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

1 Incomplete L U Decomposition

1.1 The code of incomplete LU scheme . . . .

2 Strongly Implicit Procedure

2.1 The code of SIP scheme. . . . .

Chapter 1

Incomplete L U Decomposition

Question-1:

Code for Incomplete LU decomposition (ILU): Consider 2D conduction problem governed by equation.

$$\frac{\partial^2 T}{\partial X^2} + \frac{\partial^2 T}{\partial Y^2} = 0$$

Take a square domain of width and height of 1 unit. The boundary conditions are: At x = 0: T = 0 At x = 1: T = 0 At y = 0: T = 0 At y = 1: T = 1 Discretise the governing equation and solve the obtained system of linear equations by writing code for Incomplete LU decomposition (ILU). This problem has an analytical solution which is given by series solution given by

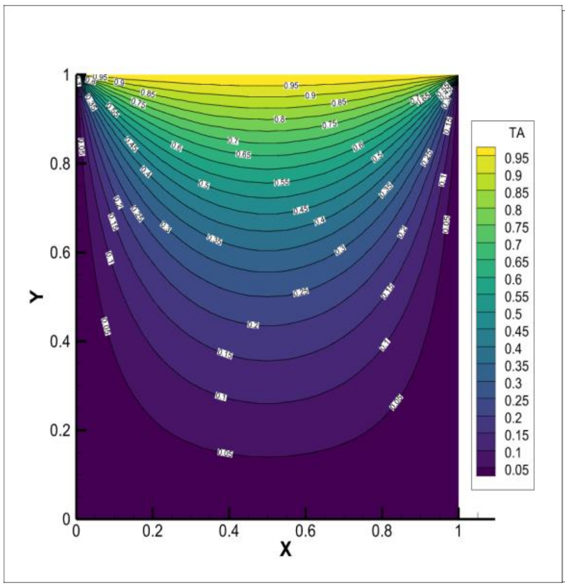
T(x, y) = 2 /π $\sum_{n=1}^{inf} \frac{-1^{n+1}+1}{n}$ sin(nπx) sinh nπy /sinh(nπ)

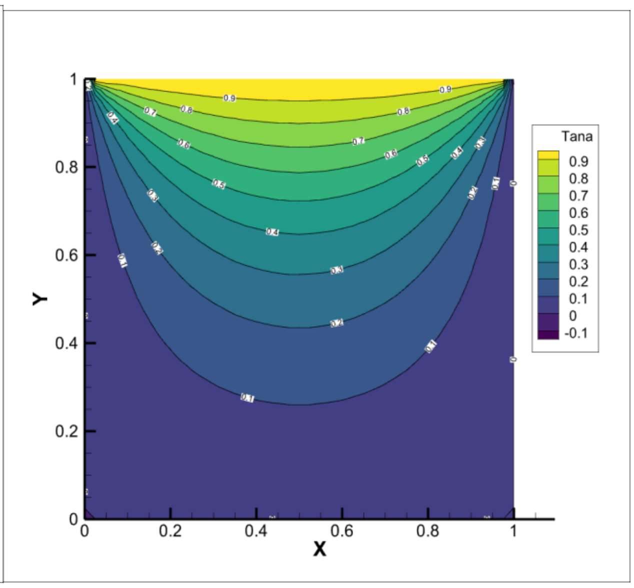
(a) Compare the temperature contours obtained by the code and with those obtained by using the above analytical expression (show the temperature contours figures side by side).

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

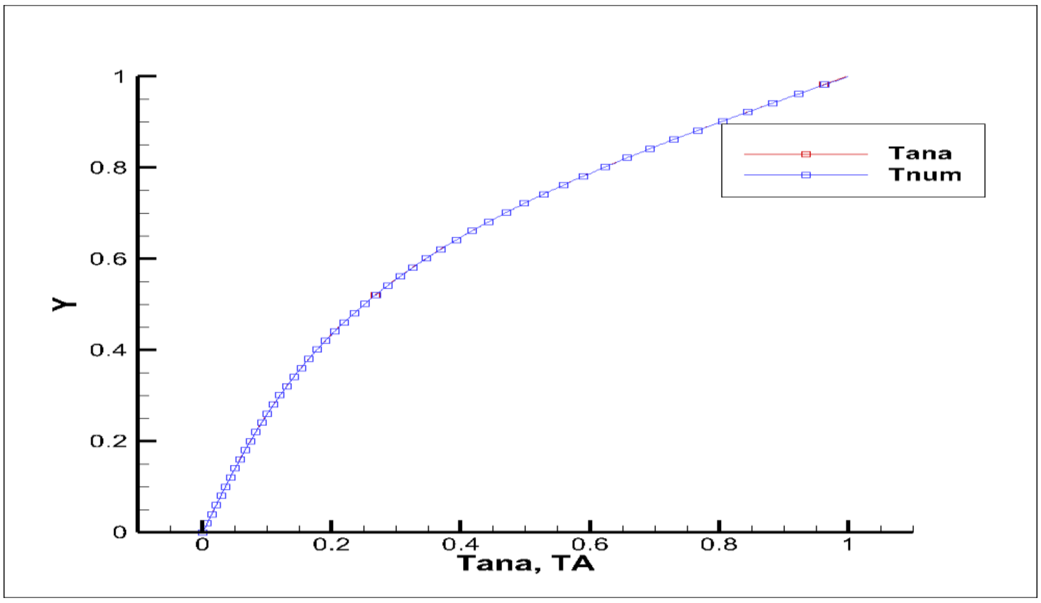
(c) Compare the temperature variation with y along mid-horizontal plane y = 0.5 from the code and the above analytical expression. Plot both temperature profiles in the same figure.

1. Ans

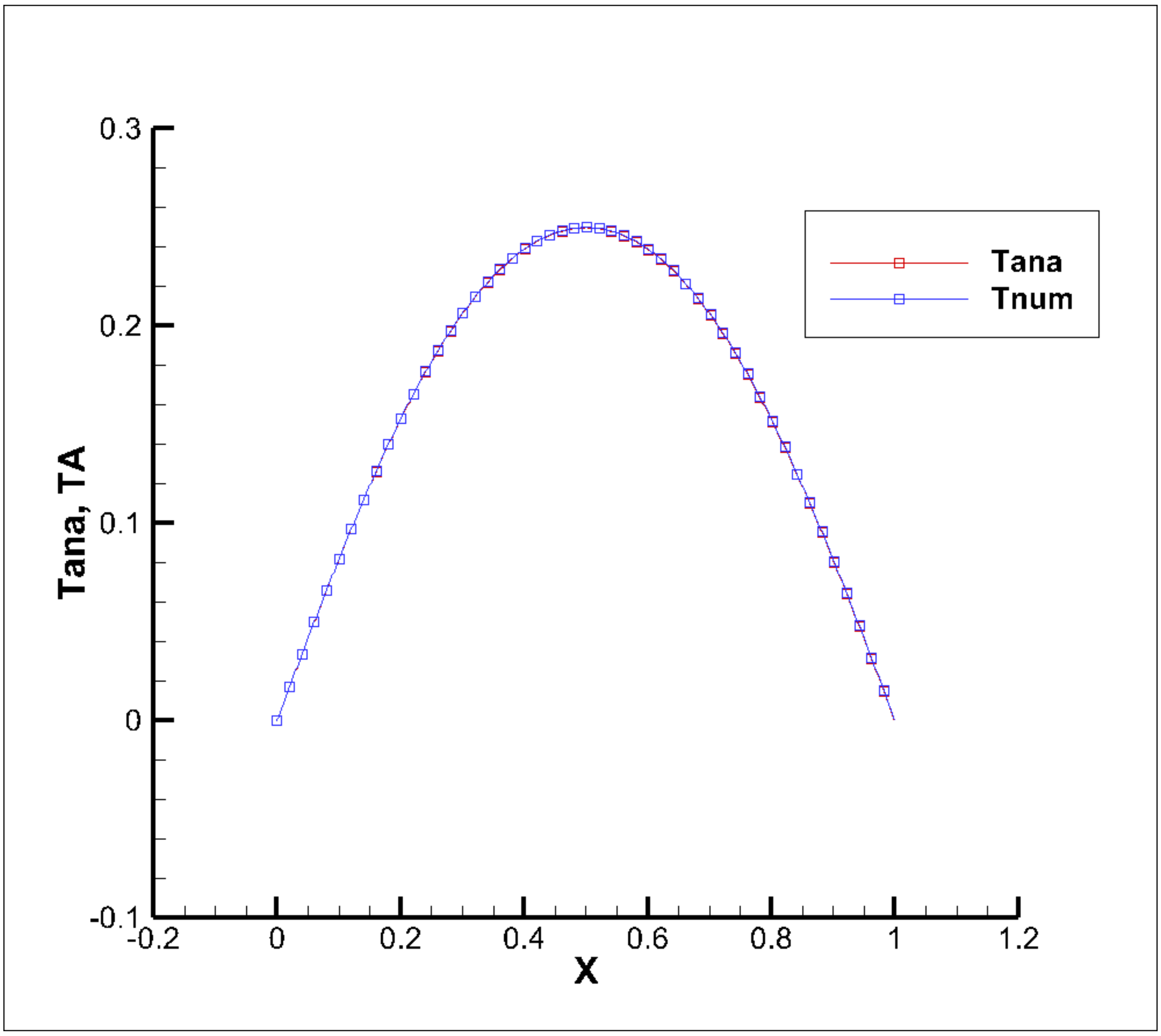




fig(a): Numerical results fig(b): Analytical results



2. Comparison of T variation along mid-vertical line (line x = 0.5)



3. Fig: Comparison of T variation along mid-vertical line (line x = 0.5)

1.1 The code of Incomplete LU decompostion

```
# include<stdio.h>
# include<math.h>


# define nj 41
# define ni 41
int L(int i,int j)
{
   int l;
   l = (i-1) *nj + (j-1);
   return l;
}
int main()
{
   double
T[ni*nj],z[ni*nj],b[ni*nj],Tem[ni*nj],TA[ni][nj],Tana[ni][nj];
   double
ap,as,aw,an,ae,dx,dy,beta,error,sum,sum1,pi=(22.0/7);
   double
lw[ni*nj],ls[ni*nj],ue[ni*nj],un[ni*nj],lp[ni*nj],mnw[ni*nj],ztem
[ni*nj],
         mse[ni*nj],p1[ni*nj],p2[ni*nj],B[ni*nj],r[ni][nj];
   double Lenght =1.0,width =1.0;
```

```
int i,j,l,k,iter=0;
dx = Lenght/(ni-1);
dy = width/(nj-1);
beta = dx/dy;
ae = 1.0;
aw = 1.0;
as = beta * beta;
an = beta * beta;
ap = -2*(1+(beta*beta));

//bounary conditions and U and L matrix
for ( i = 1; i <= ni; i++)
{
   for ( j = 1; j <=nj; j++)
   {
      l = L(i,j);
      lw[l] = aw;
      ls[l] = as;
      lp[l] = ap  - lw[l] * ue[l-nj]- ls[l] * un[l-1];
      un[l] = an/lp[l];
      ue[l] = ae/lp[l];
      z[l] =0.0;
      T[l] =0.0;
      b[l]=0.0;
```

```
//printf("%f\t",lp[l]);

if (i==1)
{
   lp[l] = 1.0;
   b[l] = 0.0;
   lw[l] = 0.0;
   ls[l] = 0.0;
   ue[l] = 0.0;
   un[l] = 0.0;
   b[l] =0.0;

}
if (j== 1)
{
   lp[l] = 1.0;
   b[l] = 0.0;
   lw[l] = 0.0;
   ls[l] = 0.0;
   ue[l] = 0.0;
   un[l] = 0.0;
   b[l] =0.0;
}
if (i==ni)
```

```
        {
            lp[l] = 1.0;
            b[l] = 0.0;
            lw[l] = 0.0;
            ls[l] = 0.0;
            ue[l] = 0.0;
            un[l] = 0.0;
            b[l] =0.0;
        }
        if (j==nj)
        {
            lp[l] = 1.0;
            b[l] = 0.0;
            lw[l] = 0.0;
            ls[l] = 0.0;
            ue[l] = 0.0;
            un[l] = 0.0;
            b[l] = 1.0;
        }

        //printf("%f\t",un[l]);
    }

}
```

```
// creating the N matrix
for ( i =1; i <ni; i++)
{
   for ( j = 1; j < nj; j++)
   {
      l = L(i,j);
      mnw[l] = lw[l]*un[l-nj];
      mse[l] = ls[l]*ue[l-1];
      //printf("%f\t",mse[l]);
   }

}



error =1.0;


while (error>1e-6)
{
   //creating b prime matrix
   for ( i = 1; i<= ni; i++)
   {
      for ( j = 1; j<= nj; j++)
   {
      l = L(i,j);
      Tem[l] = T[l];
   }


}
for ( i = 1; i<= ni; i++)
{
   for ( j = 1; j<= nj; j++)
   {
      l = L(i,j);
      B[l] = b[l] + mnw[l] * T[l-nj+1] + mse[l] * T[l+nj-1];


   }


}
//z[0] = B[0]/lp[0];
// forward substiturion
for ( i = 1; i<= ni; i++)
{
   for ( j = 1; j<= nj; j++)
   {
      sum =0.0;
      l = L(i,j);
```

```
        z[l]  = (B[l]-(ls[l] * z[l-1] + lw[l] * z[l-nj]))/lp[l];



    }


}
// backward substitution
//T[nj*ni]=z[nj*ni];
for ( i = ni; i>= 1; i--)
{


    for ( j =nj; j>=1; j--)
    {


        l = L(i,j);
        T[l] = z[l] -(un[l] * T[l+1] + ue[l] * T[l+nj]);




    }




}
error = 0.0;
for ( i = 1; i<= ni; i++)
```

```
    {
        for ( j = 1; j<= nj; j++)
        {
            l = L(i,j);
            error = error + pow((T[l]-Tem[l]),2);

        }

    }
    error = sqrt(error);
    iter = iter + 1;
    printf("iter = %d\t error = %f\n",iter,error);
}
// numerical solution
FILE*file;
file = fopen("T.plt","w");
fprintf(file," VARIABLES = \"X\",\"Y\",\"TA\"\n");
fprintf(file," ZONE F = POINT\n");
fprintf(file,"I=%d, J=%d\n",ni,nj);
    for ( i = 1; i<= ni; i++)
    {
        for ( j = 1; j<= nj; j++)
        {
            l = L(i,j);
            TA[i-1][j-1] = T[l];
            //TA[ni-1][j-1] =0.0;
            //TA[i-1][nj-1]=1.0;
            //TA[i-1][0] =0.0;
            //TA[0][j-1]=0.0;


            //printf("%f\t",TA[i-1][j-1]);
            fprintf(file,"%f\t%f\t%f\n",(i-1)*dx,(j-1)*dy,TA[i-1][j-
1]);


         }

       }
   // analytical solution
   FILE*file1;
   file1 = fopen("Tanlytical.plt","w");
   fprintf(file1," VARIABLES = \"X\",\"Y\",\"Tana\"\n");
   fprintf(file1," ZONE F = POINT\n");
   fprintf(file1,"I=%d, J=%d\n",ni,nj);

   for ( i = 0; i < ni; i++)
   {
      for ( j = 0; j <nj; j++)
      {
        Tana[i][j] =0.0;
      }


   }
   for ( i = 0; i < ni; i++)
   {
      for ( j = 0; j <nj ; j++)
      {
         for ( k = 1; k <220; k++)
         {
            Tana[i][j] = Tana[i][j] + (2.0/pi) * ((pow(-1,k+1)+1)/k ) *
sin(k*i*dx*pi)* (sinh(k*pi*j*dy)/sinh(k*pi));
         }


      }


   }
   for ( i = 0; i < ni; i++)
   {
      for ( j = 0; j <nj; j++)
      {
         printf("%f\t",Tana[i][j]);
```

```
            fprintf(file1,"%f\t%f\t%f\n",i*dx,j*dy,Tana[i][j]);
        }


    }



}
```

Chapter 2

Strongly Implicit Procedure

Question-2: Code for Strongly Implicit procedure (SIP): Consider 2D conduction problem governed by equation

$$\frac{\partial^2 T}{\partial X^2} + \frac{\partial^2 T}{\partial Y^2} = 0$$

Take a domain of width L and height W.

The boundary conditions are:

At x = 0: T = 0

At x = L: T = 0

At y = 0: T = 0

At y = W: T = Tm sin πx /L

Take L = 2 and W = 2. Take Tm = 2.

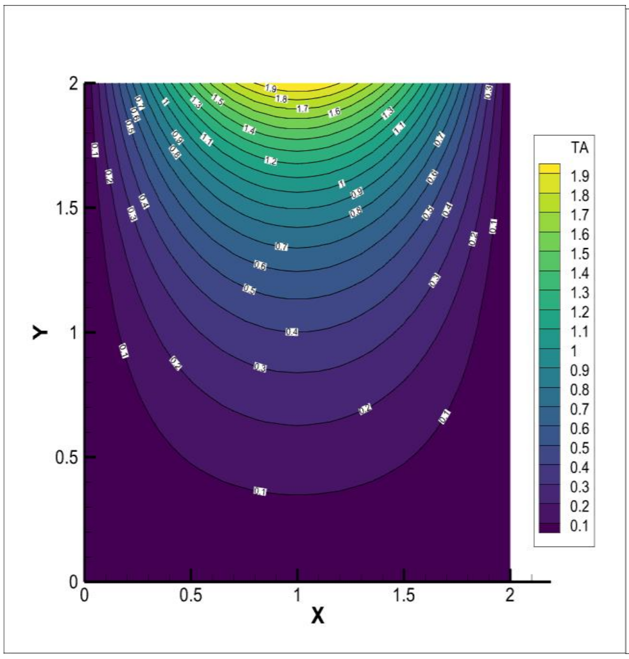
Discretise the governing and solve the obtained system of linear equations by writing code for Strongly Implicit procedure (SIP). This problem has an analytical solution which is given by series solution given by

T(x, y) = $\sum_{n=1}^{inf} T_m$ sin(nπx/L) sinh (nπy/L) /sinh(nπW/L)

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

(b) Compare the temperature variation with y along mid-vertical plane x = L/2 from the code and the above analytical expression. Plot both temperature profiles in the same figure.

(c) Compare the temperature variation with x along mid-horizontal plane y = W/2 from the code and the above analytical expression. Plot both temperature profiles in the same figure.



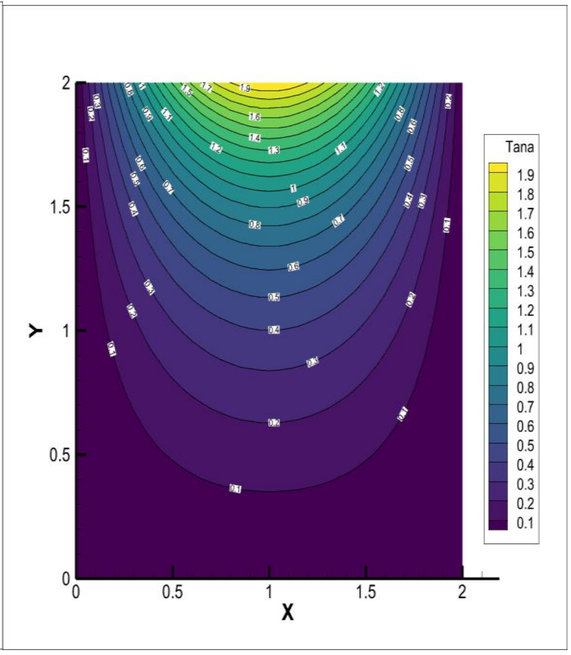


Fig :NUMRICAL

Fig:ANALYTICAL

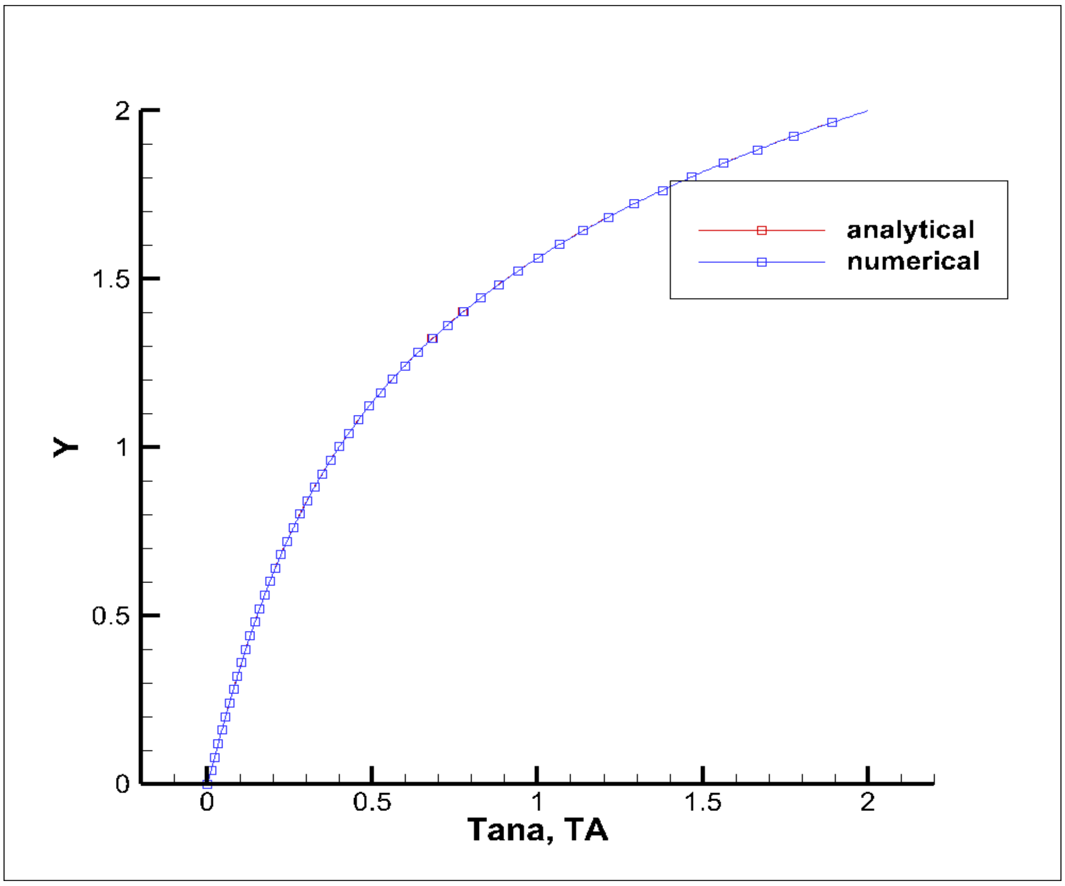


Fig: COMPARISON AT X =L//2

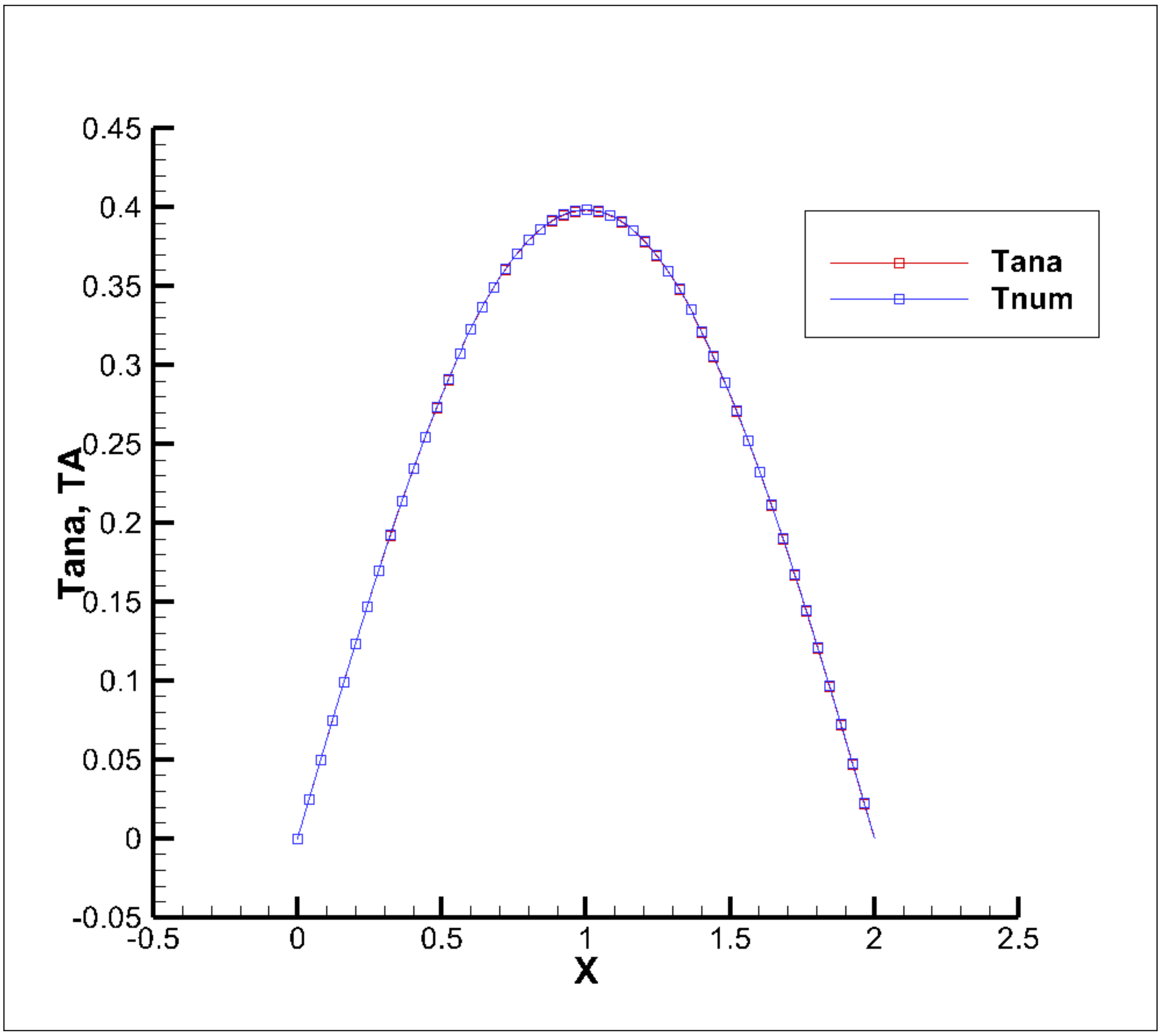


Fig: COMPARISON AT X =W//2

2.2 The code of Strongly implicit procedure(SIP)

```
# include<stdio.h>
# include<math.h>

# define nj 41
# define ni 41
int L(int i,int j)
{
   int l;
   l = (i-1) *nj + (j-1);
   return l;
}
int main()
{
   double T[ni*nj],z[ni*nj],b[ni*nj],Tem[ni*nj],TA[ni][nj],Tana[ni][nj];
   double ap,as,aw,an,ae,dx,dy,beta,error,sum,sum1,pi=(22.0/7);
   double
lw[ni*nj],ls[ni*nj],ue[ni*nj],un[ni*nj],lp[ni*nj],mnw[ni*nj],ztem[ni*nj],
         mse[ni*nj],p1[ni*nj],p2[ni*nj],B[ni*nj],r[ni][nj];
   double Lenght =2.0,width =2.0,alpla=0.8,k1[ni],k2[ni];

   int i,j,l,k,iter=0;
   dx = Lenght/(ni-1);
   dy = width/(nj-1);
```

```
beta = dx/dy;

ae = 1.0;

aw = 1.0;

as = beta * beta;

an = beta * beta;

ap = -2*(1+(beta*beta));


//bounary conditions and U and L matrix

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

{

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

   {

      l = L(i,j);

      lw[l] = aw/(1+alpla*un[l-nj]);

      ls[l] = as/(1+alpla*ue[l-1]);

      lp[l] = ap + alpla* (lw[l] * un[l-nj]+ ls[l] * ue[l-1]) - lw[l]*ue[l-nj]-ls[l]*un[l-
1];

      un[l] = (an-(alpla * lw[l]*un[l-nj]))/lp[l];

      ue[l] = (ae-(alpla * ls[l] *ue[l-1]))/lp[l];

      z[l] =0.0;

      T[l] =0.0;

      b[l]=0.0;

      //printf("%f\t",lp[l]);


      if (i==1)

      {

         lp[l] = 1.0;
```

```
    b[l] = 0.0;
    lw[l] = 0.0;
    ls[l] = 0.0;
    ue[l] = 0.0;
    un[l] = 0.0;
    b[l] =0.0;

}
if (j== 1)
{
    lp[l] = 1.0;
    b[l] = 0.0;
    lw[l] = 0.0;
    ls[l] = 0.0;
    ue[l] = 0.0;
    un[l] = 0.0;
    b[l] =0.0;
}
if (i==ni)
{
    lp[l] = 1.0;
    b[l] = 0.0;
    lw[l] = 0.0;
    ls[l] = 0.0;
    ue[l] = 0.0;
    un[l] = 0.0;
```

```
            b[l] =0.0;
        }
        if (j==nj)
        {
            lp[l] = 1.0;
            b[l] = 0.0;
            lw[l] = 0.0;
            ls[l] = 0.0;
            ue[l] = 0.0;
            un[l] = 0.0;
            b[l] = 2*sin(pi*(i-1)*dx*0.5);
        }

        //printf("%f\t",lp[l]);
    }

}



// creating the N matrix
for ( i =1; i <ni; i++)
{
    for ( j = 1; j < nj; j++)
    {
        l = L(i,j);
        mnw[l] = lw[l]*un[l-nj];
        mse[l] = ls[l]*ue[l-1];
        //printf("%f\t",mnw[l]);
    }

}



error =1.0;

while (error>1e-6)
{
    //creating b prime matrix
    for ( i = 1; i<= ni; i++)
    {
        for ( j = 1; j<= nj; j++)
        {
            l = L(i,j);
            Tem[l] = T[l];
        }

    }
    for ( i = 1; i<= ni; i++)
    {
        for ( j = 1; j<= nj; j++)
        {
            l = L(i,j);
            B[l] = b[l] + mnw[l]*(T[l-nj+1]- alpla*(T[l-nj]+T[l+1]-T[l]) )+ mse[l] *(
T[l+nj-1] - alpla*(T[l-1]+T[l+nj]-T[l]) );


        }


    }
    //z[0] = B[0]/lp[0];
    // forward substiturion
    for ( i = 1; i<= ni; i++)
    {
        for ( j = 1; j<= nj; j++)
        {
            sum =0.0;
            l = L(i,j);
            z[l]  = (B[l]-(ls[l] * z[l-1] + lw[l] * z[l-nj]))/lp[l];



        }


    }
    // backward substitution
    //T[nj*ni]=z[nj*ni];
    for ( i = ni; i>= 1; i--)
    {

        for ( j =nj; j>=1; j--)
        {
            l = L(i,j);
            T[l] = z[l]-(un[l] * T[l+1] + ue[l] * T[l+nj]);



        }



    }
    error = 0.0;
    for ( i = 1; i<= ni; i++)
    {
        for ( j = 1; j<= nj; j++)
        {
            l = L(i,j);
            error = error + pow((T[l]-Tem[l]),2);


        }


    }
    error = sqrt(error);
    iter = iter + 1;
    printf("iter = %d\t error = %f\n",iter,error);
}
// numerical solution
FILE*file;
```

```
file = fopen("Tsip.plt","w");
fprintf(file," VARIABLES = \"X\",\"Y\",\"TA\"\n");
fprintf(file," ZONE F = POINT\n");
fprintf(file,"I=%d, J=%d\n",ni,nj);
   for ( i = 1; i<= ni; i++)
   {
      for ( j = 1; j<= nj; j++)
      {
         l = L(i,j);
         TA[i-1][j-1] = T[l];
         TA[ni-1][j-1] =0.0;
         TA[i-1][nj-1]=2*sin(pi*(i-1)*dx*0.5);
         TA[i-1][0] =0.0;
         TA[0][j-1]=0.0;
         Tana[i-1][j-1]=0.0;



         printf("%f\t",TA[i-1][j-1]);
         fprintf(file,"%f\t%f\t%f\n",(i-1)*dx,(j-1)*dy,TA[i-1][j-1]);




      }


   }
// analytical solution
FILE*file1;
```

```
   file1 = fopen("Tanlyticalsip.plt","w");

   fprintf(file1," VARIABLES = \"X\",\"Y\",\"Tana\"\n");

   fprintf(file1," ZONE F = POINT\n");

   fprintf(file1,"I=%d, J=%d\n",ni,nj);

sum=0.0;

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

   {

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

      {


            k1[i] = sin((pi*i*dx)/2);

            k2[j] = sinh((pi*j*dy)/2);

            Tana[i][j] = (2* k1[i]* (k2[j]/sinh(pi*2/2)));

      }

     /// printf("new\n");

   }

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

   {

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

      {


        //printf("%f\t",Tana[i][j]);

        fprintf(file1,"%f\t%f\t%f\n",i*dx,j*dy,Tana[i][j]);

      }


   }
```

```
}
```