ASSINGMENT 2

1. **FTCS**

CODE

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
   int n=101,iteration=0,i;
   char name[50];
   double dx=1.0/(n-1),Re=100.0,phi_new[n],phi_old[n],dt=1.0e-3,gamma
=(dt/(Re*dx*dx)), error,t=0.0;

   for(i=0;i<n;i++)
   {
      phi_new[0]=0.0;
      if (i==(n-1))
      {
         phi_new[i]=1.0;
      }
      else
      {
         phi_new[i]=0.0;

      }
   }
   //Explicit:FTCS

   while(t<50)
   {
      if((iteration+100)%100==0)
      {
         sprintf(name, "Velocity_ftcs_%f.dat",t);
         FILE*file1;
         file1=fopen(name,"w");
         for(i=0;i<n;i++)
```

```
        {

            fprintf(file1,"\n%lf\t\t%lf\n",phi_new[i],i*dx);
        }
        fclose(file1);
      }
      for(i=0;i<n;i++)
      {
        phi_old[i]=phi_new[i];
      }
      for(i=0;i<(n-1);i++)
      {
        phi_new[i]=gamma*phi_old[i+1]+(1-
2*gamma)*phi_old[i]+gamma*phi_old[i-1];
      }
      t=t+dt;
       error=0;
          for( int i=0;i<n-1;i++)
            {

             error=error+pow(((pow((phi_new[i]-phi_old[i]),2))/n),0.5);

            }
            printf("iteration=%d\t",iteration);
            printf("error=%.10f\n",error);
            t=t+dt;
            printf("TIME2a=%.10f\n",t);


            iteration++;
          }
            while(error < 1e-8);
          return 0;

        }
```

2. **Point Gauss-Seidel iterative method**

CODE

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
   int n=101,iteration=0,i;
   char name[50];
   double dx=1.0/(n-1),Re=100.0,phi_new[n],phi_old[n],dt=1.0e-2,gamma
=(dt/(Re*dx*dx)), error,t=0.0;

   for(i=0;i<n;i++)
   {
      phi_new[0]=0.0;
      if (i==(n-1))
      {
         phi_new[i]=1.0;
      }
      else
      {
         phi_new[i]=0.0;

      }
   }
   //implicit:BTCS(point gauss -seidal iterative method)

   while(t<50)
   {
      if((iteration+100)%100==0)
      {
         sprintf(name, "velocity_%f.dat",t);
         FILE*file1;
         file1=fopen(name,"w");
         for(i=0;i<n;i++)
         {
```

```
            fprintf(file1,"\n%lf\t\t%lf\n",phi_new[i],i*dx);
          }
          fclose(file1);
        }
        for(i=0;i<n;i++)
        {
          phi_old[i]=phi_new[i];
        }
        for(i=1;i<(n-1);i++)
        {

phi_new[i]=(1/(1+2*gamma))*(phi_old[i]+gamma*(phi_new[i+1]+phi_new
[i-1]));
        }

         error=0;
            for( int i=0;i<n-1;i++)
              {

                error=error+pow(((pow((phi_new[i]-phi_old[i]),2))/n),0.5);

              }
              printf("iteration=%d\t",iteration);
              printf("error=%.10f\n",error);
              t=t+dt;
              printf("TIME2a=%.10f\n",t);


              iteration++;
            }
              while(error < 1e-6);
            return 0;

          }
```

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

CODE

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
   int n=101,iteration=0,i;
   char name[50];
   double dx=1.0/(n-1),Re=100.0,phi_new[n],phi_tem[n],dt=1.0e-2,gamma
=(dt/(Re*dx*dx)), error,t=0.0,P[n],Q[n],d[n],ai,bi,ci;
   ai=-(1+2*gamma);
   bi=gamma;
   ci=gamma;
   for(i=0;i<n;i++)
   {
      phi_new[0]=0.0;
      if (i==(n-1))
      {
         phi_new[i]=1.0;
      }
      else
      {
         phi_new[i]=0.0;

      }
   }
   //implicit:BTCS(TDMA)

   while(t<50)
   {
      if((iteration+100)%100==0)
      {
         sprintf(name, "velocity_btcs_tdma%f.dat",t);
         FILE*file1;
         file1=fopen(name,"w");
         for(i=0;i<n;i++)
```

```
    {

       fprintf(file1,"\n%lf\t\t%lf\n",phi_new[i],i*dx);
    }
    fclose(file1);
}
for(i=0;i<n;i++)
{
    phi_tem[i]=phi_new[i];
}
for(i=1;i<(n-1);i++)
{
    d[0]=-phi_new[i];
    P[0]=-bi/ai;
    Q[0]=d[0]/ai;
    d[i]=-phi_new[i];
    P[i]=-bi/(ai+ci*P[i-1]);
    Q[i]=(d[i]-ci*Q[i-1])/(ai+ci*P[i-1]);
    phi_new[i]=P[i]*phi_new[i+1]+Q[i];
}

 error=0;
     for( int i=0;i<n-1;i++)
        {

         error=error+pow(((pow((phi_new[i]-phi_tem[i]),2))/n),0.5);

        }
        printf("iteration=%d\t",iteration);
        printf("error=%.10f\n",error);
        t=t+dt;
        printf("TIME2a=%.10f\n",t);


        iteration++;
     }
        while(error < 1e-6)
```

```
return 0;

        }
```

4. **Crank-Nicolson: Line Gauss-Seidel iterative method (TriDiagonal Matrix Algorithm)**

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
   int n=101,iteration=0,i;
   char name[50];
   double dx=1.0/(n-1),Re=100.0,phi_new[n],phi_old[n],dt=1.0e-2,gamma
=(dt/(Re*dx*dx)), error,t=0.0,P[n],Q[n],d[n],ai,bi,ci;
   ai=-(1+2*gamma);
   bi=gamma;
   ci=gamma;
   for(i=0;i<n;i++)
   {
      phi_new[0]=0.0;
      if (i==(n-1))
      {
         phi_new[i]=1.0;
      }
      else
      {
         phi_new[i]=0.0;

      }
   }
   //implicit:BTCS(TDMA)

   while(t<50)
   {
```

```
        if((iteration+100)%100==0)
        {
           sprintf(name, "velocity_CN_tdma%f.dat",t);
           FILE*file1;
           file1=fopen(name,"w");
           for(i=0;i<n;i++)
           {

              fprintf(file1,"\n%lf\t\t%lf\n",phi_new[i],i*dx);
           }
           fclose(file1);
        }
        for(i=0;i<n;i++)
        {
           phi_old[i]=phi_new[i];
        }
        for(i=1;i<(n-1);i++)
        {
           d[0]=-(gamma*phi_old[i+1]+(1-
2*gamma)*phi_old[i]+gamma*phi_old[i-1]);
           P[0]=-bi/ai;
           Q[0]=d[0]/ai;
           d[i]=-phi_new[i];
           P[i]=-bi/(ai+ci*P[i-1]);
           Q[i]=(d[i]-ci*Q[i-1])/(ai+ci*P[i-1]);
           phi_new[i]=P[i]*phi_new[i+1]+Q[i];
        }

         error=0;
            for( int i=0;i<n-1;i++)
              {

               error=error+pow(((pow((phi_new[i]-phi_old[i]),2))/n),0.5);

              }
              printf("iteration=%d\t",iteration);
              printf("error=%.10f\n",error);
```

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


    iteration++;
  }
    while(error < 1e-6);
 return 0;

 }
```

The output