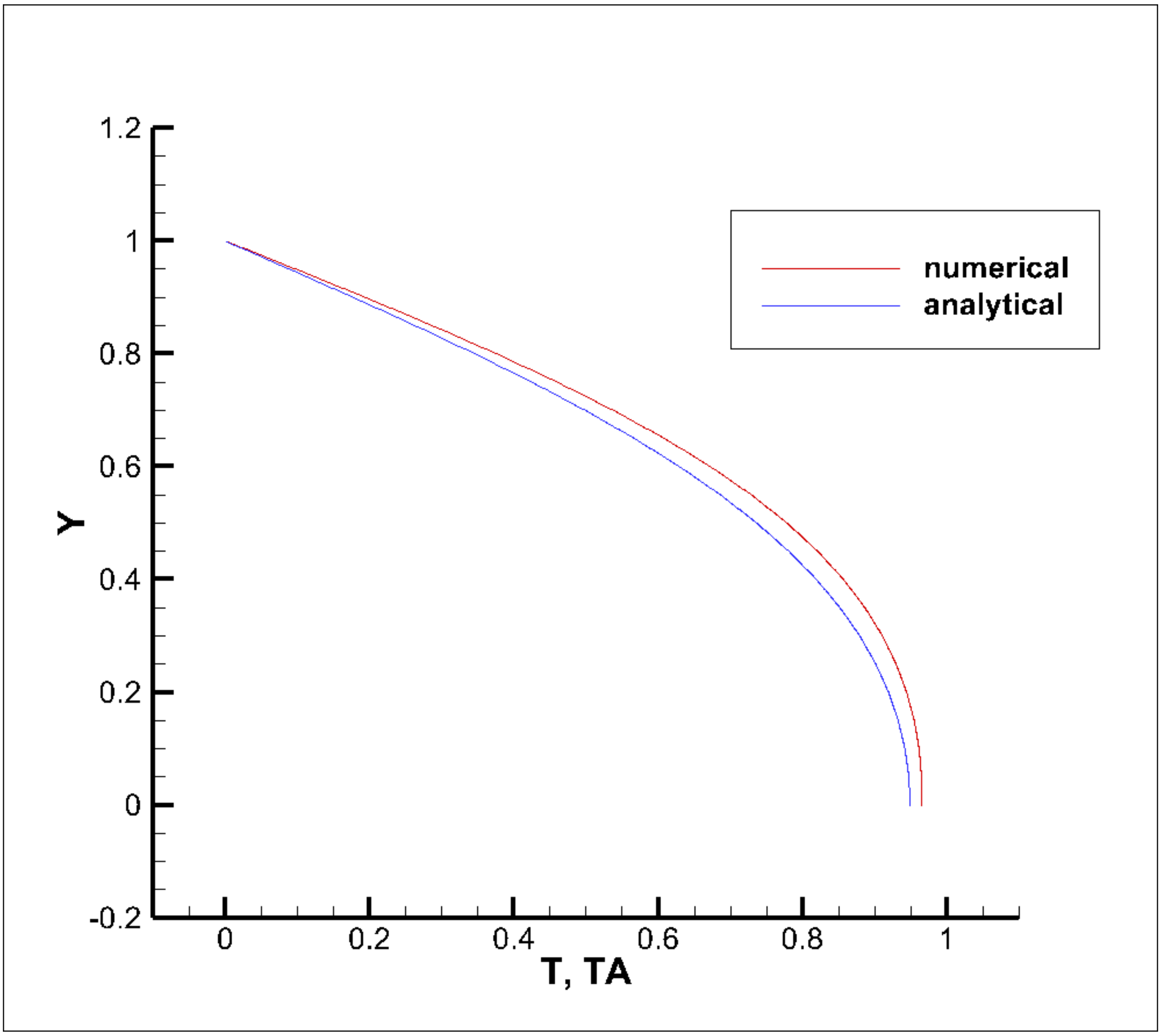

Fig 4: at x=0.5, t=0.1, analytical at x=0.5, t = 0.1

(e)

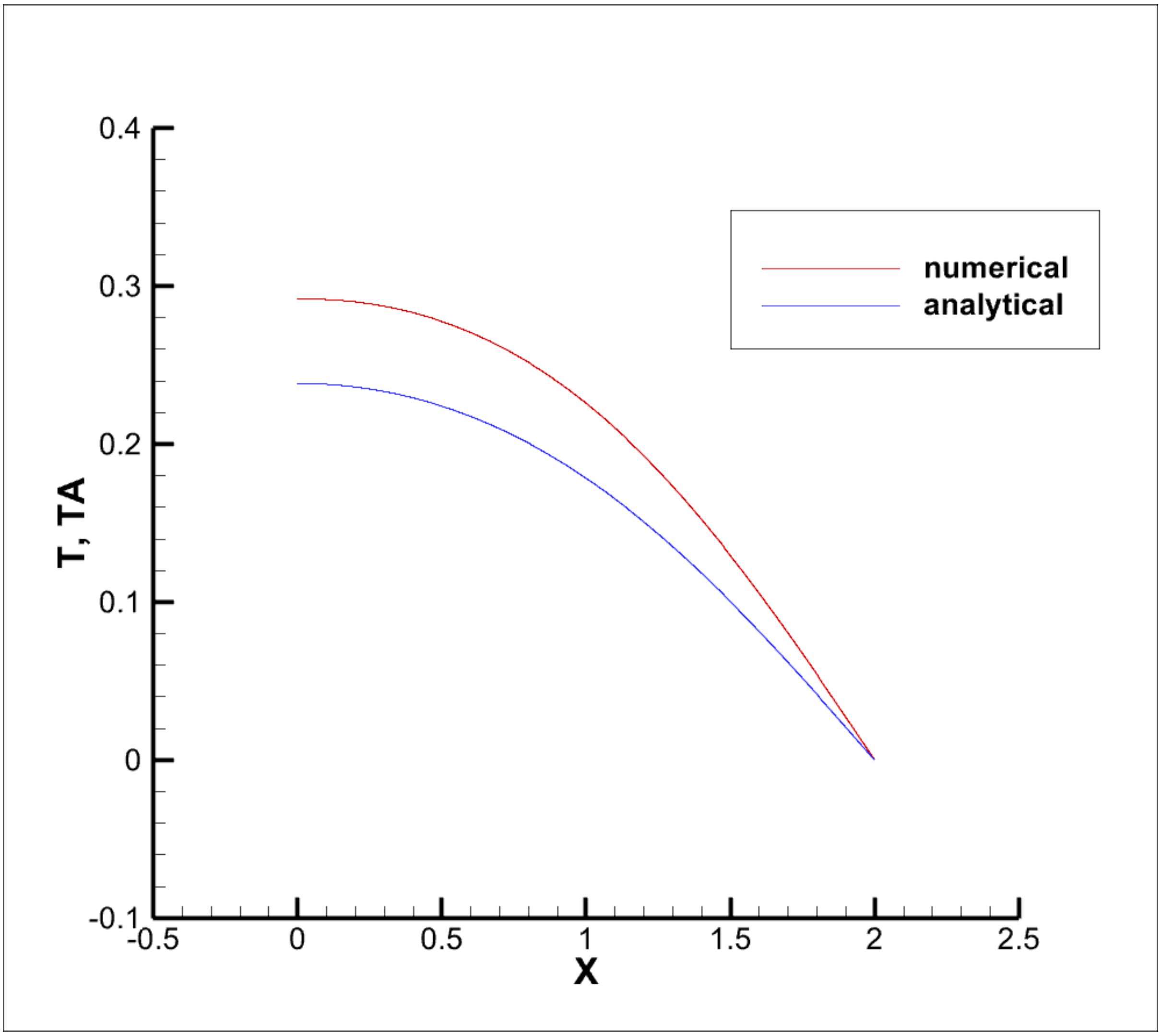

Fig 5: at y=0.5, t=0.5, analytical at y=0.5, t=0.5

(f)

For m = 30 and n=20 L/4 = 7.5≅ 8 and W/4 = 5

Steady state is shown by following figure (Temperature Vs time)

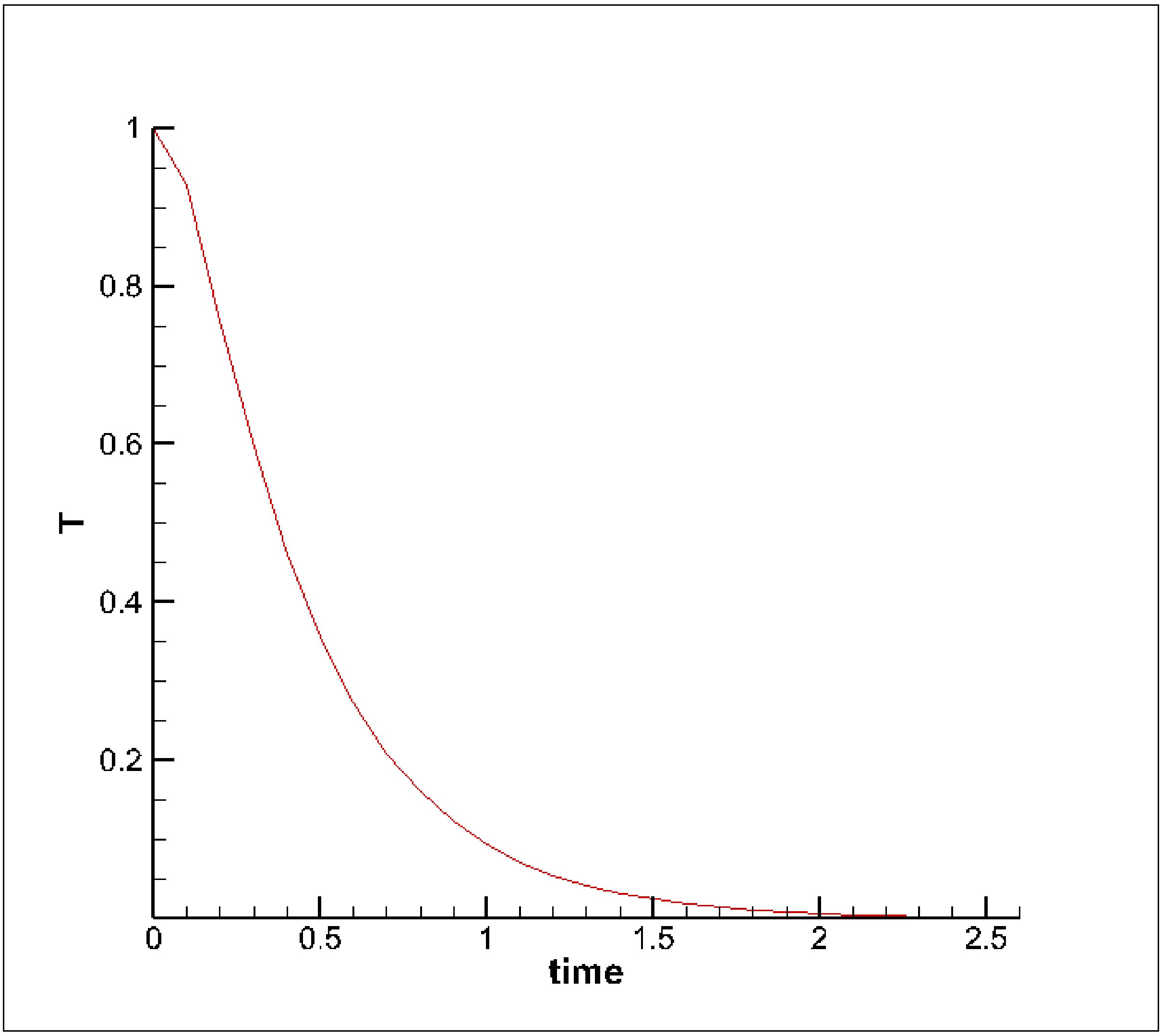

Fig: steady state

The code

```
#include<stdio.h>

#include<stdlib.h>

#include <math.h>

int main(void)

{

    int n=31,m=21,iteration=0,i,j,k;

    char name[10];

    char Aname[10];

    double T[n][m],Tem[n][m],TA[n][m];

    double F1[n][m],F2[n][n],F3[n][m],F4[n][m];

    double K1[n][m],K2[n][n],K3[n][m],K4[n][m];

    double t = 0.0, dx, dy,error=1.0 , alpha = 1.0, dt = 0.001,sumA=0.0 ,
sumB=0.0,la[m],lb[n],pi = (22.0/7.0),L=2.0,W=1.0;

    double LA, LB,term1[n],term2[m];



    dx = 2.0/(n-1);

    dy = 1.0 / (m-1);

    printf("%.9f\n",pi);

    int sum ;

    FILE*file2;

    file2 = fopen("df_bound.m","w");

    fprintf(file2,"df_bound=[");

    // Boundary Conditions

    for ( i = 0; i < n; i++)
```

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

       T[i][j]=0.0;
       T[i][j] = 1.0;
       T[n-1][j]= 0.0;
       T[i][m-1]=0.0;



       //printf("%f\t",T[i][j]);
       fprintf(file2,"%f\t",T[i][j]);

    }
    fprintf(file2,"\n");

}

// Analytical solutions
while (t<10)
{
    for (  i = 0; i < n; i++)
    {
        term1[i] = 0.0;
    }
    for ( j = 0; j < m; j++)
```

```
{
   term2[j]=0.0;
}



if((iteration+100)%100==0)
{
   sprintf(Aname, "Analytical_Temperature_%f.plt",t);
   FILE*file4;
   file4=fopen(Aname,"w");
   fprintf(file4,"VARIABLES= \"X\",\"Y\",\"TA\"\n");
   fprintf(file4,"ZONE F=POINT\n ");
   fprintf(file4,"j=%d, I=%d\n",n,m);

   for(i=0;i<n;i++)
   {
      for ( j = 0; j < m; j++)
      {
         fprintf(file4,"\n%lf\t%lf\t%lf\n",i*dx,j*dy,TA[i][j]);
      }
   }
   fclose(file4);
}
 for(i=0;i<n;i++)
  {
  for(k=0;k<1000;k++)
```

```
{
 LA = pi*(2 * k +1)/(2*L);
 term1[i]=term1[i]+(((pow((-1),k)))/(LA*L))*(exp(-alpha*LA*LA*t))*(cos(LA*i*dx));



}
//printf("%f\t",k1[i])
}
for(j=0;j<m;j++)
{

 for(k=0;k<1000;k++)
{
LB = pi*(2 * k +1)/(2*W);
 term2[j]=term2[j]+(((pow((-1),k)))/(LB*W))*(exp(-alpha*LB*LB*t))*(cos(LB*j*dy));
}
}

for ( i = 0; i < n; i++)
{
   for ( j = 0; j <m; j++)
   {
      TA[i][j] = 4 * term2[j] *term1[i] ;
   }
```

```
    }



    t = t+dt;
    iteration ++;
    printf("iteration=%d\ttime=%f\n",iteration,t);


}


iteration =0;
//Runga Kutta method
t=0.0;
FILE*fstdy;
fstdy = fopen("steady.plt","w");
fprintf(fstdy,"VARIABLES= \"time\",\"T\"\n");


while(t<10)
{
    if((iteration+100)%100==0)
    {
        sprintf(name, "Temperature_%f.plt",t);
        FILE*file1;
```

```
    file1=fopen(name,"w");
    fprintf(file1,"VARIABLES= \"X\",\"Y\",\"T\"\n");
    fprintf(file1,"ZONE F=POINT\n ");
    fprintf(file1,"j=%d, I=%d\n",n,m);

    for(i=0;i<n;i++)
    {
        for ( j = 0; j < m; j++)
        {
            fprintf(file1,"\n%lf\t%lf\t%lf\n",i*dx,j*dy,T[i][j]);
        }
    }
    fclose(file1);
}
// calculating F1ij
for ( i = 1; i < n-1; i++)
{
    for ( j = 1; j < m-1; j++)
    {
        F1[n][m] = alpha *( (T[i-1][j]- 2* T[i][j] + T[i+1][j])/(dx*dx)
        + (T[i][j-1] - 2* T[i][j] + T[i][j+1])/(dy*dy) );

        K1[i][j] = T[i][j] + dt * (F1[i][j])/2.0 ;
    }

}
```

```
// Nuemanmm Boundary condtions for K1
for ( i = 0; i <n; i++)
{
   for ( j = 0; j < m; j++)
   {
      K1[i][0] = K1[i][1];
      K1[0][j] = K1[1][j];
   }

}

// Calculating F2ij
for ( i = 1; i <n-1; i++)
{
   for ( j = 1; j < m-1; j++)
   {
      F2[i][j] = alpha * ( ( K1[i-1][j] - 2 * K1[i][j] + K1[i+1][j] )/(dx*dx)
      + ( K1[i][j-1] - 2 * K1[i][j] + K1[i][j+1])/(dy*dy) );

      K2[i][j] = T[i][j] + dt * (F2[i][j])/2.0 ;
   }

}
// Nuemanmm Boundary condtions for K2
for ( i = 0; i < n; i++)
{
```

```
    for ( j = 0; j < m; j++)
    {
        K2[i][0] = K2[i][1];
        K2[0][j] = K2[1][j];
    }

}

// Calculating F3ij
for ( i = 1; i < n-1; i++)
{
    for ( j = 1; j < m-1; j++)
    {
        F3[i][j] = alpha * ( ( K2[i-1][j] - 2 * K2[i][j] + K2[i+1][j] )/(dx*dx)
        + ( K2[i][j-1] - 2 * K2[i][j] + K2[i][j+1] )/(dy*dy) );

        K3[i][j] = T[i][j] + dt * F3[i][j] ;
    }

}

//  Nuemanmm Boundary condtions for K3
for ( i = 0; i < n; i++)
{
    for ( j = 0; j < m; j++)
    {
```

```
            K3[i][0] = K3[i][1];
            K3[0][j] = K3[1][j];
        }

    }

    // Calculating F4ij
    for ( i = 1; i < n-1; i++)
    {
        for ( j = 1; j < m-1; j++)
        {
            F4[i][j] = alpha * ( ( K3[i-1][j] - 2 * K3[i][j] + K3[i+1][j])/(dx*dx)
            + ( K3[i][j-1] - 2 * K3[i][j] + K3[i][j+1])/(dy*dy) );

            T[i][j] = T[i][j] + dt * ( F1[i][j] + 2 * F2[i][j] + 2 * F3[i][j] + F4[i][j]
)/6.0 ;
        }

    }

    // Nuemanmm Boundary condtions for Tij
    for ( i = 0; i < n; i++)
    {
        for ( j = 0; j < m; j++)
        {
            T[i][0] = T[i][1];
            T[0][j] = T[1][j];
```

```
   }


}
// calculating for the steady state at L/4 and W/4


if ((iteration+100)%100==0)
{
   for ( i = 0; i < n; i++)
{
   for ( j = 0; j < m; j++)
   {


      if (T[8][5]> 1e-3)
      {
         fprintf(fstdy,"%f\t%f\n",t,T[8][5]);
      }


   }


}
}






printf("iteration=%d\t",iteration);
```

```
        t=t+dt;

        printf("TIME2a=%.10f\n",t);



        iteration++;

    }



}
```