```
   /*Anexo 7 - Resolucao numerica do problema P1D com condicoes de fronteira do
   tipo Dirichlet para o campo de inducao magnetica e condicoes de fronteira não
   lineares para a Vorticidade */

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

  /*Definicao da constante DimMax, que representa o numero de linhas e de
colunas
  da matriz U, definida a seguir*/
  #define DimMax 1000

  //matrizes necessarias à resolucao da eq Laplace
  double A[DimMax][DimMax]={0};
  double U[DimMax][DimMax]={0};

  //vorticidade
  double W[DimMax][DimMax]={0};

  //2 membro da discretizacao da eq laplace
  double M[DimMax]={0};
  double B[DimMax][DimMax]={0};

  /*A matriz A e factorizada em LZ, uma vez que se pretende resolver a
  equacao matricial pelo processo de factorizacao*/
  double L[DimMax][DimMax]={0};
  double Z[DimMax]={0};
  double X[DimMax]={0};

  //vector velocidade
  double V[DimMax][DimMax]={0};

  /*Inicio do programa principal*/
  int main(){

  /*Declaracao das variaveis inteiras intervenientes no programa*/
  /*Declaracao das variaveis inteiras intervenientes no programa*/
  /*J-nºsubintervalos no espaço*/
  /*N-nºsubintervalos no tempo*/
  int J,N,j,n,k;

  /*Declaracao das variaveis reais intervenientes no programa
   /*dx - passo espaço  */
  /*dt - passo tempo  */
  /* [a,b]=dominio espacial */
  /*x e t representam as coordenadas dos pontos*/
  double t,dt,T,a,b,x,dx,pi,c,e,f,r,p,g,h;



  /*Definicao da constante pi*/
  pi=4*atan(1);

   //funcao fronteira - dada
   /*double g(double t){
    return (1.0);
  }

    //funcao fronteira - dada
   double K2(double t){
```

```
    return (0);
  } */


  /*Definicao dos ficheiros onde serao guardados os resultados numericos*/
    FILE *f1;
    FILE *f2;
    FILE *f3;

  /*Pede-se ao utilizador que introduza o nº de subintervalos pretendido*/
  printf("valor de J:\n");
  scanf("%d",&J);

  /*Pede-se ao utilizador que introduza o nº de subintervalos pretendido*/
  printf("valor de N:\n");
  scanf("%d",&N);

  /*Definicao dos extremos dos intervalos*/
  a=0.0;
  b=1.0;
  T=2.5;

  /*definicao dos passos de espaco e tempo*/
  dx=(b-a)/(J+1);
  dt=T/N;


   /*Definicao  da matriz A:da sua diagonal, das diagonais acima e abaixo da
   diagonal principal */
        A[1][1]=2+dx*dx;
        A[J][J]=2+dx*dx;

   for(j=2;j<=J-1;j++){
       A[j][j]=2+dx*dx;
   }

   for(j=1;j<=J-1;j++){
       A[j][j+1]=-1.0;
       A[j+1][j]=-1.0;
   }

  /* vorticidade inicial */
   for(j=0;j<J/2;j++){
         x=a+dx*j;
        W[j][0]=0.0;
}
    for(j=J/2;j<=J+1;j++){
        W[j][0]=0.0;
}

  /* campo magnético na fronteira é dado */
   for (n=0;n<=N;n++){
       t=n*dt;
          B[0][n]=2.0+sin(pi*t);
          B[J+1][n]=0;
}

////////////////////////////////////resolucao do sistema/////////////////

     for (n=0;n<=N;n++){
           t=n*dt;
```

```
/* Determinacao da solucao numerica da eq Laplace, aplicando ao sistema AB=W
a factorizacao A=LZ  */


         /* 2ºmenbro da equacao de Helmholtz */
         for(j=2;j<=J-1;j++){
         M[j]=dx*dx*W[j][n];
         }
         M[1]=dx*dx*W[1][n]+ B[0][n];
         M[J]=dx*dx*W[J][n] +B[J+1][n];

         L[1][1]=A[1][1];
         U[1][2]=A[1][2]/L[1][1];
         Z[1]=M[1]/L[1][1];

         for (j=2;j<=J-1;j++){
         L[j][j-1]=A[j][j-1];
         L[j][j]=A[j][j]-L[j][j-1]*U[j-1][j];
         U[j][j+1]=A[j][j+1]/L[j][j];
         Z[j]=(M[j]-L[j][j-1]*Z[j-1])/L[j][j];
         }

         L[J][J-1]=A[J][J-1];
         L[J][J]=A[J][J]-L[J][J-1]*U[J-1][J];
         Z[J]=(M[J]-L[J][J-1]*Z[J-1])/L[J][J];

         B[J][n]=Z[J];

         for (j=J-1;j>=1;j--){
         B[j][n]=Z[j]-U[j][j+1]*B[j+1][n];
         }

    /* Determinacao da solucao numerica da velocidade  */
         V[0][n]=(B[0][n]-B[1][n])/dx;
         V[J+1][n]=(B[J][n]-B[J+1][n])/dx;

          for (j=1;j<=J;j++)
          {
            V[j][n]=(B[j-1][n]-B[j+1][n])/(2*dx);
          }

/* Condicao de estabilidade do esquema  */
/* for (j=0;j<=J+1;j++){
         if(V[j][n]>0){
           if(dt>=(dx/V[j][n]))
            {
            printf("o esquema nao e estavel \n","w");
            printf("dt=%f dx/V[j][n]=%f j=%d n=%d\n",dt,dx/V[j][n],j,n);
            scanf("%f",&a);
            }
         }
         if(V[j][n]<0){
          if(dt<=(dx/V[j][n]))
            {
            printf("o esquema nao e estavel \n","w");
            printf("dt=%f dx/V[j][n]=%f j=%d n=%d\n",dt,dx/V[j][n],j,n);
            scanf("%f",&a);
            }
         }
   }*/
```

```
    /* corrente de nucleacao  */
     r=2.0;
    /* alpha */
     p=1.0;

    /* definicao de uma variavel auxiliar */
    c=V[0][n];
    g=fabs(c);

//Resolução numerica da vorticidade  ////////////////////////////////////////

      if(c>0)
             {
                if(g>r)
                   {
                     W[0][n+1]= p*((g-r)/g);
                     }
                 else
                 {
                   W[0][n+1]=0;
                   }
            }
       else
           {
                if(c=0)
                {
                  W[0][n+1]=W[0][n]-(dt/dx)*W[0][n]*(V[1][n]-V[0][n]);

                  }
                else
                {
                  W[0][n+1]=W[0][n]-(dt/dx)*( V[0][n]*(W[1][n]-W[0][n])
                  + W[0][n]*(V[1][n]-V[0][n]));
                }
           }

       ////////////////////////////////////////////
          for (j=1;j<=J;j++)
           {
               f=V[j][n];
               x=a+dx*j;
               if(f>0)
            {
                  W[j][n+1]=W[j][n]-(dt/dx)*(V[j][n]*(W[j][n]-W[j-1][n])
                  + W[j][n]*(V[j][n]-V[j-1][n]));
            }
           else
           {
             if(f=0)
              {
               W[j][n+1]=W[j][n]- (dt/(2*dx))*W[j][n]*(V[j+1][n]-V[j-1][n]);
               }
               else
               {
               W[j][n+1]=W[j][n]-(dt/dx)*( V[j][n]*(W[j+1][n]-W[j][n])
               + W[j][n]*(V[j+1][n]-V[j][n]));
               }
           }
           }
           ////////////////////////////////////////////
```

```
              e=V[J+1][n];
              h=fabs(e);

            if(e>0)
               {
               W[J+1][n+1]=W[J+1][n]- (dt/dx)*(V[J+1][n]*(W[J+1][n]-W[J][n])
               + W[J+1][n]*(V[J+1][n]-V[J][n]));
               }
              else
              {
                  if(e=0)
                  {
                   W[J+1][n+1]=W[J+1][n]-(dt/dx)*W[J+1][n]*(V[J+1][n]-V[J][n]);
                   }
                   else
                  {
                     if(h>r)
                        {
                        W[J+1][n+1]= p*((h-r)/h);
                         }
                     else
                         {
                         W[J+1][n+1]=0;
                         }
                        }
              }

              ////////////////////////////////////////////
   }

 /* escrita dos valores obtidos */

 /* criação do ficheiro com os valores do campo magnetico */
f1=fopen("campo magnetico anexo 7 ","w");
  for (n=0;n<=N;n++){
          t=n*dt;
             for (j=0;j<=J+1;j++){
             x=a+dx*j;
             fprintf(f1,"%f %f %f\n",x,t,B[j][n]);
             }
           fprintf(f1,"\n");
    }
     fprintf(f1,"\n");
fclose(f1);

 /* criacao do ficheiro com os valores da velocidade */
f2=fopen(" velocidade anexo 7 ","w");
  for (n=0;n<=N;n++){
          t=n*dt;
             for (j=0;j<=J+1;j++){
             x=a+dx*j;
             fprintf(f2,"%f %f %f\n",x,t,V[j][n]);
             }
           fprintf(f2,"\n");
    }
           fprintf(f2,"\n");
fclose(f2);


/* criacao do ficheiro com os valores da vorticidade */
```

```
 f3=fopen("vorticidade anexo 7 ","w");
    for (n=0;n<=N;n++){
           t=n*dt;
              for (j=0;j<=J+1;j++){
              x=a+dx*j;
              fprintf(f3,"%f %f %f\n",x,t,W[j][n]);
              }
            fprintf(f3,"\n");
      }
      fprintf(f3,"\n");
 fclose(f3);

}
```