Assignment 1

1. **Problem**
   - **I. Jacobi iterative method**
     - i. Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 31
# define n 21
int main()
{
   double dx = 6.0/(m-1),dy = 4.0/(n-1),psi_old[n][m],psi_new[n][m],as,aw,ap,ae,an, Error;
   int iteration = 0,i,j;
   Error=1;
   ap=2.0*((1.0/pow(dx,2.0))+(1.0/pow(dy,2.0)));
   as=(1.0/pow(dy,2.0));
   aw=(1.0/pow(dx,2.0));
   ae=(1.0/pow(dx,2.0));
   an=(1.0/pow(dy,2.0));
   //boudary conditions
   for(i=0; i<n ; i++)
   {
      for(j=0 ; j<m ; j++)
      {
         if(i==0)
         {
            //bottom boundary
            if(j<=5){
               psi_new[i][j]=0;
```

```
            }
            if(j >= 6){
            psi_new[i][j]=100.0;
            }
        }
        //top boundary
        else if(i == n-1 )
        {
            psi_new[i][j]=0.0;
        }
        //left boundary
        else if(j == 0){
            psi_new[i][j]=0.0;
            }
        else{
            psi_new[i][j]=0.0;
            }
    }

}
 //jacobi iterative method
FILE *file1;
file1= fopen("error.txt","w");
fprintf(file1,"Iter\t Error\n");
 while(Error>1e-6)
{
    for( i=0; i<n ; i++)
    {
```

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

  for( i=1; i<n-1 ; i++)
  {
    for(j=1 ; j<m-1 ; j++)
    {
        psi_new[i][j]=(1/ap)*((ae*psi_old[i][j+1])+(aw*psi_old[i][j-
1])+(an*psi_old[i+1][j])+(as*psi_old[i-1][j]));
    }
  }
  for(i=0; i<n ; i++)
  {
    psi_new[i][m-1]=psi_new[i][m-2];
  }
  Error=0;
  for(int i=1; i<n-1 ; i++)
  {
        for(j=1 ; j<m-1 ; j++)
        {
            Error = Error + pow((psi_new[i][j]-psi_old[i][j]),2);

        }
  }
   Error=sqrt(Error/((m-2)*(n-2)));
  iteration=iteration+1;
```

```
        printf("Iteration= %d\t", iteration);
        printf("Error= %.9f\n", Error);
        fprintf(file1, "%d \t %.9f \n", iteration, Error);
    }

    FILE *file2;
    file2=fopen("Stream1a.plt","w");
    fprintf(file2,"VARIABLES= \"X\",\"Y\",\"PHI\"\n");
    fprintf(file2, "ZONE t=\"BLOCK1\", J=31,I=21,F= POINT \n\n");

    for( i=0; i<n ; i++)
    {
          for(j=0 ; j<m ; j++)
          {
             fprintf(file2, "%lf \t %lf \t %lf \n",j*dy,i*dx,psi_new[i][j]);
             printf("%f\t",psi_new[i][j]);
          }
    }

    return 0;

}
```

II. Point Gauss-Seidel iterative method

ii. Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 31
```

```
# define n 21


int main()
{
    double dx = 6.0/(m-1),dy = 4.0/(n-1),psi_old[n][m],psi_new[n][m],as,aw,ap,ae,an,
Error,tem[n][m];
   int iteration = 0,i,j;
   Error=1;
   ap=2.0*((1.0/pow(dx,2.0))+(1.0/pow(dy,2.0)));
   as=(1.0/pow(dy,2.0));
   aw=(1.0/pow(dx,2.0));
   ae=(1.0/pow(dx,2.0));
   an=(1.0/pow(dy,2.0));
   for(int i=0; i<n ; i++)
      {
      for(int j=0 ; j<m ; j++)
      {
         if(i==0)
         {
            if(j<=4){
               psi_new[i][j]=0;
            }
         else if(j >= 5){
            psi_new[i][j]=100.0;
            }
         }
         else if(i == n-1 ){
            psi_new[i][j]=0.0;
            }
```

```
        else if(j == 0){
            psi_new[i][j]=0.0;
            }
        else{
            psi_new[i][j]=0.0 ;
              }
            }
    }
    //point gauss-seidal method
    FILE *file1;
    file1= fopen("error.txt","w");
    fprintf(file1,"Iter\t Error\n");
     while(Error>1e-6)
    {
       Error=0;
       for( i=0; i<n ; i++)
       {
          for( j=0 ; j<m ; j++)
          {
             tem[i][j]=psi_new[i][j];
          }
          }

       for( i=1; i<n-1 ; i++)
       {
          for(j=1 ; j<m-1 ; j++)
          {
             psi_new[i][j]=(1/ap)*((ae*psi_new[i][j+1])+(aw*psi_new[i][j-
1])+(an*psi_new[i+1][j])+(as*psi_new[i-1][j]));
```

```
            Error = Error + pow((psi_new[i][j]-tem[i][j]),2);


        }
    }
    for(i=0; i<n ; i++)
    {
        psi_new[i][m-1]=psi_new[i][m-2];
    }


            Error =sqrt(Error/(m*n));
            printf("iteration %d\t",iteration);
            printf("error %.10lf\n",Error);
            fprintf(file1, "%d\t%.10f\n",iteration,Error);
            iteration++;
 }
 FILE *file2;
file2=fopen("Stream1b.plt","w");
 fprintf(file2,"VARIABLES= \"x\",\"y\",\"PHI\"\n");
 fprintf(file2, "ZONE t=\"BLOCK1\", J=31,I=21,F= POINT \n\n");

 for(int i=0; i<n ; i++)
 {
        for(int j=0 ; j<m ; j++)
        {
           fprintf(file2, "%lf\t%lf\t%lf\n",j*dy,i*dx,psi_new[i][j]);
            printf("%f\t",psi_new[i][j]);
        }
 }
```

```
 return 0;
}
```

iii. Line Gauss-Seidel iterative method (TriDiagonal Matrix Algorithm)

A. Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 31
# define n 21
int main(){
  double psi_new[n][m],B,tem[n][m],ai,bi,ci,d[m-2],P[m-2],Q[m-2],dy=4.0/(n-1), dx=6.0/(m-1), error;
  int i,j,iteration=0;
   B = (dx/dy);
   ai=-2*(1+pow(B,2));
   bi=1;
   ci=1;
    for(i=0; i<n ; i++)
   {
      for(j=0 ; j<m; j++)
      {
         if(i==0)
         {
            //bottom boundary
            if(j<=5){
               psi_new[i][j]=0;
            }
            if(j >= 6){
```

```
            psi_new[i][j]=100.0;
            }
        }
        //top boundary
        else if(i == n-1 )
        {
            psi_new[i][j]=0.0;
        }
        //left boundary
        else if(j == 0){
            psi_new[i][j]=0.0;
            }
        else{
            psi_new[i][j]=0.0;
            }
    }

 }
 FILE *file1;
 file1=fopen("error.txt","w");
 error=1;
while(error > 1e-6)
{
  error = 0.0;
    for(j=1; j<m-1;j ++)
    {
    d[0]=-(pow(B,2))*(psi_new[0][j+1]+psi_new[0][j-1]);
    P[0]=-bi/ai;
```

```
    Q[0]=d[0]/ai;

    for(i=1; i<n-1; i++){
        d[i]=-(pow(B,2))*(psi_new[i][j+1]+psi_new[i][j-1]);;

              P[i]=-(bi/(ai+ci*P[i-1]));
              Q[i]=(d[i]-ci*Q[i-1])/(ai+ci*P[i-1]);
    }
     for(i=n-2; i>0; i--){
        tem[i][j]=psi_new[i][j];
           psi_new[i][j]=P[i]*psi_new[i+1][j]+Q[i];
         error+=pow((psi_new[i][j]- tem[i][j]),2.0);
    }

 }

  for (i=0; i<n; i++){
   psi_new[i][m-1]=psi_new[i][m-2];
  }

  error=sqrt(error/((m-2)*(n-2)));
  iteration = iteration+1;
  printf("iteration %d\t",iteration);
  printf("error %.9f\n",error);
  fprintf(file1,"%d\t%.9lf\n",iteration,error);
}

 FILE *file2;
```

```
file2=fopen("tdmam1.plt","w");
 fprintf(file2,"VARIABLES= \"x\",\"y\",\"PHI\"\n");
 fprintf(file2, "ZONE t=\"BLOCK1\", J=31,I=21,F= POINT \n\n");

 for(int i=0; i<n; i++)
 {
      for(int j=0 ; j<m; j++)
      {
        fprintf(file2, "%lf\t%lf\t%lf\n",j*dy,i*dx,psi_new[i][j]);
         printf("%f\t",psi_new[i][j]);
      }
 }

 return 0;

}
```

iv. Alternating Direction Implicit method (ADI)

A. Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 31
# define n 21
int main()
{
   double tem[m][n], psi_new[m][n],P[m-2],Q[m-2],d[m-2],d1[n-2],P1[n-2],Q1[n-
2],dx=6.0/(m-1),dy=4.0/(n-1),as,aw,ap,ae,an,aj,bj,cj,ai,bi,ci ,error;
   int i,j, iteration = 0;
   as=(1.0/pow(dy,2.0));
   aw=(1.0/pow(dx,2.0));
```

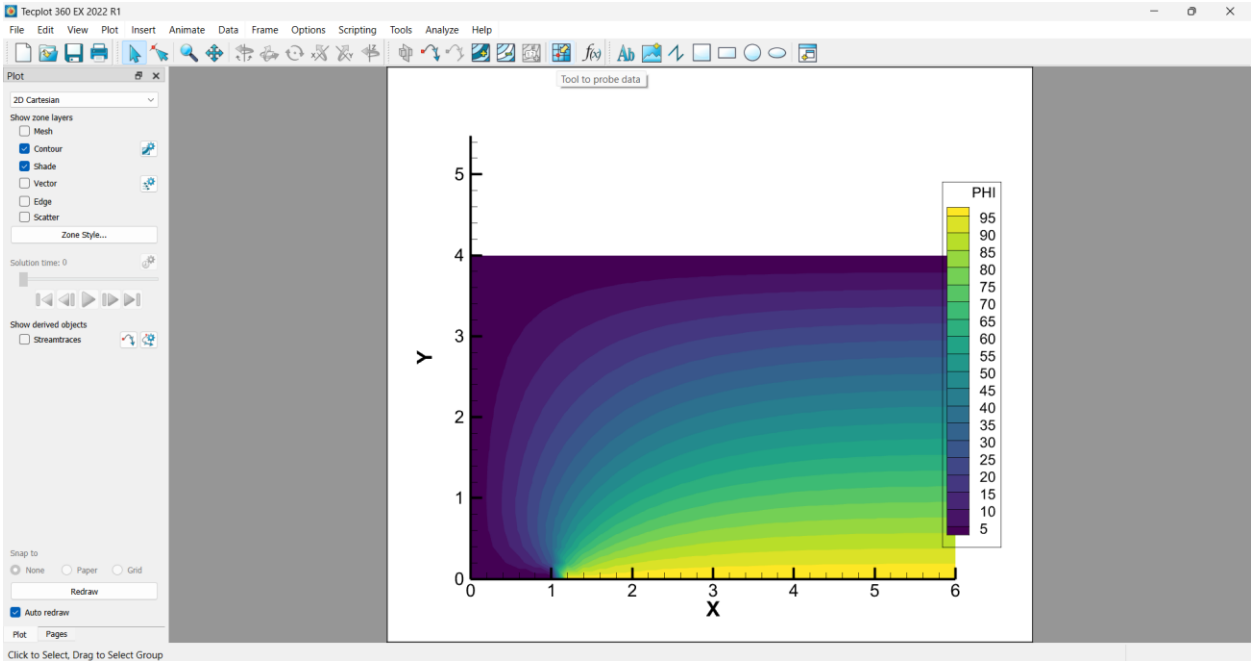
```
ap=2.0*((1.0/pow(dx,2.0))+(1.0/pow(dy,2.0)));
ae=(1.0/pow(dx,2.0));
an=(1.0/pow(dy,2.0));
ai=ap;
bi=-ae;
ci=-aw;
aj=ap;
bj=-an;
cj=-as;

for(i=0; i<m ; i++)
{
   for(j=0 ; j<n; j++)
   {
      if(j==0)
      {
         //bottom boundary
         if(i<=5){
            psi_new[i][j]=0;
         }
         if(i>= 6){
         psi_new[i][j]=100.0;
         }
      }
      //top boundary
      else if(j == n-1 )
      {
         psi_new[i][j]=0.0;
      }
      //left boundary
      else if(i == 0){
         psi_new[i][j]=0.0;
         }
      else{
         psi_new[i][j]=0.0;
         }
   }

}

FILE *file1;
```

```
    file1=fopen("error.txt","w");
    error=1;
    while(error > 1e-6)
    {



         for(i=0; i<m; i++)
         {
        for(j=0; j<n; j++)
        {
         tem[i][j]= psi_new[i][j];
       }
}
   for(j=1; j<n-1; j++)
   {
    d[0]=an*(psi_new[0][j+1]+psi_new[0][j-1]);
    P[0]=-bi/ai;
    Q[0]=d[0]/ai;

    for(i=1; i<m-1; i++)
    {
       d[i]=an*(psi_new[i][j+1]+psi_new[i][j-1]);
       P[i]=-(bi/(ai+ci*P[i-1]));
       Q[i]=(d[i]-ci*Q[i-1])/(ai+ci*P[i-1]);
    }
    for(i=m-2; i>0; i--)
    {

        psi_new[i][j]=P[i]*psi_new[i+1][j]+Q[i];

    }
    }
    for(i=1; i<m-1; i++)
    {
       d1[0]=ae*(psi_new[i+1][0]+psi_new[i-1][0]);
       P1[0]=-(bj/aj);
       Q1[0]=(d1[0])/aj;

    for(j=1; j<n-1; j++)
    {
```

```
        d1[j]=ae*(psi_new[i+1][j]+psi_new[i-1][j]);
        P1[j]=-(bj/(aj+cj*P1[j-1]));
        Q1[j]=(d1[j]-cj*Q1[j-1])/(aj+cj*P1[j-1]);
    }
    for(j=n-2; j>0; j--){

    psi_new[i][j]=P1[j]*psi_new[i][j+1]+Q1[j];
    }
    }

    for (j=0; j<n; j++)
    {
    psi_new[m-1][j]=psi_new[m-2][j] ;
    }
    error=0.0;
       for(i=0; i<m; i++)
       {
           for(j=0; j<n; j++){
           error+=pow((psi_new[i][j]-tem[i][j]),2);
        }
    }
      iteration=iteration+1 ;
      error=sqrt(error/((m-2)*(n-2)));
      printf("iteration %d\t",iteration);
      printf("error %.9f\n",error);
      fprintf(file1,"%d\t%lf\n",iteration,error);
      }
 FILE *file2;
file2=fopen("adi_1.plt","w");
fprintf(file2,"VARIABLES= \"x\",\"y\",\"PHI\"\n");
fprintf(file2, "ZONE t=\"BLOCK1\", J=31,I=21,F= POINT \n\n");
for(i = 0; i < m; i++)  {
       for(j = 0; j < n; j++)
{
    fprintf(file2,"%lf\t%lf\t%lf\n",dx*i,dy*j,psi_new[i][j]);
     printf("%f\t",psi_new[i][j]);

 }
}
return 0;
```

}

v. The output



The conclusion

Method in term speed

Adi method> Tdma>Point gauss seidel> jacobi

2. **Problem 2**

i. **Jacobi iterative method**

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 41
int main()
{
   double dx = 1.0/(m-1),dy = 1.0/(m-1),psi_old[m][m],psi_new[m][m],as,aw,ap,ae,an, Error;
    int iteration = 0,i,j;
   Error=1;
   ap=2.0*((1.0/pow(dx,2.0))+(1.0/pow(dy,2.0)));
   as=(1.0/pow(dy,2.0));
   aw=(1.0/pow(dx,2.0));
   ae=(1.0/pow(dx,2.0));
   an=(1.0/pow(dy,2.0));
   //boudary conditions

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

        psi_new[0][j]=1.0;
        psi_new[m-1][j]=1.0;
        psi_new[i][0]=1.0;
       if (j==m-1)
       {
```

```
          psi_new[i][j]=0.0;
        }
        else
        {
          psi_new[i][j]=0.0;
        }

      }

  }
  FILE *file1;
  file1= fopen("error.txt","w");
  fprintf(file1,"Iter\t Error\n");
  while (Error>1e-6)
  {
     for ( i = 0; i <m; i++)
     {
        for ( j = 0; j <m; j++)
        {
           psi_old[i][j]=psi_new[i][j];
        }

     }
     for ( i = 1; i <m-1; i++)
     {
        for ( j = 0; j <m-1; j++)
        {
           psi_new[i][j]=(1/ap)*((ae*psi_old[i][j+1])+(aw*psi_old[i][j-
1])+(an*psi_old[i+1][j])+(as*psi_old[i-1][j]));
```

```
        }


    }
    Error=0.0;
    for ( i = 0; i < m; i++)
    {
        for ( j =0; j< m; j++)
        {
            Error = Error + pow((psi_new[i][j]-psi_old[i][j]),2);
        }


    }
    Error=sqrt(Error/((m-2)*(m-2)));
    iteration=iteration+1;
    printf("Iteration= %d\t", iteration);
    printf("Error= %.9f\n", Error);
    fprintf(file1, "%d \t %.9f \n", iteration, Error);




}

FILE *file2;
file2=fopen("Stream1Ta.plt","w");
fprintf(file2,"VARIABLES= \"X\",\"Y\",\"T\"\n");
fprintf(file2, "ZONE t=\"BLOCK1\", J=41,I=41,F= POINT \n\n");
```

```
    for( i=0; i<m ; i++)
    {
        for(j=0 ; j<m ; j++)
        {
            fprintf(file2, "%lf \t %lf \t %lf \n",j*dy,i*dx,psi_new[i][j]);
            printf("%f\t",psi_new[i][j]);
        }
    }


    return 0;
}
```

ii. Point Gauss-Seidel iterative method

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 41
int main()
{
   double dx = 1.0/(m-1),dy = 1.0/(m-1),tem[m][m],psi_new[m][m],as,aw,ap,ae,an, Error;
    int iteration = 0,i,j;
   Error=1;
   ap=2.0*((1.0/pow(dx,2.0))+(1.0/pow(dy,2.0)));
   as=(1.0/pow(dy,2.0));
   aw=(1.0/pow(dx,2.0));
   ae=(1.0/pow(dx,2.0));
   an=(1.0/pow(dy,2.0));
   //boudary conditions

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

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

     psi_new[0][j]=1.0;
     psi_new[m-1][j]=1.0;
     psi_new[i][0]=1.0;
    if (j==m-1)
    {
      psi_new[i][j]=0.0;
    }
    else
    {
     psi_new[i][j]=0.0;
    }

   }

}
FILE *file1;
file1= fopen("error.txt","w");
fprintf(file1,"Iter\t Error\n");
while (Error>1e-6)
{
    Error=0.0;
   for ( i = 0; i <m; i++)
   {
      for ( j = 0; j <m; j++)
      {
         tem[i][j]=psi_new[i][j];
      }

   }
   for ( i = 1; i <m-1; i++)
   {
      for ( j = 0; j <m-1; j++)
      {
         psi_new[i][j]=(1/ap)*((ae*psi_new[i][j+1])+(aw*psi_new[i][j-
1])+(an*psi_new[i+1][j])+(as*psi_new[i-1][j]));
         Error = Error + pow((psi_new[i][j]-tem[i][j]),2);
```

```
            }

        }
        Error=sqrt(Error/((m-2)*(m-2)));
        iteration=iteration+1;
        printf("Iteration= %d\t", iteration);
        printf("Error= %.9f\n", Error);
        fprintf(file1, "%d \t %.9f \n", iteration, Error);



    }

    FILE *file2;
    file2=fopen("Stream1Tb.plt","w");
    fprintf(file2,"VARIABLES= \"X\",\"Y\",\"T\"\n");
    fprintf(file2, "ZONE t=\"BLOCK1\", J=41,I=41,F= POINT \n\n");

    for( i=0; i<m ; i++)
    {
          for(j=0 ; j<m ; j++)
          {
             fprintf(file2, "%lf \t %lf \t %lf \n",j*dy,i*dx,psi_new[i][j]);
             printf("%f\t",psi_new[i][j]);
          }
    }



    return 0;
}
```

iii. Line Gauss-Seidel iterative method (TriDiagonal Matrix Algorithm)

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 41
int main()
{
```

```
   double dx = 1.0/(m-1),dy = 1.0/(m-
1),tem[m][m],psi_new[m][m],Error,B,ai,bi,ci,d[m],P[m],Q[m];
   int iteration = 0,i,j;
   Error=1.0;
   B=(dx/dy);
   ai=-2*(1+pow(B,2));
   bi=1.0;
   ci=1.0;

   //boudary conditions

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

        psi_new[0][j]=1.0;
        psi_new[m-1][j]=1.0;
        psi_new[i][0]=1.0;
       if (j==m-1)
       {
          psi_new[i][j]=0.0;
       }
       else
       {
         psi_new[i][j]=0.0;
       }

      }

   }
   FILE *file1;
   file1= fopen("error.txt","w");
   fprintf(file1,"Iteration\t Error\n");
   while (Error>1e-6)
   {
    Error=0.0;
      for ( j = 1;  j< m; j++)
      {
        d[0]=-pow(B,2)*(psi_new[0][j+1]+psi_new[0][j-1]);
```

```
    P[0]=-bi/ai;
    Q[0]=d[0]/ai;

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

     d[i]=-pow(B,2)*(psi_new[i][j+1]+psi_new[i][j-1]);
     P[i]=-(bi/(ai+ci*P[i-1]));
     Q[i]=(d[i]-ci*Q[i-1])/(ai+ci*P[i-1]);



   }

   for ( i = m-2; i > 0; i--)
   {
     tem[i][j]=psi_new[i][j];
      psi_new[i][j]=P[i]*psi_new[i+1][j]+Q[i];
       Error = Error + pow((psi_new[i][j]-tem[i][j]),2.0);
   }
   }
   Error=sqrt(Error/((m-2)*(m-2)));
   iteration=iteration+1;
   printf("Iteration= %d\t", iteration);
   printf("Error= %.9f\n", Error);
   fprintf(file1, "%d \t %.9f \n", iteration, Error);


}

FILE *file2;
file2=fopen("Stream1Tcccv.plt","w");
fprintf(file2,"VARIABLES= \"X\",\"Y\",\"T\"\n");
fprintf(file2, "ZONE t=\"BLOCK1\", J=41,I=41,F= POINT \n\n");

for( i=0; i<m ; i++)
{
      for(j=0 ; j<m ; j++)
      {
         fprintf(file2, "%lf \t %lf \t %lf \n",j*dy,i*dx,psi_new[i][j]);
         printf("%f\t",psi_new[i][j]);
      }
```

```
    }

    return 0;

}
```

iv. Alternating Direction Implicit method (ADI)

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
# define m 41
int main()
{

   double dx = 1.0/(m-1),dy = 1.0/(m-
1),tem[m][m],psi_new[m][m],Error,B,ai,bi,ci,d[m],P[m],Q[m],d1[m],P1[m],Q1[m];
   int iteration = 0,i,j;
   Error=1.0;
   B=(dx/dy);
   ai=-2*(1+pow(B,2));
   bi=1.0;
   ci=1.0;
   //boudary conditions
   for ( i = 0; i < m; i++)
   {
      for ( j = 0; j <m; j++)
      {

        psi_new[0][j]=1.0;
        psi_new[m-1][j]=1.0;
        psi_new[i][0]=1.0;
       if (j==m-1)
       {
         psi_new[i][j]=0.0;
       }
       else
       {
        psi_new[i][j]=0.0;
       }
```

```
   }
}
 FILE *file1;
file1= fopen("error.txt","w");
fprintf(file1,"Iteration\t Error\n");
   while (Error>1e-6)
{
   for ( i = 0; i < m; i++)
   {
      for (j  = 0; j <m; j++)
      {
       tem[i][j]=psi_new[i][j];
      }

   }


   for ( j = 0.5;  j< m-1; j++)
   {
     d[0]=-pow(B,2)*(psi_new[0][j+1]+psi_new[0][j-1]);
     P[0]=-bi/ai;
     Q[0]=d[0]/ai;

   for (i = 0.5; i < m-1; i++)
   {

      d[i]=-pow(B,2)*(psi_new[i][j+1]+psi_new[i][j-1]);
      P[i]=-(bi/(ai+ci*P[i-1]));
      Q[i]=(d[i]-ci*Q[i-1])/(ai+ci*P[i-1]);



   }
     for ( i = m-2; i > 0; i--)
   {

       psi_new[i][j]=P[i]*psi_new[i+1][j]+Q[i];

   }
   }
   for ( i = 1; i < m-1; i++)
```

```
    {
      d1[0]=-pow(B,2)*(psi_new[i+1][0]+psi_new[i-1][0]);
      P1[0]=-bi/ai;
      Q1[0]=d1[0]/ai;

      for ( j = 1; j < m-1; j++)
      {
       d1[j]=-pow(B,2)*(psi_new[i][j+1]+psi_new[i][j-1]);
       P1[j]=-(bi/(ai+ci*P1[j-1]));
       Q1[j]=(d1[j]-ci*Q1[j-1])/(ai+ci*P1[j-1]);

      }
      for ( j = m-2; j >0; j--)
      {
        psi_new[i][j]=P1[j]*psi_new[i+1][j]+Q1[j];
      }


    }
    Error=0.0;
    for ( i = 0; i <m; i++)
    {
        for ( j = 0; j < m; j++)
        {
          Error = Error + pow((psi_new[i][j]-tem[i][j]),2.0);
        }

    }
    Error=sqrt(Error/((m-2)*(m-2)));
    iteration=iteration+1;
    printf("Iteration= %d\t", iteration);
    printf("Error= %.9f\n", Error);
    fprintf(file1, "%d \t %.9f \n", iteration, Error);


}
 FILE *file2;
file2=fopen("Stream1Tddd.plt","w");
fprintf(file2,"VARIABLES= \"X\",\"Y\",\"T\"\n");
fprintf(file2, "ZONE t=\"BLOCK1\", J=41,I=41,F= POINT \n\n");
```

```
    for( i=0; i<m ; i++)
    {
        for(j=0 ; j<m ; j++)
        {
            fprintf(file2, "%lf \t %lf \t %lf \n",j*dy,i*dx,psi_new[i][j]);
            printf("%f\t",psi_new[i][j]);
        }
    }
    return 0;
}
```

v. The output