```
//#############################################################################################
//
// Copyright, 2014, Wang Xian, Xi'an Jiaotong university
// -- couette-in.c (Oct.2014)

// -- Implicit expression of FDM for Couette Flow by Thomas scheme

//#############################################################################################


     #include <stdio.h>
     #include <stdlib.h>
     #include <malloc.h>
     #include <assert.h> //what is this?
     #include <math.h>
     #include <string.h>



int main(int argc, char *argv[])
{

//------------------------------ define varaibles ---------------------------------------------

      int   N;                     /* 节点编号 0-N  */
      int   ny;                   /* number of nodes in y direction */  // ny = N+1
      int  j;                     /* variable for loop */
      int  tt;                    // 计数器
      int  time_max;              /* maximum number of time step(loop), not time  */

      double dt;                  /* time step */
      double dy;                  /* cell dimension */

      double t;
      double E;

      double Re;                  /* Re number */
      double eps,eps1;
      double A,B;

      double *y;                  /* y[j] coordinate */
      double *u;                  /* u[j] at t */
      double *un;                 /* un[j] at t+dt */
      double *K,*K1,*B1;

      FILE  * file_dat;

      file_dat=fopen("result.plt","w");
//--------------------------------  input data ------------------------------------
   N = 20 ;
   Re = 5000 ;
   ny = N+1 ;
   time_max = 30000;
   E = 1.0 ;
//-------------------------------------------------------------------------------
    u = (double *) malloc(ny*sizeof(double));
    un = (double *) malloc(ny*sizeof(double));
    y = (double *) malloc(ny*sizeof(double));
    K = (double *) malloc(ny*sizeof(double));
    K1 = (double *) malloc(ny*sizeof(double));
    B1 = (double *) malloc(ny*sizeof(double));
  //-------------------------------------------------

    dy = 1.0 / N  ;
```

```
//  dt = 0.9 * (0.5*Re*dy*dy) ;  // 0.5*Re*dy*dy 为显式的稳定性条件
 dt = E*Re*dy*dy;

    u[0]  = 0.0 ;
    un[0] = 0.0 ;
    u[N]  = 1.0 ;
    un[N] = 1.0 ;
    y[0]  = 0.0 ;
    y[N] = 1.0 ;

        for(j=1;j<N;j++)
        {
          u[j] = 0.0 ;
          un[j] =0.0 ;
          y[j] = y[j-1] + dy ;
        }

   printf("---------dt = %f-------------\n",dt);

   A=-dt/(2.0*dy*dy*Re);
   B=1.0-2.0*A;
   B1[1]=B;

   eps = 0.0;
   eps1= 0.0;
   tt=0;
        for(t=0.0;tt<time_max;t=t+dt)
        {
            tt=tt+1;
            eps1=0.0;

            for(j=1;j<N;j++)
            {
              K[j] = ( 1.0 + 2.0*A ) * u[j] - A * ( u[j+1] + u[j-1] );
            }
              K[N-1] = K[N-1]-A;


           K1[1] = K[1];
           for(j=2;j<N;j++)
            {
              B1[j] = B - A*A/B1[j-1];
              K1[j] = K[j] - K1[j-1]*A/B1[j-1];
            }


           un[N-1] = K1[N-1]/B1[N-1];
           for(j=N-2;j>0;j--)
           {
           un[j] = ( K1[j] - A * un[j+1] ) / B1[j] ;
           }

            for(j=1;j<N;j++)
            {
              u[j]=un[j];

              eps = u[j]-y[j] ;

              if (eps < 0) eps = -eps;

              if (eps > eps1)  eps1 = eps ;          //找最大误差，最大误差小于1E-4，则收敛
            }
```

```
    if (eps1 < 0.0001) time_max = tt;

        }

  printf("---------converage_t = %d-------------\n",tt);


//-----------------output-----------------------------------------------



    fputs("VARIABLES=Y,U\n",file_dat);
    fprintf(file_dat, "ZONE I=%6d, F=POINT\n",ny);

         for (j=0; j<ny; j++)
          {
            fprintf(file_dat,"%15.9f %15.9f\n",y[j],u[j]);
         }

    fclose(file_dat);




    return 0;


}/***** end main *****/
```