ANEXO 1
CÓDIGO FUENTE

Se presenta a continuación el código fuente desarrollado en FORTRAN 95, mediante la implementación del algoritmo SIMPLERT para la solución de convección forzada laminar entre conductos de sección anular. El código consta de un programa principal desde el cual se hace un llamado a una serie de módulos y subrutinas.

```
!
! U.C.V.                  CARACAS JUNIO 2004                      E.I.M
!
! PROGRAMA PARA EL ANÁLISIS VÍA NUMÉRICA DE TRANSFERENCIA DE CALOR
! POR CONVECCIÓN FORZADA ENTRE DOS CILINDROS CONCÉNTRICOS, RÉGIMEN
! LAMINAR Y PERMANENTE
!
! Realizado por: López, Estefanía y  Ponce, Marco
!
! Método Numérico:                  Volúmenes Finitos
! Esquema de Discretización:        Hibrido de diferenciación
! Algoritmo de Solución:            SIMPLERT
!
!     VARIABLES:
!
! Ur:       Componente de la velocidad en la dirección radial (m/s)
! Uz:       Componente de la velocidad en la dirección axial (m/s)
! p:        Presión (Pa)
! Re:       Número de Reynolds
! mu:       Viscosidad   cinemática (N.s/m^2)
! nu:       Viscosidad  dinámica (m^2/s)
! rho:      Densidad (Kg/m^3)
! k:        Conductividad Térmica (W/m.K)
! Cp:       Calor específico (KJ/Kg.K)
! Di:       Diámetro del cilindro interno (m)
! Do:       Diámetro del cilindro externo (m)
! Dh:       Diámetro hidráulico (m)
! DiDo:     Relación de diámetros
! Nr:       Número de volúmenes en la dirección radial
! Nz:       Número de volúmenes en la dirección axial
! Mr:       Número de divisiones en la dirección radial
! Mz:       Número de divisiones en la dirección axial
! Pr:       Número de Prandlt
! Lr:       Longitud en la dirección radial (Lr=Dh/2)
! Lz:       Longitud en la dirección axial
! alpha:    Difusividad térmica (m^2/s)
! Vi:       Velocidad de entrada (m/s^2)
! Ti:       Temperatura de entrada  (K)
! Tsi:      Temperatura superficial del cilindro interno (K)
! Tso:      Temperatura superficial del cilindro externo (K)
! deltar:   Distancia entre nodos en la dirección radial (m)
!           (deltar=Lr/Nr)
```

```
! deltaz:   Distancia entre nodos en la dirección axial  (m)
!           (deltaz=Lz/Nz)
! Deltar:   Longitud del volumen en la dirección radial (m)
! Deltaz:   Longitud del volumen en la dirección axial  (m)
! Relaxv:   Factor de relajación para las Velocidades
! Relaxp:   Factor de relajación para la presión
! Relaxcp:  Factor de relajación para la corrección de presión
! Relaxt:   Factor de relajación para la corrección de temperatura
! Tolt:     Tolerancia mínima permitida (temperatura)
! Tolv:     Tolerancia mínima permitida (velocidad)
! Tolp:     Tolerancia mínima permitida (presión)
! Max:      Número máximo de Iteraciones
! Iter:     Número de Iteraciones
! Err:      Error
! Errt:     Error mínimo permitido (temperatura)
! Errv:     Error mínimo permitido (velocidad)
! Errp:     Error mínimo permitido (presión)
! Dat:      Diferencia Adimensional de Temperatura
! Tm:       Temperatura media (K)
! FILEDAT:  Archivo de Datos
! FILERES:  Archivo de Resultados
!
! ESTRUCTURA DEL ARCHIVO DE DATOS:
!
! Cc,Nr,Nz,Lz,Di,Do,Vi,Ti,Tsi,Tso,Tolp,Tolt,Tolv,Relaxv,
! Relaxp,Relaxt,Relaxcp,Max
!
```

```
!=====================================================================
!=====================================================================
!                            PROGRAMA PRINCIPAL                      =
!=====================================================================
!=====================================================================
```

```
PROGRAM CONVEC

      USE Malla
      USE Contadores
      USE Arreglos
      USE Arreglos_T
      USE Constantes
      USE Fluido
      USE Variables_Geometricas
      USE Caracteres
      USE Err

      IMPLICIT NONE

      INTEGER :: I,J
      INTEGER :: HI,MI,SI,SSI,HF,MF,SF,SSF,Mes,Dia,Ano

      CALL GETDAT(Ano,Mes,Dia)

      CALL DATO

      CALL CALCULOS_PRELIMINARES
```

```
      CALL MATRIZ_DINAMICA

      CALL INICIALIZACION_MATRICES(Uz,Ur,P,T,Apr,Apz,Upz, &
      Tm,Um,TMM,Prm,T0,P0,U0r,U0z,r,z,b,rho,mu,alpha,k)

      CALL VECTORES_rz(r,z)

      CALL GETTIM(HI,MI,SI,SSI)
      CALL SIMPLERT(Uz,Ur,U0z,U0r,P,P0,T,T0,r,z,b,Apr,Apz, &
      rho,mu,alpha,k)

      CALL GETTIM(HF,MF,SF,SSF)

      CALL TMEDIA(T,Uz,Ur,r,TM,UM,TMM,alpha,rho,mu,rhom,mum, &
      alpham,Apr,Apz)

      CALL CALCULOS_FINALES(ReD,Pr,Um,Tm,rho,mu,alpha,rhom, &
      mum,alpham,Prm,k,T,qsim,qsom,GzD,z,NuDi,NuDo,musi,muso &
      Uz,Ur,Uzmax,Tmax)

      CALL Led_Let (ReD,Pr,Um,Tm,rho,mu,alpha,k,T,qsim,qsom, &
      z,musi,muso,Uz,Ur,Uzmax,Tmax)

      CALL SALRES(Dia,Mes,Ano,Uz,Ur,P,T,HF,MF,SF,TM,UM,TMM, &
      HI,MI,SI,r,z,rho,mu,alpha,ReD,Pr,Prm,GzD,NuDi,NuDo)

      CALL SALRES_TPV(Uz,Ur,P,T,r,z,rho,mu,alpha,Pr)

      CALL SALRES_GzD(GzD,NuDi,NuDo)

END PROGRAM CONVEC
```

SUBRUTINA ÁREAS: realiza el cálculo de las áreas para cada punto del volumen de control.

```
SUBROUTINE AREAS_VC(J)

      Use Variables_VC
      Use Constantes
      Use Variables_Geometricas

      IMPLICIT NONE

      INTEGER :: J

      Areae=Pi*(rt+rb)*deltar
      Areaw=Pi*(rt+rb)*deltar
      Areat=2*Pi*rt*deltaz
      Areab=2*Pi*rb*deltaz
      AreaUzp=Pi*(rt+rb)*deltar
      AreaUrp=2*Pi*rp*deltaz

END SUBROUTINE AREAS_VC
```

SUBRUTINA CÁLCULOS FINALES: realiza los cálculos de las tasas de flujo de calor promedio en las paredes, y los valores de algunas propiedades promedio a partir de los campos de presión velocidad y temperatura obtenidos previamente.

```
SUBROUTINE CALCULOS_FINALES(ReD,Pr,Um,Tm,rho,mu,alpha,rhom, &
mum,alpham,Prm,k,T,qsim,qsom,GzD,z,NuDi,NuDo,musi,muso,Uz,Ur,&
Uzmax,Tmax)

      Use Malla
      Use Variables_VC
      Use Variables_Geometricas
      Use Constantes

      IMPLICIT NONE

      INTEGER :: I,J,ITi,ITo
      REAL(8) :: Um(0:Mz),Tm(0:Mz),mum(0:Mz),rhom(0:Mz)
      REAL(8) :: alpham(0:Mz),GzD(0:Mz),z(0:Mz),ReD(0:Mz),
      REAL(8) :: Pr(0:Mz,0:Mr),Prm(0:Mz),k(0:Mz,0:Mr)
      REAL(8) :: T(0:Mz,0:Mr),rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
      REAL(8) :: alpha(0:Mz,0:Mr),qsim(0:Mz),qsom(0:Mz)
      REAL(8) :: muso(0:Mz),musi(0:Mz),NuDi(0:Mz),NuDo(0:Mz)
      REAL(8) :: rhoso(0:Mz),rhosi(0:Mz) Uzmax(0:Mr),Tmax(0:Mr)
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),kt,kb
      REAL(8) :: Cfi(0:Mz),Cfo(0:Mz),deltaUrsi(0:Mz)
      REAL(8) :: deltaUrso(0:Mz),Um2

!     Calculo de la tasa de flujo promedio en las paredes

      IF (CLASEIII==1) THEN

            qsi=0.0D0
            qso=0.0D0

            DO I=1,Nz

                  CALL AREAS_VC(J)

                  deltatsi=T(2*I,2)-T(2*I,0)

                  deltatso=T(2*I,2*Nr)-T(2*I,2*Nr+2)

                  kb=(k(2*I,0)+k(2*I,2))*0.5

                  kt=(k(2*I,2*Nr)+k(2*I,2*Nr))*0.5

                  qsim(I)=(deltatsi*kt*Areat)/deltaz

                  qsom(I)=(deltatso*kb*Areab)/deltaz

                  qsi=qsim(I)+qsi

                  qso=qsom(I)+qso
```

```
            END DO
      ELSE
            IF (CLASEIII==2) THEN
                  qso=0.0D0
                  DO I=1,Nz
                        CALL AREAS_VC(J)
                        deltatso=T(2*I,2*Nr)-T(2*I,2*Nr+2)
                        kt=(k(2*I,2*Nr)+k(2*I,2*Nr))*0.5
                        qsom(I)=(deltatso*kb*Areab)/deltaz
                        qso=qsom(I)+qso
                  END DO
            ELSE
                  IF (CLASEIII==3) THEN
                        qsi=0.0D0
                        DO I=1,Nz
                              CALL AREAS_VC(J)
                              deltatsi=T(2*I,2)-T(2*I,0)
                              kb=(k(2*I,0)+k(2*I,2))*0.5
                              qsim(I)=(deltatsi*kt*Areat)/deltaz
                              qsi=qsim(I)+qsi
                        END DO
                  END IF
            END IF
            END IF
      REDmed=0.0D0
      Tmed=0.0D0
      Umed=0.0D0

      deltaTsi=(2*Tsi-T(2*I,2*Nr))
      deltaTso=(2*Tso-T(2*I,2))

      DO I=1,Nz
```

```
            DO J=1,Nr
                        ReD(I)=(Um(I)*rhom(I)*Dh)/mum(I)
            Pr(2*I,2*J)=mu(2*I,2*J)/(alpha(2*I,2*J)*rho(2*I,2*J))
                        Prm(I)=mum(I)/(alpham(I)*rhom(I))
                        GzD(I)=(ReD(I)*Prm(I)*Dh)/z(2*I+1)
                        kb=(k(2*I,0)+k(2*I,2))*0.5
                        kt=(k(2*I,2*Nr)+k(2*I,2*Nr))*0.5
                        Tb=(T(2*I,0)+T(2*I,2))*0.5
                        Tt=(T(2*I,2*Nr)+T(2*I,2*Nr))*0.5
                        NuDi(I)=(qsim(I)*Dh)/(kb*(Tb-Tm(I)))
                        NuDo(I)=(qsom(I)*Dh)/(kt*(Tt-Tm(I)))
            END DO
      END DO
!     CALCULO DE ALGUNAS PROPIEDADES PROMEDIO
      DO I=1,Nz
            Tmed=Tm(I)+Tmed
            ReDmed=ReD(I)+ReDmed
            Umed=Um(I)+Umed
            Prmed=Prm(I)+Prmed
      END DO
      Tmed=Tmed/Nz
      ReDmed=ReDmed/Nz
      Umed=Umed/Nz
      Prmed=Prmed/Nz
END SUBROUTINE CALCULOS_FINALES
```

SUBRUTINA CÁLCULOS PRELIMINARES: realiza los cálculos de las dimensiones de la malla, los radios de los cilindros, el diámetro hidráulico, las condiciones de entrada (dinámicas y térmicas) y las propiedades termofísicas dependiendo del fluido.

```
SUBROUTINE CALCULOS_PRELIMINARES
      USE Malla
```

```
      USE Constantes
      USE Variables_Geometricas
      USE Fluido

      IMPLICIT NONE

!     Calculo de PI=arcoseno(sen(90º))

      Pi=2*Dasin(1.0D0)

!     Calculo de la dimensión de la malla  (Mr,Mz)

      Mr=2*Nr+3
      Mz=2*Nz+3

      IJ=MAX0(Mr,Mz)

!     Calculo de los radio del cilindro interno y externo

      Di=RD*Do
      Ri=Di/2.0D0
      Ro=Do/2.0D0

!     Calculo del Diámetro Hidráulico

      Dh=(Do-Di)

      Lr=(Ro-Ri)

!     Calculo de los delta de la malla (deltar=deltaz)

      deltaz=Lz/(2*Nz)

      deltar=Lr/(2*Nr)

!     Condiciones de entrada Prescritas (para Temperatura y
!     para flujo de Calor)

      IF (claseIII==1) THEN

            Tsi=Xsi
            Tso=Xso

            T0i=(Ti+Tsi+Tso)/3.0D0
            Tempmax=DMAX1(Ti,Tsi,Tso)
            Tempmin=DMIN1(Ti,Tsi,Tso)

      ELSE

            IF (claseIII==2) THEN

                  qsi=Xsi
                  Tso=Xso

            ELSE

                  IF (claseIII==3) THEN
```

```
                        Tsi=Xsi
                        qso=Xso

                  ELSE

!                       CLASEIII=4

                        qsi=Xsi
                        qso=Xso

                  END IF

            END IF


      END IF

!     Condiciones de propiedades termofisicas de entrada según
!     el fluido

      IF (CLASEI==1) THEN

            rhoi=Arho*(Ti-273.15)**5+Brho*(Ti-273.15)**4 &
            +Crho*(Ti-273.15)**3+Drho*(Ti-273.15)**2 &
             +Erho*(Ti-273.15)+Frho

            mui=Amu/DEXP(Bmu/Ti)

            nui=mui/rhoi

            ki=(Ti**Ak)*DEXP((Bk-(Ck*Ti)))

            Cpi=ACp*(Ti)**6+BCp*(Ti)**5+CCp*(Ti)**4  &
            +DCp*(Ti)**3+ECp*(Ti)**2+FCp*(Ti)+GCp

            alphai=ki/(rhoi*Cpi)

            Pri=mui/(rhoi*alphai)

      ELSE

            rhoi=Arho+Brho*Ti

            nui=(10D0**(10D0**(Anu+Bnu*DLOG10(Ti)))  &
            -0.7D0)/(1000**2)

            mui=nui*rhoi

            ki=Ak+Bk*Ti

            Cpi=ACp+BCp*Ti

            alphai=ki/(rhoi*Cpi)

            Pri=mui/(rhoi*alphai)

      END IF
```

```
! Calculo de Re o velocidad de entrada según el dato introducido

      IF (claseII==1) THEN

            ReDi=Xi
            Vi=ReDi*nui/Dh

      ELSE

            Vi=Xi
            ReDi=Vi*Dh/nui

      END IF

END SUBROUTINE CALCULOS_PRELIMINARES
```

SUBRUTINA C CONTORNO T: asigna los valores de las condiciones de controno de la temperatura.

```
SUBROUTINE C_CONTORNO_T(T)

      Use Malla
      Use Constantes

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: T(0:Mz,0:Mr)


      DO J=1,Nr

            T(2*Nz+2,2*J)=Tempmax

            T(0,2*J)=Ti

            T(2*Nz,2*J)=T(Nz-2,2*J)

            T(2*Nz+2,2*J)=T(Nz-2,2*J)

      END DO

      DO I=1,Nz

            T(2*I,0)=2*Tsi-T(2*I,2)

            T(2*I,2*Nr+2)=2*Tso-T(2*I,2*Nr)

      END DO

END SUBROUTINE C_CONTORNO_T
```

SUBRUTINA C_CONTORNO_Uz: asigna los valores de las condiciones de controno de la componente u_z de la velocidad.

```
SUBROUTINE C_CONTORNO_Uz(Uz)

      Use Malla
      Use Constantes

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: Uz(0:Mz,0:Mr)


!     SECCION DE ENTRADA Y SECCION DE SALIDA

      DO J=1,Nr

            Uz(1,2*J)=Vi

            Uz(2*Nz+1,2*J)=Uz(2*Nz-1,2*J)

            Uz(2*Nz+3,2*J)=Uz(2*Nz-1,2*J)

      END DO


!     PARED INTERNA Y PARED EXTERNA DE LOS CILINDROS

      DO I=1,Nz

            Uz(2*I+1,2*Nr+2)=-Uz(2*I+1,2*Nr)

            Uz(2*I+1,0)=-Uz(2*I+1,2)

      END DO

END SUBROUTINE C_CONTORNO_Uz
```

SUBRUTINA CLASE_I: selecciona el archivo de datos a abrir dependiendo del tipo de fluido.

```
SUBROUTINE Clase_I

      Use Constantes
      Use Caracteres
      Use Fluido

111   FORMAT(///F30.15///F30.15///F30.15///F30.15///F30.15///F30.15,&
      ///F30.15,///F30.15///F30.15///F30.15///F30.15///F30.15, &
```

```
      ///F30.15,///F30.15///F30.15,///F30.15///F30.15///F30.15, &
      ///F30.15///F30.15)

      IF (claseI==1) THEN

            OPEN(6,FILE='Agua.txt',STATUS='OLD')
            Fluid='Agua'

      END IF

      IF (claseI==2) THEN

            OPEN(6,FILE='O 228.txt',STATUS='OLD')
            Fluid='OIL 228'

      END IF

      IF (claseI==3) THEN

            OPEN(6,FILE='P 22.txt',STATUS='OLD')
            Fluid='PUROLUB 22'

      END IF

      IF (claseI==4) THEN

            OPEN(6,FILE='P 150.txt',STATUS='OLD')
            Fluid='PUROLUB 150'

      END IF

      IF (claseI==5) THEN

            OPEN(6,FILE='P 320.txt',STATUS='OLD')
            Fluid='PUROLUB 320'

      END IF

      READ(6,111) Arho,Brho,Crho,Drho,Erho,Frho,Anu,Bnu, &
      Amu,Bmu,Ak,BkCk,ACp,BCp,CCp,DCp,ECp,FCp,GCp

END SUBROUTINE Clase_I
```

SUBRUTINA CRRV: corrige los valores del campo de velocidades, utilizando el campo de correccion de presiones, cálculado utilizando la Subrutina PRES

```
SUBROUTINE CRRV(Uz,Ur,P,Apr,Apz,rho,mu,r)

      USE Malla
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE
```

```
      REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),rho(0:Mz,0:Mr)
      REAL(8) :: mu(0:Mz,0:Mr)Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)
      REAL(8) :: P(0:Mz,0:Mr),r(0:Mr)
      INTEGER :: I,J


      DO I=1,Nz-1

            DO  J=1,Nr

                  CALL VC_Uz(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  dpz=AreaUzp/Azp

                  Uz(2*I+1,2*J)=Uz(2*I+1,2*J)+dpz*(Pw-Pe)

                  CALL C_CONTORNO_Uz(Uz)

            END DO

      END DO


      DO I=1,Nz

            DO J=1,Nr-1

                  CALL VC_Ur(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  dpr=AreaUrp/Arp

                  Ur(2*I,2*J+1)=Ur(2*I,2*J+1)+dpr*(Pb-Pt)

            END DO

      END DO
```

END SUBROUTINE CRRV

SUBRUTINA CVRG: verifica la convergencia de los campos de presión, velocidad y temperatura.

```
SUBROUTINE CVRG(Uz,Ur,P,U0z,U0r,P0,T,T0)

      USE Malla
      USE Err

      IMPLICIT NONE
```

```
      INTEGER :: I,J
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr)
      REAL(8) :: P(0:Mz,0:Mr),T(0:Mz,0:Mr)
      REAL(8) :: U0z(0:Mz,0:Mr),U0r(0:Mz,0:Mr)
      REAL(8) :: T0(0:Mz,0:Mr),P0(0:Mz,0:Mr)

      ErrUr1 =0.0D0
      ErrUz1 =0.0D0
      ErrP1 =0.0D0
      ErrV =0.0D0
      ErrT1 =0.0D0

      DO I=0,Nz+1

            DO  J=0,Nr+1

                  ErrT =DABS(T(2*I,2*J)-T0(2*I,2*J))
                  ErrT1=DMAX1(ErrT1,ErrT)
                  ErrUr =DABS(Ur(2*I,2*J+1)-U0r(2*I,2*J+1))
                  ErrUr1=DMAX1(ErrUr,ErrUr1)
                  ErrUz =DABS(Uz(2*I+1,2*J)-U0z(2*I+1,2*J))
                  ErrUz1=DMAX1(ErrUz,ErrUz1)
                  ErrV=DMAX1(ErrUz1,ErrUr1,ErrV)
                  ErrPP =DABS(P(2*I,2*J)-P0(2*I,2*J))
                  ErrP1=DMAX1(ErrPP,ErrP1)

            END DO

      END DO

      ErrP =ErrP1
      ErrV=ErrV
      ErrT =ErrT1
```

END SUBROUTINE CVRG

SUBRUTINA DATO: realiza la lectura de los valores del archivo de datos.

SUBROUTINE DATO

```
      USE Malla
      USE Constantes
      USE Fluido
      USE Variables_Geometricas
      USE Caracteres

      IMPLICIT NONE

      CHARACTER*1 :: S

      S=CHAR(178)

44          FORMAT(/)
            WRITE(*,33)
```

```
            WRITE(*,33)
33          FORMAT(/,/,/)
            WRITE(*,*)(S,Q=1,78)
            WRITE(*,44)
            WRITE(*,11)
11          FORMAT(25X,'ESCUELA DE INGENIERIA, &
            MECANICA',/,/,37x,'U.C.V',/, &
            /,31X,'PROGRAMA "CONVEC"')
            WRITE(*,22)
22          FORMAT()
            WRITE(*,* )(S,Q=1,78)
            WRITE(*,44)
55          FORMAT(X,'Presione [ENTER] para continuar')
            WRITE(*,55)
            WRITE(*,22)
            READ(*,*)
66          FORMAT(/,/)
77          FORMAT(///I4///I4///F30.15///F30.15///F30.15)
88          FORMAT(///F30.15///F30.15///F30.15///F30.15///F30.15, &
            ///F30.15///F30.15///F30.15)
99          FORMAT(///I1///I1///I1///I1///F30.15///F30.15///F30.15, &
            ///F30.15///F30.15///F30.15///F30.15///I10///F30.15, &
            ///F30.15///F30.15)

      OPEN(8,FILE='Res.txt')
      OPEN(10,FILE='Grafico.plt')
      OPEN(11,FILE='GzD_NuD.plt')
      WRITE(*,22)
      OPEN(5,FILE='Dat.txt',STATUS='OLD')
      READ(5,99) claseI,claseII,claseIII,claseIV,Xi,Xso,Xsi, &
      Ti,TolP,TolT,TolV,Max,RelaxP,RelaxT,RelaxV

      CALL Clase_I

      OPEN(7,FILE='Malla.txt',STATUS='OLD')
      READ(7,77) Nz,Nr,Do,RD,Lz

      WRITE(*,66)

END SUBROUTINE DATO
```

SUBRUTINA ENER: resuelve la ecuación de energía.

```
SUBROUTINE ENER(Uz,Ur,T,r,z,alpha,Apr,Apz,k)

      USE Malla
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
```

```
      REAL(8) :: FAEE,FAWW,FATT,FABB
      REAL(8) :: AWW(IJ),AEE(IJ),ATT(IJ),ABB(IJ),Ap(IJ)
      REAL(8) :: X(IJ),RA(IJ)
      REAL(8) :: T(0:Mz,0:Mr),k(0:Mz,0:Mr)
      REAL(8) :: alpha(0:Mz,0:Mr),Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr),ktm,kbm,Atm,Abm

!-----------------------------------------------
!              BARRIDO OESTE-ESTE               -
!-----------------------------------------------

      DO J=1,Nr

            DO I=1,Nz

                  CALL VC_T(I,J,Ur,Uz,r,T,alpha,Apr,Apz)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS

                  Fe=Areae*Uze
                  Fw=Areaw*Uzw
                  Ft=Areat*Urt
                  Fb=Areab*Urb

                  ! COEFICIENTES DIFUSIVOS

                  De=alphae*Areae/deltaz
                  Dw=alphaw*Areaw/deltaz
                  Dt=alphat*Areat/deltar
                  Db=alphab*Areab/deltar

                  ! ESQUEMA DE DISCRETIZACION

                  CALL ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

                  AEE(I)=FAEE
                  AWW(I)=FAWW
                  ATT(I)=FATT
                  ABB(I)=FABB

                  Ap(I)=AEE(I)+AWW(I)+ATT(I)+ABB(I)

                  RA(I)=ATT(I)*TTT+ABB(I)*TBB

            END DO

            IF (CLASEIII==2) THEN

                  CALL FLUJO_CALOR (k,T,r,ktm,kbm,Atm,Abm)

                  Tsi=T(2*I,2)+((qsi*deltaz)/(2*ktm*Abm))

            ELSE
                  IF (CLASEIII==3) THEN
```

```
                        CALL FLUJO_CALOR(k,T,r,ktm,kbm,Atm,Abm)
                                    Tso=T(2*I,2*Nr) &
                                    +((qso*deltaz)/(2*ktm*Atm))

                                    Tsi=T(2*I,2) &
                                    +((qsi*deltaz)/(2*ktm*Abm))

                  ELSE

                        IF (CLASEIII==4) THEN

                              CALL FLUJO_CALOR (k,T,r,ktm, &
                              kbm,Atm,Abm)

                              Tso=T(2*I,2*Nr) &
                              +((qso*deltaz)/(2*ktm*Atm))

                              Tsi=T(2*I,2) &
                              +((qsi*deltaz)/(2*ktm*Abm))

                        END IF

                  END IF

            END IF

            RA(1)=RA(1)+AWW(1)*T(0,2*J)

            RA(Nz)=RA(Nz)+AEE(Nz)*T(2*Nz+2,2*J)

            CALL TRIDIAG(Nz,AWW,Ap,AEE,RA,X)

            DO I=1,Nz

                  T(2*I,2*J)=T(2*I,2*J)+RelaxT*(X(I)-T(2*I,2*J))

            END DO

      END DO

      CALL C_CONTORNO_T(T)

!-----------------------------------------------
!            BARRIDO ARRIBA-ABAJO -
!-----------------------------------------------

      DO I=1,Nz

            DO  J=1,Nr

                  CALL VC_T(I,J,Ur,Uz,r,T,alpha,Apr,Apz)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS
                  Fe=Areae*Uze
```

```
      Fw=Areaw*Uzw
      Ft=Areat*Urt
      Fb=Areab*Urb

      ! COEFICIENTES DIFUSIVOS

      De=alphae*Areae/deltaz
      Dw=alphaw*Areaw/deltaz
      Dt=alphat*Areat/deltar
      Db=alphab*Areab/deltar

      ! ESQUEMA DE DISCRETIZACION

      CALL ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

      AEE(J)=FAEE
      AWW(J)=FAWW
      ATT(J)=FATT
      ABB(J)=FABB

      Ap(J)=AEE(J)+AWW(J)+ATT(J)+ABB(J)

      RA(J)=AEE(J)*TEE+AWW(J)*TWW

END DO

IF (CLASEIII==2) THEN

      CALL FLUJO_CALOR (k,T,r,ktm,kbm,Atm,Abm)

            Tsi=T(2*I,2)+((qsi*deltaz)/(2*ktm*Abm))

ELSE

      IF (CLASEIII==3) THEN

            CALL FLUJO_CALOR (k,T,r,ktm,kbm,Atm,Abm)

            Tso=T(2*I,2*Nr+((qso*deltaz)/(2*ktm*Atm))

            Tsi=T(2*I,2)+((qsi*deltaz)/(2*ktm*Abm))

      ELSE

            IF (CLASEIII==4) THEN

                  CALL FLUJO_CALOR (k,T,r,ktm,kbm &
                  Atm,Abm)

                  Tso=T(2*I,2*Nr) &
                  +((qso*deltaz)/(2*ktm*Atm))

                  Tsi=T(2*I,2) &
                  +((qsi*deltaz)/(2*ktm*Abm))

            END IF
      END IF
```

```
            END IF

                  RA(1)=RA(1)+ABB(1)*(2*Tsi-T(2*I,2))

                  RA(Nr)=RA(Nr)+ATT(Nr)*(2*Tso-T(2*I,2*Nr))

                  CALL TRIDIAG(Nr,ABB,Ap,ATT,RA,X)

                  DO J=1,Nr

                        T(2*I,2*J)=T(2*I,2*J) &
                        +RelaxT*(X(J)-T(2*I,2*J))

                  END DO

      END DO

      CALL C_CONTORNO_T(T)
```

END SUBROUTINE ENER

SUBRUTINA ESQUEMA DISCRETIZACIÓN: selecciona y calcula los coeficientes según el esquema de discretización asignado en el archivo de datos (esquemas híbrido y de ley de potencia).

```
SUBROUTINE ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

      USE Variables_VC
      USE Constantes
      IMPLICIT NONE

      REAL(8) :: FAEE,FAWW,FATT,FABB

      IF (ClaseIV==1) THEN

!     ESQUEMA HIBRIDO DE DISCRETIZACION

      FAEE=DMAX1(0.0D0,-Fe,(De-0.5*Fe))
      FAWW=DMAX1(0.0D0,Fw,(Dw+0.5*Fw))
      FATT=DMAX1(0.0D0,-Ft,(Dt-0.5*Ft))
      FABB=DMAX1(0.0D0,Fb,(Db+0.5*Fb))

      ELSE

!     ESQUEMA LEY DE POTENCIA (POWER-LAW)

      FAEE=DMAX1(0.0D0,(1-0.1*DABS(Fe/De))**5)+DMAX1(Fe,0.0D0)
      FAWW=DMAX1(0.0D0,(1-0.1*DABS(Fw/Dw))**5)+DMAX1(Fw,0.0D0)
      FATT=DMAX1(0.0D0,(1-0.1*DABS(Ft/Dt))**5)+DMAX1(Ft,0.0D0)
      FABB=DMAX1(0.0D0,(1-0.1*DABS(Fb/Db))**5)+DMAX1(Fb,0.0D0)

      END IF
```

```
END SUBROUTINE ESQUEMA_DISCRETIZACION
```

SUBRUTINA FLUJO_CALOR: calcula las temperaturas de las superficies de los cilindros cuando la entrada de datos viene dada por un valor prescrito de flujo de calor en una o en ambas superficies.

```
SUBROUTINE FLUJO_CALOR (k,T,r,ktm,kbm,Atm,Abm)

      Use Constantes
      Use Malla
      Use Variables_VC
      Use Variables_Geometricas

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: T(0:Mz,0:Mr),k(0:Mz,0:Mr),r(0:Mr),ktm,kbm
      REAL(8) :: kso(0:Mz),ksi(0:Mz),Atm,Abm,rtm,rbm

      kbm=0.0D0
      ktm=0.0D0

      DO I=1,Nz

            DO J=1,Nr

                  kso(I)=(k(2*I,2*J+2)+k(2*I,2*J))*0.5
                  ksi(I)=(k(2*I,2*J)+k(2*I,2*J-2))*0.5

                  kbm=ksi(I)+kbm
                  ktm=kso(I)+ktm

            END DO

      END DO

      rtm=r(Nr+1)
      rbm=r(1)

      Atm=2*Pi*rtm*deltaz
      Abm=2*Pi*rbm*deltaz

      ktm=1
      kbm=1

END SUBROUTINE FLUJO_CALOR
```

SUBRUTINA INICIALIZACIÓN MATRICES: inicializa todas las matrices en cero.

```
SUBROUTINE INICIALIZACION_MATRICES(Uz,Ur,P,T,Apr,Apz,Upz, &
Tm,Um,TMM,Prm,T0,P0,U0r,U0z,r,z,b,rho,mu,alpha,k)

            Use Malla
            Use Constantes
            Use Arreglos_T

            IMPLICIT NONE

            INTEGER :: I,J
            REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),T(0:Mz,0:Mr)
            REAL(8) :: TP(0:Mz,0:Mr),P(0:Mz,0:Mr) ,Um(0:Mz)
            REAL(8) :: Upz(0:Mz,0:Mr),Apr(0:Mz,0:Mr),z(0:Mz)
            REAL(8) :: Apz(0:Mz,0:Mr),k(0:Mz,0:Mr)
            REAL(8) :: rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
            REAL(8) :: alpha(0:Mz,0:Mr),Prm(0:Mz),Tm(0:Mz)
            REAL(8) :: TMM(0:Mz,0:Mr),T0(0:Mz,0:Mr)
            REAL(8) :: b(0:Mz,0:Mr),P0(0:Mz,0:Mr)
            REAL(8) :: U0r(0:Mz,0:Mr),U0z(0:Mz,0:Mr),r(0:Mr)

      !     INICIALIZACION DE MATRICES

            DO I=0,Mz

                  DO J=0,Mr

                        Uz(I,J)=0.0D0
                        Ur(I,J)=0.0D0
                        U0z(I,J)=0.0D0
                        U0r(I,J)=0.0D0
                        P0(I,J)=0.0D0
                        P(I,J)=0.0D0
                        T(I,J)=Ti
                        T0(I,J)=0.0D0
                        TMM(I,J)=0.0D0
                        Apr (I,J)=0.0D0
                        Apz (I,J)=0.0D0
                        Upz(I,J)=0.0D0
                        b(I,J)=0.0D0
                        mu(I,J)=0.0D0
                        nu(I,J)=0.0D0
                        rho(I,J)=0.0D0
                        alpha(I,J)=0.0D0
                        k(I,J)=0.0D0
                        Cp(I,J)=0.0D0
                        Pr(I,J)=0.0D0


                  END DO

            END DO


            DO I=0,Mr

                  r(I)=0.0D0
```

```
            END DO

            DO J=0,Mz

                  Tm(J)=0.0D0
                  Um(J)=0.0D0
                  ReD(J)=0.0D0
                  mum(J)=0.0D0
                  rhom(J)=0.0D0
                  alpham(J)=0.0D0
                  Prm(J)=0.0D0
                  z(J)=0.0D0
                  qsim(J)=0.0D0
                  qsom(J)=0.0D0
                  GzD(J)=0.0D0
                  NuDi(J)=0.0D0
                  NuDo(J)=0.0D0
                  Wsi(J)=0.0D0
                  Wso(J)=0.0D0

            END DO
```

END SUBROUTINE INICIALIZACION_MATRICES

SUBRUTINA M_CVRG: se encarga de guardar el valor anterior de velocidad presión y temperatura con el fin de compararlas posteriormente (en la subrutina de convergencia) con los valores obtenidos en la nueva iteración.

```
SUBROUTINE M_CVRG(Uz,U0z,Ur,U0r,P,P0,T,T0)

            Use Malla
            IMPLICIT NONE

            INTEGER :: I,J
            REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),T(0:Mz,0:Mr)
            REAL(8) :: P(0:Mz,0:Mr) U0r(0:Mz,0:Mr)
            REAL(8) :: U0z(0:Mz,0:Mr),T0(0:Mz,0:Mr)
            REAL(8) :: P0(0:Mz,0:Mr)

            DO  I=0,Nz+1

                  DO  J=0,Nr+1

                        U0z(2*I+1,2*J)=Uz(2*I+1,2*J)
                        U0r(2*I,2*J+1)=Ur(2*I,2*J+1)
                        P0(2*I,2*J)=P(2*I,2*J)
                        T0(2*I,2*J)=T(2*I,2*J)

                  END DO

            END DO
```

END SUBROUTINE M_CVRG

SUBRUTINA MATRÍZ DINÁMICA: en esta subrutina se dimensionan las matrices y vectores dinámicamente.

```
SUBROUTINE MATRIZ_DINAMICA

      Use Malla
      USE Arreglos
      USE Arreglos_T

      IMPLICIT NONE

      ALLOCATE(Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),P(0:Mz,0:Mr), &
      U0z(0:Mz,0:Mr),U0r(0:Mz,0:Mr),P0(0:Mz,0:Mr),Apr(0:Mz,0:Mr), &
      Apz(0:Mz,0:Mr),T(0:Mz,0:Mr),T0(0:Mz,0:Mr),Upz(0:Mz,0:Mr), &
      TMM(0:Mz,0:Mr),Um(0:Mz),Tm(0:Mz),Prm(0:Mz),z(0:Mz),r(0:Mr), &
      b(0:Mz,0:Mr),rho(0:Mz,0:Mr),nu(0:Mz,0:Mr),mu(0:Mz,0:Mr), &
      k(0:Mz,0:Mr),alpha(0:Mz,0:Mr),Cp(0:Mz,0:Mr),ReD(0:Mz), &
      Pr(0:Mz,0:Mr),rhom(0:Mz),mum(0:Mz),alpham(0:Mz),qsim(0:Mz), &
      qsom(0:Mz),GzD(0:Mz),NuDi(0:Mz),NuDo(0:Mz),musi(0:Mz), &
      muso(0:Mz),Wsi(0:Mz),Wso(0:Mz),Uanal(0:Mr),Unum(0:Mr), &
      Uzmax(0:Mz),Tmax(0:Mz))

END SUBROUTINE MATRIZ_DINAMICA
```

MODULOS: se declaran todas las variables clasificadas según su función y su naturaleza.

```
MODULE Arreglos

REAL(8), DIMENSION(:,:) ,ALLOCATABLE :: Uz,Ur,P,U0z,U0r,P0,Apr,Apz
REAL(8), DIMENSION(:,:) ,ALLOCATABLE :: T,T0T,TMM,b,rho,mu
REAL(8), DIMENSION(:,:) ,ALLOCATABLE :: Upz,alpha,k
REAL(8), DIMENSION(:)   ,ALLOCATABLE :: TM,UM,Prm,z,r

END MODULE Arreglos

MODULE Arreglos_T

REAL(8), DIMENSION(:,:) ,ALLOCATABLE :: nu,Cp,Pr
REAL(8), DIMENSION(:)   ,ALLOCATABLE :: ReD,rhom,mum,alpham,qsim,
REAL(8), DIMENSION(:)   ,ALLOCATABLE :: qsom,GzD,NuDi,NuDo,musi
REAL(8), DIMENSION(:)   ,ALLOCATABLE :: muso,Wsi,Wso,Uanal,Unum,
REAL(8), DIMENSION(:)   ,ALLOCATABLE :: Uzmax,Tmax

END MODULE Arreglos_T

MODULE Caracteres

CHARACTER(15) :: FILEREST,FILERES,FILEDAT,FLUIDAT,FLUID
CHARACTER( 1) :: MERI,MERF
```

```
END   MODULE Caracteres

MODULE Malla

INTEGER :: Mr,Mz,IJ,Nr,Nz,Mrz

END   MODULE Malla

MODULE Variables_Geometricas

REAL(8) :: Lr,Lz,deltar,deltaz,dtheta,st,se,sb,sp,sw,Ro,Ri,Di
REAL(8) :: Do,Dh,RD

END   MODULE Variables_Geometricas

MODULE Fluido

REAL(8) :: Arho,Brho,Crho,Drho,Erho,Frho,Amu,Bmu,Ak,Bk,Ck, &
           ACp,BCp,CCp,DCp,ECp,FCp,GCp,Anu,Bnu

END   MODULE Fluido

MODULE Constantes

INTEGER :: Max,claseI,claseII,claseIII,claseIV,Q,ITamax,JTamax
INTEGER :: IUzmax,JUzmax,INuDi,INuDo,ICfi,ICfo,ICfmax,INuDmax
REAL(8) :: Vi,RelaxV,Relaxp,Relaxt,deltaTsi,deltaTso,deltaP
REAL(8) :: Tolp,Tolt,Tolv,Tempmax,Tempmin,Prmed,Pi,ledCf,letNuD
REAL(8) :: ledVmax,letTamax,ReDmed
REAL(8) :: Tso,Tsi,qsi,qso,Ti,Xso,Xsi,Xi,T0i,ReDi,Pri,Tmed,Umed
REAL(8) :: alphai,ki,Cpi,nui,mui,rhoi,mud,rhod,Redd,Prd,alphad

END   MODULE Constantes

MODULE Contadores

INTEGER :: Iter,KK,ZZ

END   MODULE Contadores

MODULE Err

REAL(8) :: ErrP,ErrUz,ErrUr,ErrPP,ErrP1,ErrUz1
REAL(8) :: ErrUr1,ErrV,ErrT,ErrT1

END   MODULE Err

MODULE Variables_VC

REAL(8) :: alphae,alphaw,alphat,alphab,alphaTT,alphap,alphaBB
REAL(8) :: rhoe,rhow,rhot,rhob,rhoTT,rhop,rhoBB,rhoEE,rhoWW
REAL(8) :: mue,muw,mut,mub,muTT,mup,muBB,muWW,muEE
REAL(8) :: Ure,Urw,Urt,Urb,UrTT,Urp,UrBB,UrWW,UrEE
```

```
REAL(8) :: Uze,Uzw,Uzt,Uzb,UzTT,Uzp,UzBB,UzWW,UzEE
REAL(8) :: Te,Tw,Tt,Tb,TTT,Tp,TBB,TWW,TEE,alphaWW,alphaEE
REAL(8) :: re,rw,rt,rb,rTT,rp,rBB,rWW,rEE
REAL(8) :: Pe,Pw,Pt,Pb,PTT,Pp,PBB,PWW,PEE
REAL(8) :: Aze,Azw,Azt,Azb,AzTT,Azp,AzBB,AzWW,AzEE
REAL(8) :: Are,Arw,Art,Arb,ArTT,Arp,ArBB,ArWW,ArEE
REAL(8) :: Fe,Fw,Ft,Fb,De,Dw,Dt,Db,dEE,dWW,dTT,dBB,dpr,dpz
REAL(8) :: Areae,Areaw,Areat,Areab,Areawe,Areabt,AreaUzp,AreaUrp

END   MODULE Variables_VC
```

SUBRUTINA PRES: resuelve las ecuaciones de presión del algoritmo SIMPLERT.

```
SUBROUTINE PRES(Uz,Ur,Apr,Apz,P,r,z,b,rho)

      USE Malla
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: AWW(IJ),AEE(IJ),ATT(IJ),ABB(IJ),Ap(IJ)
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),P(0:Mz,0:Mr)
      REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)
      REAL(8) :: b(0:Mz,0:Mr),rho(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr)
      REAL(8) :: RA(IJ),X(IJ)

!-----------------------------------------------
!               BARRIDO OESTE-ESTE -
!-----------------------------------------------
      DO J=1,Nr

            DO I=1,Nz

                  CALL VC_P(I,J,Ur,Uz,r,P,Apz,Apr,rho)

                  b(2*I,2*J)=(rhow*Uzw-rhoe*Uze)*deltar  &
                             +(rhob*Urb-rhot*Urt)*deltaz

            END DO

      END DO

      DO J= 1,Nr

            DO I= 1,Nz

                  CALL VC_P(I,J,Ur,Uz,r,P,Apz,Apr,rho)

                  CALL AREAS_VC(J)
```

```
      de=Areae/Aze
      dw=Areaw/Azw
      dt=Areat/Art
      db=Areab/Arb


      IF(I.EQ.1) THEN

            AWW(I)= 0.0D0
            AEE(I)=rhoe*Areae*de

      ELSE
            AWW(I)=rhow*Areaw*dw
            AEE(I)=rhoe*Areae*de

      END IF

      IF (J.EQ.1) THEN

            ATT(I)=rhot*Areat*dt
            ABB(I)=0.0D0
            RA(I)=b(2*I,2*J)+ATT(I)*PTT

      ELSE

            IF(J.EQ.Nr) THEN

                  ATT(I)=0.0D0
                  ABB(I)=rhob*Areab*db
                  RA(I)=b(2*I,2*J)+ABB(I)*PBB

            ELSE

                  IF(J.GT.1.AND.J.LT.Nr) THEN

                        ATT(I)=rhot*Areat*dt
                        ABB(I)=rhob*Areab*db

                  RA(I)=b(2*I,2*J)+ABB(I)*PBB+ATT(I)*PTT

                  END IF

            END IF

      END IF

      Ap(I)=ATT(I)+ABB(I)+AEE(I)+AWW(I)

END DO

CALL TRIDIAG(Nz,AWW,Ap,AEE,RA,X)

DO I=1,Nz

      P(2*I,2*J)=P(2*I,2*J)+RelaxP*(X(I)-P(2*I,2*J))

END DO
```

```
      END DO

!-----------------------------------------------
!             BARRIDO SUR-NORTE         -
!-----------------------------------------------

      DO I=1,Nz

            DO J=1,Nr

                  CALL VC_P(I,J,Ur,Uz,r,P,Apz,Apr,rho)

                  CALL AREAS_VC(J)

                  de=Areae/Aze
                  dw=Areaw/Azw
                  dt=Areat/Art
                  db=Areab/Arb

                  IF(J.EQ.1) THEN

                        ATT(J)=rhot*Areat*dt
                        ABB(J)=0.0D0

                  ELSE
                        IF(J.EQ.Nr) THEN

                              ATT(J)=0.0D0
                              ABB(J)=rhob*Areab*db

                        ELSE

                              IF(J.GT.1.AND.J.LT.Nr) THEN

                                    ATT(J)=rhot*Areat*dt
                                    ABB(J)=rhob*Areab*db

                              END IF

                        END IF

                  END IF

                  IF(I.EQ.1) THEN

                        AEE(J)=rhoe*Areae*de
                        AWW(J)=0.0D0
                        RA(J)=b(2*I,2*J)+AEE(J)*PEE

                  ELSE

                        AEE(J)=rhoe*Areae*de
                        AWW(J)=rhow*Areaw*dw
                        RA(J)=b(2*I,2*J)+AEE(J)*PEE+AWW(J)*PWW

                  END IF
```

```
                  Ap(J)=ATT(J)+ABB(J)+AEE(J)+AWW(J)

            END DO

                  CALL TRIDIAG(Nr,ABB,Ap,ATT,RA,X)

            DO J=1,Nr

                  P(2*I,2*J)=P(2*I,2*J)+RelaxP*(X(J)-P(2*I,2*J))

            END DO

      END DO

END SUBROUTINE PRES
```

SUBRUTINA PROPIEDADES_T: calcula las propiedades termofísicas en función de la temperatura.

```
SUBROUTINE PROPIEDADES_T(rho,mu,alpha,T,k)

      Use Malla
      Use Arreglos_T
      Use Fluido
      Use Constantes

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: alpha(0:Mz,0:Mr),mu(0:Mz,0:Mr),rho(0:Mz,0:Mr)
      REAL(8) :: T(0:Mz,0:Mr),k(0:Mz,0:Mr)

!     Cálculo de las propiedades del fluido en función de la
!     temperatura


      DO J=0,Nr+1

            DO I=0,Nz+1

                  IF (T(2*I,2*J)<=0) THEN

                        T(2*I,2*J)=Ti

                  END IF

                  IF (CLASEI==1) THEN

                        rho(2*I,2*J)=Arho*(T(2*I,2*J)-273.15)**5  &
                        +Brho*(T(2*I,2*J)-273.15)**4  &
                        +Crho*(T(2*I,2*J)-273.15)**3  &
                        +Drho*(T(2*I,2*J)-273.15)**2  &
                        +Erho*(T(2*I,2*J)-273.15)+Frho
```

```
                        mu(2*I,2*J)=Amu/DEXP(Bmu/T(2*I,2*J))

                        nu(2*I,2*J)=mu(2*I,2*J)/rho(2*I,2*J)

                        k(2*I,2*J)=(T(2*I,2*J)**Ak)*DEXP((Bk-
                        (Ck*T(2*I,2*J))))


                        Cp(2*I,2*J)=ACp*(T(2*I,2*J))**6 &
                        +BCp*(T(2*I,2*J))**5  &
                        +CCp*(T(2*I,2*J))**4+DCp*(T(2*I,2*J))**3  &
                        +ECp*(T(2*I,2*J))**2+FCp*(T(2*I,2*J))+GCp


            alpha(2*I,2*J)=k(2*I,2*J)/(rho(2*I,2*J)*Cp(2*I,2*J))


            Pr(2*I,2*J)=mu(2*I,2*J)/(rho(2*I,2*J)*alpha(2*I,2*J))

                  ELSE

                        rho(2*I,2*J)=Arho+Brho*T(2*I,2*J)


      nu(2*I,2*J)=(10D0**(10D0**(Anu+Bnu*DLOG10(T(2*I,2*J))))-
0.7D0)/(1000**2)

                        mu(2*I,2*J)=nu(2*I,2*J)*rho(2*I,2*J)

                        k(2*I,2*J)=Ak+Bk*T(2*I,2*J)

                        Cp(2*I,2*J)=ACp+BCp*T(2*I,2*J)


            alpha(2*I,2*J)=k(2*I,2*J)/(rho(2*I,2*J)*Cp(2*I,2*J))


            Pr(2*I,2*J)=mu(2*I,2*J)/(rho(2*I,2*J)*alpha(2*I,2*J))


                  END IF

            END DO

      END DO

END SUBROUTINE PROPIEDADES_T
```

SUBRUTINA PSEDUr: calcula los valores de pseudo-velocidad en la componente radial.

```
SUBROUTINE PSEDUr(Uz,Ur,r,z,P,rho,mu,Apr,Apz)

      USE Malla
```

```
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),Apr(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr),P(0:Mz,0:Mr),rho(0:Mz,0:Mr)
      REAL(8) :: AEE,AWW,ATT,ABB,Ap,Apz(0:Mz,0:Mr),mu(0:Mz,0:Mr)

!-----------------------------------------------
!         VELOCIDAD EN DIRECCION Y 'Urp'         -
!-----------------------------------------------

      DO J=1,Nr

            DO I=1,Nz

                  CALL VC_Ur(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS

                  Fe=rhoe*Uze*Areae*(1/rp)
                  Fw=rhow*Uzw*Areaw*(1/rp)
                  Ft=rhot*Urt*Areat*(1/rt)
                  Fb=rhob*Urb*Areab*(1/rb)

                  ! COEFICIENTES DIFUSIVOS

                  De=mue*Areae/deltaz*(1/rp)
                  Dw=muw*Areaw/deltaz*(1/rp)
                  Dt=mut*Areat/deltar*(1/rt)
                  Db=mub*Areab/deltar*(1/rb)

                  ! ESQUEMA DE DISCRETIZACION

                  CALL ESQUEMA_DISCRETIZACION(AEE,AWW,ATT,ABB)

                  Ap=AEE+AWW+ATT+ABB

                  Apr(2*I,2*J+1)=Ap

Ur(2*I,2*J+1)=(AEE*UrEE+AWW*UrWW+ATT*UrTT+ABB*UrBB)/Apr(2*I,2*J+1)

            END DO

      END DO


END SUBROUTINE PSEDUr
```

SUBRUTINA PSEDUz: calcula los valores de pseudo-velocidad en la componente axial.

```
SUBROUTINE PSEDUz(Uz,Ur,r,z,P,rho,mu,Apr,Apz)

      USE Malla
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),Apz(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr),P(0:Mz,0:Mr),rho(0:Mz,0:Mr)
      REAL(8) :: AEE,AWW,ATT,ABB,Ap,Apr(0:Mz,0:Mr),mu(0:Mz,0:Mr)

!-----------------------------------------------
!         VELOCIDAD EN DIRECCION X 'Uzp'         -
!-----------------------------------------------

      DO J=1,Nr

            DO I=1,Nz

                  CALL VC_Uz(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS

                  Fe=rhoe*Uze*Areae
                  Fw=rhow*Uzw*Areaw
                  Ft=rhot*Urt*Areat
                  Fb=rhob*Urb*Areab

                  ! COEFICIENTES DIFUSIVOS

                  De=mue*Areae/deltaz
                  Dw=muw*Areaw/deltaz
                  Dt=mut*Areat/deltar
                  Db=mub*Areab/deltar

                  ! ESQUEMA DE DISCRETIZACION

                  CALL ESQUEMA_DISCRETIZACION(AEE,AWW,ATT,ABB)

                  Ap=AEE+AWW+ATT+ABB

                  Apz(2*I+1,2*J)=Ap

Uz(2*I+1,2*J)=(AEE*UzEE+AWW*UzWW+ATT*UzTT+ABB*UzBB)/Apz(2*I+1,2*J)
```

```
            END DO

      END DO

      CALL C_CONTORNO_Uz(Uz)

END SUBROUTINE PSEDUz
```

SUBRUTINA Led_Let: aplica el criterio de discernimiento entre flujo en desarrollo y flujo desarrollado.

```
SUBROUTINE Led_Let (ReD,Pr,Um,Tm,rho,mu,alpha,k,T,qsim,qsom,z, &
musi,muso,Uz,Ur,Uzmax,Tmax)

      Use Malla
      Use Variables_VC
      Use Variables_Geometricas
      Use Constantes

      IMPLICIT NONE

      INTEGER :: I,J,ITi,ITo
      REAL(8) :: Um(0:Mz),Tm(0:Mz),mum(0:Mz),rhom(0:Mz),alpham(0:Mz)
      REAL(8) :: ReD(0:Mz),Pr(0:Mz,0:Mr),Prm(0:Mz),k(0:Mz,0:Mr)
      REAL(8) :: GzD(0:Mz),z(0:Mz),T(0:Mz,0:Mr),qsom(0:Mz)
      REAL(8) :: rhosi(0:Mz), rho(0:Mz,0:Mr),mu(0:Mz,0:Mr),
      REAL(8) :: alpha(0:Mz,0:Mr),qsim(0:Mz), muso(0:Mz),musi(0:Mz)
      REAL(8) :: NuDi(0:Mz),NuDo(0:Mz),rhoso(0:Mz)
      REAL(8) :: Uzmax(0:Mz),Tmax(0:Mz),Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr)
      REAL(8) :: Cfi(0:Mz),Cfo(0:Mz),deltaUrsi(0:Mz),deltaUrso(0:Mz)
      REAL(8) :: TMM(0:Mz,0:Mr),kt,kb,Um2
      DO I=1,Nz

            DO J=1,Nr

                  IF (Uzmax(I)<Uz(2*I+1,2*J)) THEN

                        Uzmax(I)=Uz(2*I+1,2*J)

                  END IF

            END DO

      END DO

      deltaTsi=(2*Tsi-T(2*I,2*Nr))
      deltaTso=(2*Tso-T(2*I,2))

      DO I=1,Nz+1

            DO J=1,Nr

                  IF(TM(I).EQ.Tsi) THEN
```

```
                  TMM(I,J)=TMM(I-1,J)
            ELSE
                  TMM(I,J)=(Tsi-T(2*I,2*J))/(Tsi-TM(I))
            END IF
      END DO
END DO
DO I=1,Nz
      DO J=1,Nr
            IF (Tmax(I)<TMM(I,J)) THEN
                  Tmax(I)=TMM(I,J)
            END IF
      END DO
END DO
DO I=1,Nz
      DO J=1,Nr
            musi(I)=0.5*(mu(2*I,0)+mu(2*I,2))
            muso(I)=0.5*(mu(2*I,2*Nr+2)+mu(2*I,2*Nr))
            rhosi(I)=0.5*(rho(2*I,0)+rho(2*I,2))
            rhoso(I)=0.5*(rho(2*I,2*Nr+2)+rho(2*I,2*Nr))
            deltaUrsi(I)=Ur(2*I,3)-Ur(2*I,1)
            deltaUrso(I)=Ur(2*I,2*Nr+3)-Ur(2*I,2*Nr+1)
            Um2=Um(I)**2
      Cfi(I)=(-2*musi(I)*deltaUrsi(I))/(rhosi(I)*Um2*deltar)
      Cfo(I)=(-2*muso(I)*deltaUrso(I))/(rhoso(I)*Um2*deltar)
            kb=(k(2*I,0)+k(2*I,2))*0.5
            kt=(k(2*I,2*Nr)+k(2*I,2*Nr))*0.5
            NuDi(I)=qsi*Dh/kb
            NuDo(I)=qso*Dh/kt
      END DO
```

```
      END DO
!     CALCULO DE LA LONGITUD DE ENTRADA TERMICA Y DINAMICA
      Do I=1,Nz
            IF (Uzmax(I)<=0.99*Uzmax(I-1)) EXIT
      END DO
      IUzmax=I
      Do I=1,Nz
            IF (Tmax(I)<=0.99*Tmax(I-1)) EXIT
      END DO
      ITamax=I
      Do I=1,Nz
            IF (NuDi(I)<=1.05*NuDi(I-1)) EXIT
      END DO
      INuDi=I
      Do I=1,Nz
            IF (NuDo(I)<=1.05*NuDo(I-1)) EXIT
      END DO
      INuDo=I
      INuDmax=IMAX0(INuDi,INuDo)
      Do I=1,Nz
            IF (Cfi(I)>=1.05*Cfi(I+1)) EXIT
      END DO
      ICfi=I
      Do I=1,Nz
            IF (Cfo(I)>=1.05*Cfo(I+1)) EXIT
      END DO
      ICfo=I
      ICfmax=IMAX0(ICfi,ICfo)
      LetNuD=Z(ITamax)
```

```
      LedCf=z(ICfi)

END SUBROUTINE Led_Let
```

SUBRUTINA SALRES: estructura el archivo de resultados.

```
SUBROUTINE SALRES(Dia,Mes,Ano,Uz,Ur,P,T,HF,MF,SF,TM,UM,TMM,HI,MI, &
SI,r,z,rho,mu,alpha,ReD,Pr,Prm,GzD,NuDi,NuDo)

      USE Malla
      USE Err
      USE Constantes
      USE Contadores
      USE Fluido
      USE Variables_Geometricas
      USE Caracteres

      IMPLICIT NONE

      INTEGER :: I,J,Mes,Dia,Ano,TIME
      INTEGER :: HI,MI,SI,SSI,HF,MF,SF,SSF,HT,MT,STT
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),P(0:Mz,0:Mr),GzD(0:Mz)
      REAL(8) :: T(0:Mz,0:Mr),Um(0:Mz),Tm(0:Mz),TMM(0:Mz,0:Mr)
      REAL(8) :: NuDi(0:Mz),T(0:Mz,0:Mr),Um(0:Mz),Tm(0:Mz
      REAL(8) :: TMM(0:Mz,0:Mr),NuDi(0:Mz),z(0:Mz),r(0:Mr)
      REAL(8) :: rho(0:Mz,0:Mr),mu(0:Mz,0:Mr),NuDo(0:Mz)
      REAL(8) :: alpha(0:Mz,0:Mr),ReD(0:Mz),Pr(0:Mz,0:Mr),Prm(0:Mz)

      HT=HF-HI
      MT=MF-MI

      STT=SF-SI
      TIME=INT((HT*3600)+(MT*60)+STT)

      IF (HI .GT. 12) THEN

            MERI = 'p'
            HI=HI-12

            ELSE

            MERI = 'a'

      END IF

      IF (HF .GT. 12) THEN

            MERF = 'p'
            HF=HF-12

      ELSE

            MERF = 'a'
```

```
      END IF

      WRITE(8,440)Dia,Mes,Ano
99    FORMAT()
      WRITE(8,99)
      WRITE(8,99)
      WRITE(8,*)'ARCHIVO DE RESULTADOS ==> "PROGRAMA CONVEC"'
440   FORMAT(1X,'Fecha DD/MM/AA : ',I2.2,'/',I2.2,'/',I4)
      WRITE(8,99)
444   FORMAT(1X,'MALLA :',i5,' FILAS (r)','   X ',i5,' COLUMNAS(z) ')
      WRITE(8,444)Nr,Nz
      WRITE(8,99)
445   FORMAT(1X,'Deltaz [m]: ',f15.8)
      WRITE(8,445)deltaz
446   FORMAT(1X,'Deltar [m]: ',f15.8)
      WRITE(8,446)deltar
447   FORMAT(1X,'    Lz [m]: ',f15.8)
      WRITE(8,447)Lz
448   FORMAT(1X,'    Lr [m]: ',f15.8)
      WRITE(8,448)Lr
449   FORMAT(1X,'    RD [m]: ',f15.8)
      WRITE(8,449)RD
450   FORMAT(1X,'    Do [m]: ',f15.8)
      WRITE(8,450)Do
451   FORMAT(1X,'    Di [m]: ',f15.8)
      WRITE(8,451)Di
452   FORMAT(1X,'    Ro [m]: ',f15.8)
      WRITE(8,452)Ro
453   FORMAT(1X,'    Ri [m]: ',f15.8)
      WRITE(8,453)Ri
      WRITE(8,99)
      WRITE(8,99)
      WRITE(8,*)'FLUIDO DE ESTUDIO '
      WRITE(8,*)Fluid
      WRITE(8,99)
      WRITE(8,99)
      WRITE(8,*)' PROPIEDADES TERMOFISICAS EN LA SECCION DE ENTRADA '
      WRITE(8,99)
      WRITE(8,99)
      WRITE(8,554)Vi
      WRITE(8,555)ReDi
      WRITE(8,556)Pri
      WRITE(8,557)mui
      WRITE(8,558)nui
      WRITE(8,559)rhoi
      WRITE(8,560)alphai
      WRITE(8,561)Cpi
      WRITE(8,562)ki
      WRITE(8,99)
      WRITE(8,99)
      WRITE(8,563)Ti
      WRITE(8,564)Tso
      WRITE(8,565)Tsi
      WRITE(8,5644)qso
      WRITE(8,5655)qsi
      WRITE(8,99)
      WRITE(8,99)
```

```
      WRITE(8,566)Umed
      WRITE(8,567)Tmed
      WRITE(8,568)ReDmed
      WRITE(8,569)Prmed
      WRITE(8,99)
      WRITE(8,99)
      WRITE(8,5600)letNuD
      WRITE(8,5611)ledCf
      WRITE(8,99)
      WRITE(8,99)

554         FORMAT(1X,'Velocidad de entrada (Vi)   = ',D25.15)
555         FORMAT(1X,'Reynolds (ReD)              = ',D25.15)
556         FORMAT(1X,'Prandlt (Pr)                = ',D25.15)
557         FORMAT(1X,'Viscosidad Dinamica (mu)    = ',D25.15)
558         FORMAT(1X,'Viscosidad Cinematica (nu)  = ',D25.15)
559         FORMAT(1X,'Densidad (rho)              = ',D25.15)
560         FORMAT(1X,'Difusividad Termica (alpha) = ',D25.15)
561         FORMAT(1X,'Calor Especifico (Cp)       = ',D25.15)
562         FORMAT(1X,'Conductividad Termica (k)   = ',D25.15)
563         FORMAT(1X,'Temperatura de entrada (Ti) = ',D25.15)
564         FORMAT(1X,'Ts Cilindro exterior (Tso)  = ',D25.15)
565         FORMAT(1X,'Ts Cilindro interior (Tsi)  = ',D25.15)
566         FORMAT(1X,'VELOCIDAD PROMEDIO          = ',D25.15)
567         FORMAT(1X,'TEMPERATURA PROMEDIO        = ',D25.15)
568         FORMAT(1X,'NUMERO DE ReD PROMEDIO      = ',D25.15)
569         FORMAT(1X,'NUMERO DE Pr  PROMEDIO      = ',D25.15)
5644        FORMAT(1X,'tasa calor Cil. ext (qso)   = ',D25.15)
5655        FORMAT(1X,'tasa calor Cil. int (qsi)   = ',D25.15)
5666        FORMAT(1X,'Error de presión            = ',D25.15)
5677        FORMAT(1X,'Error de Temperatura        = ',D25.15)
5688        FORMAT(1X,'Error de velocidad          = ',D25.15)

5600        FORMAT(1X,'Let numerica                = ',D25.15)
5611        FORMAT(1X,'Led numerica                = ',D25.15)
5622        FORMAT(1X,'Let empirica                = ',D25.15)
5633        FORMAT(1X,'led empirica                = ',D25.15)

      WRITE(8,5666)ErrP
      WRITE(8,5677)ErrT
      WRITE(8,5688)ErrV

      IF(Iter.LT.Max) THEN

      WRITE(8,99)
      WRITE(8,666)Iter
666   FORMAT(1X,'NUMERO DE ITERACIONES (ALGORITMO SIMPLERT) : ',I7)

      ELSE

      WRITE(8,99)
550   FORMAT(1X,'NOTA : NO SE LOGRO LA CONVERGENCIA LUEGO DE',i7, &
            ' ITERACIONES')
      WRITE(8,550)Max

      END IF
```

```
      WRITE(8,99)
904   FORMAT(1X,'NUMERO TOTAL DE ITERACIONES :  ',I10)
      WRITE(8,904)Iter-1
      WRITE(8,99)
901   FORMAT(1X,'Hora de Inicio :       ',I2, ':', I2.2, ':', I2.2, &
           A, 'm')
      WRITE (8,901)HI,MI,SI,MERI
902   FORMAT(1X,'Hora de Culminacion :  ',I2, ':', I2.2, ':', I2.2, &
           A, 'm')
      WRITE (8,902)HF,MF,SF,MERF
      WRITE(8,99)
903   FORMAT(1X,'TIEMPO TOTAL DE ITERACION :  ',I9,' seg')
      WRITE(8,903)TIME
      WRITE(8,99)
999   FORMAT(1000D25.16)
1000  FORMAT(D25.16)
990   FORMAT(/)

1004  format(F14.8)

      WRITE(8,990)
      WRITE(8,*) 'r[m]'
      WRITE(8,990)

      WRITE(8,1004)(r(I),I=1,Mr-2)

      WRITE(8,990)
      WRITE(8,*) 'z[m]'
      WRITE(8,990)
      WRITE(8,1004)(z(I),I=1,Mz-2)

      WRITE(8,990)

      WRITE(8,*) 'DENSIDAD [Kg/m^3]'
      WRITE(8,990)

      DO i=1,Nz
            WRITE(8,999)(rho(2*I,2*J),j=1,Nr)

      END DO

      WRITE(8,990)

      WRITE(8,*) 'VISCOSIDAD CINEMATICA [N.S/m^2]'
      WRITE(8,990)

      DO i=1,Nz

            WRITE(8,999)(mu(2*I,2*J),j=1,Nr)

      END DO

      WRITE(8,990)

      WRITE(8,*) 'DIFUSIVIDAD TERMICA [m^2/s]'
      WRITE(8,990)
```

```
DO i=1,Nz

      WRITE(8,999)(alpha(2*I,2*J),j=1,Nr)

END DO

WRITE(8,990)
WRITE(8,*)'NUMERO DE ReD PROMEDIO'
WRITE(8,990)

DO i=1,Nz

      WRITE(8,999)ReD(i)

END DO

WRITE(8,990)

WRITE(8,*) 'NUMERO DE Pr '
WRITE(8,990)

DO i=1,Nz

      WRITE(8,999)(Pr(2*I,2*J),j=1,Nr)

END DO

WRITE(8,990)
WRITE(8,*)'NUMERO DE Pr PROMEDIO'
WRITE(8,990)

DO i=1,Nz

      WRITE(8,999)Prm(i)

END DO

WRITE(8,990)
WRITE(8,*)'NUMERO DE GRAETZ '
WRITE(8,990)

DO i=1,Nz

      WRITE(8,999)GzD(i)

END DO

WRITE(8,990)

WRITE(8,990)
WRITE(8,*) 'VELOCIDAD Uz [m/s]'
WRITE(8,990)

DO i=1,Nz

      WRITE(8,999)(Uz(2*I+1,2*J),J=1,Nr)
```

```
      END DO

      WRITE(8,990)
      WRITE(8,*) 'VELOCIDAD Ur [m/s]'
      WRITE(8,990)

      DO I=1,Nz

            WRITE(8,999)(Ur(2*I,2*J+1),J=1,Nr)

      END DO

      WRITE(8,990)
      WRITE(8,*) 'PRESION P [Pa]'
      WRITE(8,990)

      DO i=1,Nz

            WRITE(8,999)(P(2*I,2*J),j=1,Nr)

      END DO

      WRITE(8,990)
      WRITE(8,*) 'TEMPERATURA T [K]'
      WRITE(8,990)

      DO i=1,Nz

            WRITE(8,999)(T(2*I,2*J),j=1,Nr)
!           WRITE(10,999)(T(2*I,2*J),j=1,Nr+1)

      END DO

      WRITE(8,990)
      WRITE(8,*)'DIFERENCIA ADIMENSIONAL DE TEMPERATURAS'
      WRITE(8,990)

      DO i=1,Nz

            WRITE(8,999)(TMM(I,J),j=1,Nr)

      END DO

            WRITE(8,990)
            WRITE(8,*)'TEMPERATURA MEDIA [K]'
            WRITE(8,990)

      DO i=1,Nz

            WRITE(8,999)Tm(i)

      END DO

      WRITE(8,990)
      WRITE(8,*)'VELOCIDAD MEDIA [K]'
      WRITE(8,990)
```

```
      DO i=1,Nz

            WRITE(8,999)Um(i)

      END DO

END SUBROUTINE SALRES
```

SUBRUTINA SALRES_GzD: salida de resultados para el TECPLOT . Gráficas de Nusselt por el diámetro contra el inverso del número de Graetz.

```
SUBROUTINE SALRES_GzD(GzD,NuDi,NuDo)

      USE Malla
      Use Variables_Geometricas

      IMPLICIT NONE

      INTEGER :: I,J,X,Y
      REAL(8) :: GzD(0:Mz),NuDi(0:Mz),NuDo(0:Mz),NuDDi,NuDDo,GzDD

111   FORMAT()
222   FORMAT(1X,' Zone  I=',i5,' F=POINT')

      WRITE(11,*)' Title="numero de Graetz" '
      WRITE(11,*)' Variables="1/GzD","NuDi","NuDo"'
      WRITE(11,111)
      WRITE(11,222)Nz
      WRITE(11,111)

      DO I=1,Nz

            GzDD=1/GzD(I)
            NuDDi=NuDi(I)
            NuDDo=NuDo(I)

            WRITE(11,*)NuDDi,NuDDo,GzDD

      END DO

END SUBROUTINE SALRES_GzD
```

SUBRUTINA SALRES_TPV: salida de resultados para el TECPLOT. Gráficas de distribución de los campos de presión, velocidad y temperatura.

```
SUBROUTINE SALRES_TPV(Uz,Ur,P,T,r,z,rho,mu,alpha,Pr)

      USE Malla
      Use Variables_Geometricas
```

```
      IMPLICIT NONE

      INTEGER :: I,J,X,Y
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),P(0:Mz,0:Mr)
      REAL(8) :: T(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr),rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
      REAL(8) :: alpha(0:Mz,0:Mr),Pr(0:Mz,0:Mr),ZZ,RR,Vr,Vz,PP,TT

111   FORMAT()
222   FORMAT(1X,' Zone  I=',i5,' J=',i5,' F=POINT')
333   FORMAT(1X,' GEOMETRY X = 0.0, Y = 0.0, T = LINE, CS = GRID,')
444   FORMAT(1X,' L = SOLID, LT =',F15.8,', C = BLACK, FC = BLACK,')
555   FORMAT(1X,' F = POINT, S = GLOBAL')

      WRITE(10,*)' Title="Campo Vectorial de Velocidad" '
      WRITE(10,*)' Variables="X","Y","z(m)","r(m)","T(K)","P(Pa)", &
      "Ur(m/s)","Uz(m/s)"'
      WRITE(10,111)
      WRITE(10,222)Nz,Nr
      WRITE(10,111)

      X=-1
      Y=-1

      DO J=1,Nr

            Y=Y+1
            RR=r(2*J)

            X=1

            DO I=1,Nz

                  X=X+1
                  ZZ=Z(2*I)
                  TT=T(2*I,2*J)
                  PP=P(2*I,2*J)
                  Vr=Ur(2*I,2*J+1)
                  Vz=Uz(2*I+1,2*J)

                  WRITE(10,*)X,Y,ZZ,RR,TT,PP,Vr,Vz

            END DO

      END DO
```

END SUBROUTINE SALRES_TPV

SUBRUTINA SIMPLERT: desarrollo del algoritmo de solución para el sistema de ecuaciones algebraicas.

```
SUBROUTINE SIMPLERT(Uz,Ur,U0z,U0r,P,P0,T,T0,r,z,b,Apr,Apz,rho,mu, &
alpha,k)
```

```
      USE Malla
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Err


      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr),P(0:Mz,0:Mr)
      REAL(8) :: U0z(0:Mz,0:Mr),U0r(0:Mz,0:Mr),P0(0:Mz,0:Mr)
      REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr),T(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr),b(0:Mz,0:Mr),k(0:Mz,0:Mr)
      REAL(8) :: alpha(0:Mz,0:Mr),mu(0:Mz,0:Mr),rho(0:Mz,0:Mr)
      REAL(8) :: T0(0:Mz,0:Mr)

      KK=0

      WRITE(*,*)'ITERANDO EL ALGORITMO SIMPLERT =>'

      DO Iter=1,Max

            KK=1+KK

            IF (Iter==1) THEN

                  CALL PROPIEDADES_T(rho,mu,alpha,T,k)

            END IF

            CALL M_CVRG(Uz,U0z,Ur,U0r,P,P0,T,T0)
            CALL PSEDUz(Uz,Ur,r,z,P,rho,mu,Apr,Apz)
            CALL PSEDUr(Uz,Ur,r,z,P,rho,mu,Apr,Apz)
            CALL PRES(Uz,Ur,Apr,Apz,P,r,z,b,rho)
            CALL ENER(Uz,Ur,T,r,z,alpha,Apr,Apz,k)
            CALL PROPIEDADES_T(rho,mu,alpha,T,k)
            CALL VELUz(Uz,Ur,P,r,z,rho,mu,Apz,Apr)
            CALL VELUr(Uz,Ur,P,r,z,rho,mu,Apz,Apr)
            CALL PRES(Uz,Ur,Apr,Apz,P,r,z,b,rho)
            CALL CRRV(Uz,Ur,P,Apr,Apz,rho,mu,r)
            CALL ENER(Uz,Ur,T,r,z,alpha,Apr,Apz,k)
            CALL PROPIEDADES_T(rho,mu,alpha,T,k)
            CALL CVRG(Uz,Ur,P,U0z,U0r,P0,T,T0)

            I=KK/10

            IF(I.EQ.1) THEN

                  WRITE(*,56)
                  WRITE(*,111)Iter
111               FORMAT('ITERANDO EL ALGORITMO SIMPLERT =>',5x, &
                  'Iteracion:',i7,/)
                  WRITE(*,99)
                  WRITE(*,222)ErrP
                  WRITE(*,333)ErrT
                  WRITE(*,444)ErrV
```

```
222               FORMAT('Epsilon P: ',D25.15,2x,/)
333               FORMAT('Epsilon T: ',D25.15,2x,/)
444               FORMAT('Epsilon V: ',D25.15,2x,/)
                  KK=0

            END IF

            IF(ErrT<TolT.AND.ErrV<TolV.AND.TOLT<ErrT) EXIT

      END DO

      IF(Iter.LT.Max) THEN

            WRITE(*,99)
            WRITE(*,666)Iter
666         FORMAT(1X,'NUMERO DE ITERACIONES (SIMPLERT) : ',I7)

      END IF

56    FORMAT(/,/)
      WRITE(*,56)

      IF(Iter.LT.Max) THEN

11    FORMAT(5x,'CONVERGENCIA LOGRADA EN')
      WRITE(*,11)
      WRITE(*,*),Iter,' ITERACIONES'

55    FORMAT(/)
      WRITE(*,77)
66    FORMAT(X,'Presione [ENTER] para continuar')
      WRITE(*,66)
      WRITE(*,55)
77    FORMAT(/,/)
      READ(*,*)

            ELSE

                  WRITE(*,*)'NO SE LOGRO LA CONVERGENCIA LUEGO DE'
                  WRITE(*,*),Max,' ITERACIONES'
                  WRITE(*,99)
88                FORMAT(X,'Presione [ENTER] para continuar')
                  WRITE(*,88)
99                FORMAT()
                  READ(*,*)

      END IF

END SUBROUTINE SIMPLERT
```

SUBRUTINA TMEDIA: calcula los valores de temperatures medias.

```
SUBROUTINE TMEDIA(T,Uz,Ur,r,TM,UM,TMM,alpha,rho,mu,rhom,mum, &
alpham,Apr,Apz)
```

```
USE Malla
USE Variables_Geometricas
USE Constantes
USE Variables_VC

IMPLICIT NONE

INTEGER :: I,J
REAL(8) :: T(0:Mz,0:Mr),Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),TM(0:Mz)
REAL(8) :: TMM(0:Mz,0:Mr),alpha(0:Mz,0:Mr),rho(0:Mz,0:Mr)
REAL(8) :: UM(0:Mz),Upz(0:Mz,0:Mr),r(0:Mr),Apz(0:Mz,0:Mr)
REAL(8) :: mum(0:Mz),rhom(0:Mz),alpham(0:Mz),mu(0:Mz,0:Mr)
REAL(8) :: Apr(0:Mz,0:Mr)

DO I=1,Nz

      DO J=1,Nr

            CALL VC_T(I,J,Ur,Uz,r,T,alpha,Apr,Apz)

            Upz(2*I,2*J)=Uzp

            Um(I)=(Uzp/Nr)+Um(I)

      END DO

END DO

DO I=1,Nz

      DO J=1,Nr

            Tm(I)=((Upz(2*I,2*J)*T(2*I,2*J))/(Um(I)*Nr))+Tm(I)

      mum(I)=((Upz(2*I,2*J)*mu(2*I,2*J))/(Um(I)*Nr))+mum(I)

      rhom(I)=((Upz(2*I,2*J)*rho(2*I,2*J))/(Um(I)*Nr))+rhom(I)

      alpham(I)=((Upz(2*I,2*J)*alpha(2*I,2*J))/(Um(I)*Nr))  &
                  +alpham(I)

      END DO

END DO

TM(0)=T(0,2)

DO I=1,Nz+1

      DO J=1,Nr

            IF(TM(I).EQ.Tsi) THEN

                  TMM(I,J)=TMM(I-1,J)

            ELSE
```

```
                        TMM(I,J)=(Tsi-T(2*I,2*J))/(Tsi-TM(I))

                  END IF

            END DO

      END DO

END SUBROUTINE TMEDIA
```

SUBRUTINA TRIDIAG: codificación del algoritmo de Thomas. Para la solucion de sistemas tridiagonales de ecuaciones.

```
SUBROUTINE TRIDIAG(N,A,B,C,G,X)

      USE Malla

      IMPLICIT NONE

      INTEGER :: N,I,J
      REAL(8) :: A(IJ),AA(IJ),BB(IJ),B(IJ)
      REAL(8) :: C(IJ),CC(IJ),G(IJ),X(IJ),T


!**  Resuelve Sistemas Tridiagonal de Ecuaciones  ******************
!**  Algoritmo de Thomas  ******************************************
!*****************************************************************
!* La Forma de la Matriz Tridiagonal es:                         *
!                                                                 *
!     D(1)   U(1)                                = C(1)           *
!     L(2)   D(2)   U(2)                         = C(2)           *
!          L(3)   D(3)   U(3)                    = C(3)           *
!                   .          *                                 *
!                              *                                 *
!                   .          *                                 *
!                        L(n-1)  D(n-1)  U(n-1) = C(n-1)         *
!                                L(n)    D(n)   = C(n)           *
!*****************************************************************

!-----------------------------------------------
!           ELIMINACION HACIA ADELANTE          -
!-----------------------------------------------

      DO I=1,N

            AA(I)=-A(I)
            BB(I)=B(I)
            CC(I)=-C(I)

      END DO

      DO I=2,N
```

```
            T=AA(I)/BB(I-1)
            BB(I)=BB(I)-CC(I-1)*T
            G(I)=G(I)-G(I-1)*T

      END DO


!-----------------------------------------------
!           SUSTITUCION HACIA ATRAS             -
!-----------------------------------------------
      X(N)=G(N)/BB(N)

      DO I=1,N-1

            J=N-I
            X(J)=(G(J)-CC(J)*X(J+1))/BB(J)

      END DO

END  SUBROUTINE TRIDIAG
```

SUBRUTINA VC_P: define todas las variables (presión, velocidad, densidad, radio, área) del volumen de control requeridas para la ecuación de presión.

```
SUBROUTINE VC_P(I,J,Ur,Uz,r,P,Apz,Apr,rho)

      Use Malla
      Use Variables_Geometricas
      Use Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: rho(0:Mz,0:Mr)
      REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),r(0:Mr)
      REAL(8) :: P(0:Mz,0:Mr)
      REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)


            PEE=P(2*I+2,2*J)
            PWW=P(2*I-2,2*J)
            PTT=P(2*I,2*J+2)
            PBB=P(2*I,2*J-2)
            Pp=P(2*I,2*J)

            Pe=(PEE+Pp)*0.5
            Pw=(Pp+PWW)*0.5
            Pt=(PTT+Pp)*0.5
            Pb=(Pp+PBB)*0.5

            rhoEE=rho(2*I+2,2*J)
            rhoWW=rho(2*I-2,2*J)
            rhoTT=rho(2*I,2*J+2)
```

```
rhoBB=rho(2*I,2*J-2)
rhop=rho(2*I,2*J)

rhoe=(rhoEE+rhop)*0.5
rhow=(rhop+rhoWW)*0.5
rhot=(rhoTT+rhop)*0.5
rhob=(rhop+rhoBB)*0.5

Uze=Uz(2*I+1,2*J)
Uzw=Uz(2*I-1,2*J)
UzTT=(Uz(2*I+1,2*J+2)+Uz(2*I-1,2*J+2))*0.5
UzBB=(Uz(2*I+1,2*J-2)+Uz(2*I-1,2*J-2))*0.5
Uzp=(Uze+Uzw)*0.5
Uzt=(UzTT+Uzp)*0.5
Uzb=(Uzp+UzBB)*0.5

Urt=Ur(2*I,2*J+1)
Urb=Ur(2*I,2*J-1)
UrEE=(Ur(2*I+2,2*J+1)+Ur(2*I+2,2*J-1))*0.5
UrWW=(Ur(2*I-2,2*J+1)+Ur(2*I-2,2*J-1))*0.5
Urp=(Urt+Urb)*0.5
Ure=(UrEE+Urp)*0.5
Urw=(Urp+UrWW)*0.5

re=r(2*J)
rw=r(2*J)
rWW=r(2*J)
rEE=r(2*J)
rp=r(2*J)
rt=r(2*J+1)
rTT=r(2*J+2)
rb=r(2*J-1)
rBB=r(2*J-2)

Aze=Apz(2*I+1,2*J)
Azw=Apz(2*I-1,2*J)
Art=Apr(2*I,2*J+1)
Arb=Apr(2*I,2*J-1)
```

END SUBROUTINE VC_P

SUBRUTINA VC_T: define todas las variables (presión, velocidad, propiedades termofísicas, temperatura, radio, área) del volumen de control requeridas para la ecuación de temperatura.

SUBROUTINE VC_T(I,J,Ur,Uz,r,T,alpha,Apr,Apz)

```
Use Malla
Use Variables_Geometricas
Use Variables_VC

IMPLICIT NONE
```

```
INTEGER :: I,J
REAL(8) :: alpha(0:Mz,0:Mr)
REAL(8) :: alphad
REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),r(0:Mr)
REAL(8) :: T(0:Mz,0:Mr)
REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)

      TEE=T(2*I+2,2*J)
      TWW=T(2*I-2,2*J)
      TTT=T(2*I,2*J+2)
      TBB=T(2*I,2*J-2)
      Tp=T(2*I,2*J)

      Te=(TEE+Tp)*0.5
      Tw=(Tp+TWW)*0.5
      Tt=(TTT+Tp)*0.5
      Tb=(Tp+TBB)*0.5

      alphaEE=alpha(2*I+2,2*J)
      alphaWW=alpha(2*I-2,2*J)
      alphaTT=alpha(2*I,2*J+2)
      alphaBB=alpha(2*I,2*J-2)
      alphap=alpha(2*I,2*J)

      alphae=(alphaEE+alphap)*0.5
      alphaw=(alphap+alphaWW)*0.5
      alphat=(alphaTT+alphap)*0.5
      alphab=(alphap+alphaBB)*0.5

      Uze=Uz(2*I+1,2*J)
      Uzw=Uz(2*I-1,2*J)
      UzTT=(Uz(2*I+1,2*J+2)+Uz(2*I-1,2*J+2))*0.5
      UzBB=(Uz(2*I+1,2*J-2)+Uz(2*I-1,2*J-2))*0.5
      Uzp=(Uze+Uzw)*0.5
      Uzt=(UzTT+Uzp)*0.5
      Uzb=(Uzp+UzBB)*0.5

      Urt=Ur(2*I,2*J+1)
      Urb=Ur(2*I,2*J-1)
      UrEE=(Ur(2*I+2,2*J+1)+Ur(2*I+2,2*J-1))*0.5
      UrWW=(Ur(2*I-2,2*J+1)+Ur(2*I-2,2*J-1))*0.5
      Urp=(Urt+Urb)*0.5
      Ure=(UrEE+Urp)*0.5
      Urw=(Urp+UrWW)*0.5

      Aze=Apz(2*I+1,2*J)
      Azw=Apz(2*I-1,2*J)
      Art=Apr(2*I,2*J+1)
      Arb=Apr(2*I,2*J-1)

      re=r(2*J)
      rw=r(2*J)
      rWW=r(2*J)
      rEE=r(2*J)
      rp=r(2*J)
      rt=r(2*J+1)
```

```
            rTT=r(2*J+2)
            rb=r(2*J-1)
            rBB=r(2*J-2)
```

END SUBROUTINE VC_T

SUBRUTINA VC_Ur: define todas las variables (presión, velocidad, propiedades termofísicas, radio, área) del volumen de control requeridas para la ecuación de la componente radial de la velocidad.

```
SUBROUTINE VC_Ur(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

      Use Malla
      Use Variables_VC
      Use Variables_Geometricas

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
      REAL(8) :: rhod,mud
      REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),r(0:Mr),P(0:Mz,0:Mr)
      REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)

            rhot=rho(2*i,2*J+2)
            rhob=rho(2*I,2*J)
            rhoWW=(rho(2*I-2,2*J)+rho(2*I-2,2*J+2))*0.5
            rhop=(rhot+rhob)*0.5
            rhoEE=(rho(2*I+2,2*J)+rho(2*I+2,2*J+2))*0.5
            rhoe=(rhoEE+rhop)*0.5
            rhow=(rhoWW+rhop)*0.5

            Pt=P(2*i,2*J+2)
            Pb=P(2*I,2*J)
            PWW=(P(2*I-2,2*J)+P(2*I-2,2*J+2))*0.5
            Pp=(Pt+Pb)*0.5
            PEE=(P(2*I+2,2*J)+P(2*I+2,2*J+2))*0.5
            Pe=(PEE+Pp)*0.5
            Pw=(PWW+Pp)*0.5

            mut=mu(2*i,2*J+2)
            mub=mu(2*I,2*J)
            muWW=(mu(2*I-2,2*J)+mu(2*I-2,2*J+2))*0.5
            mup=(mut+mub)*0.5
            muEE=(mu(2*I+2,2*J)+mu(2*I+2,2*J+2))*0.5
            mue=(muEE+mup)*0.5
            muw=(muWW+mup)*0.5

            Uze=(Uz(2*I+1,2*J+2)+Uz(2*I+1,2*J))*0.5
            Uzw=(Uz(2*I-1,2*J+2)+Uz(2*I-1,2*J))*0.5
            Uzt=(Uz(2*I+1,2*J+2)+Uz(2*I-1,2*J+2))*0.5
            Uzb=(Uz(2*I+1,2*J)+Uz(2*I-1,2*J))*0.5
```

```
      Uzp=(Uze+Uzw+Uzt+Uzb)*0.25

      UrEE=Ur(2*I+2,2*J+1)
      UrWW=Ur(2*I-2,2*J+1)
      UrTT=Ur(2*I,2*J+3)
      UrBB=Ur(2*I,2*J-1)
      Urp=Ur(2*I,2*J+1)
      Ure=(UrEE+Urp)*0.5
      Urw=(UrWW+Urp)*0.5
      Urt=(UrTT+Urp)*0.5
      Urb=(UrBB+Urp)*0.5

      Aze=(Apz(2*I+1,2*J+2)+Apz(2*I+1,2*J))*0.5
      Azw=(Apz(2*I-1,2*J+2)+Apz(2*I-1,2*J))*0.5
      Art=(Apr(2*I,2*J+3)+Apr(2*I,2*J+1))*0.5
      Arb=(Apr(2*I,2*J+1)+Apr(2*I,2*J-1))*0.5
      Arp=Apr(2*I,2*J+1)

      re=r(2*J+1)
      rw=r(2*J+1)
      rWW=r(2*J+1)
      rEE=r(2*J+1)
      rp=r(2*J+1)
      rt=r(2*J+2)
      rTT=r(2*J+3)
      rb=r(2*J)
      rBB=r(2*J-1)

END SUBROUTINE VC_Ur
```

SUBRUTINA VC_Uz: define todas las variables (presión, velocidad, propiedades termofísicas, radio, área) del volumen de control requeridas para la ecuación de la componente axial de la velocidad.

```
SUBROUTINE VC_Uz(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

      Use Malla
      Use Variables_VC
      Use Variables_Geometricas

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
      REAL(8) :: rhod,mud
      REAL(8) :: Ur(0:Mz,0:Mr),Uz(0:Mz,0:Mr),r(0:Mr),P(0:Mz,0:Mr)
      REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)

            rhoe=rho(2*I+2,2*J)
            rhow=rho(2*I,2*J)
            rhoTT=(rho(2*I+2,2*J+2)+rho(2*I,2*J+2))*0.5
```

```
rhop=(rhoe+rhow)*0.5
rhoBB=(rho(2*I+2,2*J-2)+rho(2*I,2*J-2))*0.5
rhot=(rhoTT+rhop)*0.5
rhob=(rhoBB+rhop)*0.5

Pe=P(2*I+2,2*J)
Pw=P(2*I,2*J)
PTT=(P(2*I+2,2*J+2)+P(2*I,2*J+2))*0.5
Pp=(Pe+Pw)*0.5
PBB=(P(2*I+2,2*J-2)+P(2*I,2*J-2))*0.5
Pt=(PTT+Pp)*0.5
Pb=(PBB+Pp)*0.5

mue=mu(2*I+2,2*J)
muw=mu(2*I,2*J)
muTT=(mu(2*I+2,2*J+2)+mu(2*I,2*J+2))*0.5
mup=(mue+muw)*0.5
muBB=(mu(2*I+2,2*J-2)+mu(2*I,2*J-2))*0.5
mut=(muTT+mup)*0.5
mub=(muBB+mup)*0.5

Ure=(Ur(2*I+2,2*J+1)+Ur(2*I+2,2*J-1))*0.5
Urw=(Ur(2*I,2*J+1)+Ur(2*I,2*J-1))*0.5
Urt=(Ur(2*I,2*J+1)+Ur(2*I+2,2*J+1))*0.5
Urb=(Ur(2*I,2*J-1)+Ur(2*I+2,2*J-1))*0.5
Urp=(Ure+Urw+Urt+Urb)*0.25

UzEE=Uz(2*I+3,2*J)
UzWW=Uz(2*I-1,2*J)
UzTT=Uz(2*I+1,2*J+2)
UzBB=Uz(2*I+1,2*J-2)
Uzp=Uz(2*I+1,2*J)
Uze=(UzEE+Uzp)*0.5
Uzw=(UzWW+Uzp)*0.5
Uzt=(UzTT+Uzp)*0.5
Uzb=(UzBB+Uzp)*0.5

Aze=(Apz(2*I+1,2*J)+Apz(2*I+3,2*J))*0.5
Azw=(Apz(2*I+1,2*J)+Apz(2*I-1,2*J))*0.5
Art=(Apr(2*I,2*J+1)+Apr(2*I+2,2*J+1))*0.5
Arb=(Apr(2*I,2*J-1)+Apr(2*I+2,2*J-1))*0.5
Azp=Apz(2*I+1,2*J)

re=r(2*J)
rw=r(2*J)
rWW=r(2*J)
rEE=r(2*J)
rp=r(2*J)
rt=r(2*J+1)
rTT=r(2*J+2)
rb=r(2*J-1)
rBB=r(2*J-2)
```

END SUBROUTINE VC_Uz

SUBRUTINA VECTORES_rz: calcula los valores de los vectores de radios (*r*) y longitudes (*z*).

```
SUBROUTINE VECTORES_rz(r,z)

      USE Malla
      USE Variables_Geometricas

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: z(0:Mz),r(0:Mr)

      r(0)=ri-deltar
      z(0)=-deltaz

      DO I=1,Mz

            z(I)=z(I-1)+deltaz

      END DO

      DO J=1,Mr

            r(J)=r(J-1)+deltar

      END DO

END SUBROUTINE VECTORES_rz
```

SUBRUTINA VELUr: resulve la ecuación de momentun en dirección *r*, obteniendo los valores del campo de velocidad radial, a partir de los campos de presión y temperatura.

```
SUBROUTINE VELUr(Uz,Ur,P,r,z,rho,mu,Apz,Apr)

      USE Malla
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: FAEE,FAWW,FATT,FABB
      REAL(8) :: AWW(IJ),AEE(IJ),ATT(IJ),ABB(IJ),Ap(IJ)
      REAL(8) :: b(IJ),RA(IJ),X(IJ)
      REAL(8) :: P(0:Mz,0:Mr)
```

```
      REAL(8) :: Apr(0:Mz,0:Mr),Apz(0:Mz,0:Mr)
      REAL(8) :: rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr)

!-----------------------------------------------
!             BARRIDO OESTE-ESTE -
!-----------------------------------------------
      DO J=1,Nr-1

            DO I=1,Nz

                  CALL VC_Ur(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS

                  Fe=rhoe*Uze*Areae*(1/rp)
                  Fw=rhow*Uzw*Areaw*(1/rp)
                  Ft=rhot*Urt*Areat*(1/rt)
                  Fb=rhob*Urb*Areab*(1/rb)

                  ! COEFICIENTES DIFUSIVOS

                  De=mue*Areae/deltaz*(1/rp)
                  Dw=muw*Areaw/deltaz*(1/rp)
                  Dt=mut*Areat/deltar*(1/rt)
                  Db=mub*Areab/deltar*(1/rb)

                  ! ESQUEMA DE DISCRETIZACION

                  CALL ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

                  AEE(I)=FAEE
                  AWW(I)=FAWW
                  ATT(I)=FATT
                  ABB(I)=FABB

                  Ap(I)=AEE(I)+AWW(I)+ATT(I)+ABB(I)

                  Apr(2*I,2*J+1)=Ap(I)

                  b(I)=(Pb-Pt)*deltaz

                  RA(I)=b(I)+ATT(I)*UrTT+ABB(I)*UrBB

            END DO

            RA(Nz)=RA(Nz)+AEE(Nz)*Ur(2*Nz,2*J+1)
            RA(1) =RA(1 )+AWW(1 )*Ur(0,2*J+1)

            CALL TRIDIAG(Nz,AWW,Ap,AEE,RA,X)

            DO I=1,Nz

                  Ur(2*I,2*J+1)=Ur(2*I,2*J+1) &
```

```
                  +RelaxV*(X(I)-Ur(2*I,2*J+1))

            END DO

      END DO


!-----------------------------------------------
!            BARRIDO SUR-NORTE -
!-----------------------------------------------

      DO I=1,Nz

            DO J=1,Nr-1

                  CALL VC_Ur(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS

                  Fe=rhoe*Uze*Areae*(1/rp)
                  Fw=rhow*Uzw*Areaw*(1/rp)
                  Ft=rhot*Urt*Areat*(1/rt)
                  Fb=rhob*Urb*Areab*(1/rb)

                  ! COEFICIENTES DIFUSIVOS

                  De=mue*Areae/deltaz*(1/rp)
                  Dw=muw*Areaw/deltaz*(1/rp)
                  Dt=mut*Areat/deltar*(1/rt)
                  Db=mub*Areab/deltar*(1/rb)

                  ! ESQUEMA DE DISCRETIZACION

                  CALL ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

                  AEE(I)=FAEE
                  AWW(I)=FAWW
                  ATT(I)=FATT
                  ABB(I)=FABB
                  Ap(I)=AEE(I)+AWW(I)+ATT(I)+ABB(I)

                  Apr(2*I,2*J+1)=Ap(J)

                  b(J)=(Pb-Pt)*deltaz

                  RA(J)=b(J)+AEE(J)*UrEE+AWW(J)*UrWW

            END DO

            CALL TRIDIAG(Nr-1,ABB,Ap,ATT,RA,X)

            DO J=1,Nr-1

                  Ur(2*I,2*J+1)=Ur(2*I,2*J+1) &
                  +RelaxV*(X(J)-Ur(2*I,2*J+1))
```

```
            END DO

      END DO

END SUBROUTINE VELUr
```

SUBRUTINA VELUz: resulve la ecuación de momentun en dirección *z*, obteniendo los valores del campo de velocidad axial, a partir de los campos de presión y temperatura.

```
SUBROUTINE VELUz(Uz,Ur,P,r,z,rho,mu,Apz,Apr)

      USE Malla
      USE Contadores
      USE Constantes
      USE Variables_Geometricas
      USE Variables_VC

      IMPLICIT NONE

      INTEGER :: I,J
      REAL(8) :: FAEE,FAWW,FATT,FABB
      REAL(8) :: AWW(IJ),AEE(IJ),ATT(IJ),ABB(IJ),Ap(IJ)
      REAL(8) :: b(IJ),RA(IJ),X(IJ)
      REAL(8) :: P(0:Mz,0:Mr)
      REAL(8) :: Apz(0:Mz,0:Mr),Apr(0:Mz,0:Mr)
      REAL(8) :: rho(0:Mz,0:Mr),mu(0:Mz,0:Mr)
      REAL(8) :: Uz(0:Mz,0:Mr),Ur(0:Mz,0:Mr)
      REAL(8) :: z(0:Mz),r(0:Mr)

!-----------------------------------------------
!              BARRIDO OESTE-ESTE -
!-----------------------------------------------
      DO J=1,Nr

            DO I=1,Nz

                  CALL VC_Uz(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                  CALL AREAS_VC(J)

                  ! COEFICIENTES ADVECTIVOS

                  Fe=rhoe*Uze*Areae
                  Fw=rhow*Uzw*Areaw
                  Ft=rhot*Urt*Areat
                  Fb=rhob*Urb*Areab

                  ! COEFICIENTES DIFUSIVOS

                  De=mue*Areae/deltaz
                  Dw=muw*Areaw/deltaz
                  Dt=mut*Areat/deltar
```

```
                Db=mub*Areab/deltar

                ! ESQUEMA DE DISCRETIZACION

                CALL ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

                AEE(I)=FAEE
                AWW(I)=FAWW
                ATT(I)=FATT
                ABB(I)=FABB

                Ap(I)=AEE(I)+AWW(I)+ATT(I)+ABB(I)

                Apz(2*I+1,2*J)=Ap(I)

                b(I)=(Pw-Pe)*deltar

                RA(I)=b(I)+ATT(I)*UzTT+ABB(I)*UzBB

          END DO

          RA(1)=RA(1)+AWW(1)*Uz(1,2*J)
          RA(Nz-1)=RA(Nz-1)+AEE(Nz-1)*Uz(2*Nz+1,2*J)

          CALL TRIDIAG(Nz-1,AWW,Ap,AEE,RA,X)

          DO I=1,Nz-1

                Uz(2*I+1,2*J)=Uz(2*I+1,2*J) &
                +RelaxV*(X(I)-Uz(2*I+1,2*J))

          END DO

    END DO

    CALL C_CONTORNO_Uz(Uz)

!-----------------------------------------------
!           BARRIDO SUR-NORTE -
!-----------------------------------------------

    DO I=1,Nz

          DO J=1,Nr

                CALL VC_Uz(I,J,Ur,Uz,r,P,rho,mu,Apz,Apr)

                CALL AREAS_VC(J)

                ! COEFICIENTES ADVECTIVOS

                Fe=rhoe*Uze*Areae
                Fw=rhow*Uzw*Areaw
                Ft=rhot*Urt*Areat
                Fb=rhob*Urb*Areab

                ! COEFICIENTES DIFUSIVOS
```

```
                De=mue*Areae/deltaz
                Dw=muw*Areaw/deltaz
                Dt=mut*Areat/deltar
                Db=mub*Areab/deltar

                ! ESQUEMA DE DISCRETIZACION

                CALL ESQUEMA_DISCRETIZACION(FAEE,FAWW,FATT,FABB)

                AEE(J)=FAEE
                AWW(J)=FAWW
                ATT(J)=FATT
                ABB(J)=FABB
                Ap(J)=AEE(J)+AWW(J)+ATT(J)+ABB(J)

                Apz(2*I+1,2*J)=Ap(J)

                b(J)=(Pw-Pe)*deltar

                RA(J)=b(J)+AEE(J)*UzEE+AWW(J)*UzWW

          END DO

          RA(1)=RA(1)-ABB(1)*Uz(2*I+1,2)
          RA(Nr)=RA(Nr)-ATT(Nr)*Uz(2*I+1,2*Nr)

          CALL TRIDIAG(Nr,ABB,Ap,ATT,RA,X)

          DO J=1,Nr

                Uz(2*I+1,2*J)=Uz(2*I+1,2*J)&
                +RelaxV*(X(J)-Uz(2*I+1,2*J))

          END DO

    END DO

    CALL C_CONTORNO_Uz(Uz)

END SUBROUTINE VELUz
```

ANEXO 2
PROPIEDADES TERMOFÍSICAS

- **Propiedades termofísicas de los aceites:**

Con relación a su comportamiento reológico, los hidrocarburos de interés en este trabajo pueden considerarse fluidos newtonianos que se caracterizan porque su viscosidad cinemática es alta y varía significativamente con la temperatura. Desde el punto de vista de su comportamiento térmico, los hidrocarburos en consideración se caracterizan porque su número de Prandtl es alto y varía significativamente con la temperatura.

A continuación se presenta como fueron obtenidas las funciones que describen las variaciones de las propiedades con la temperatura.

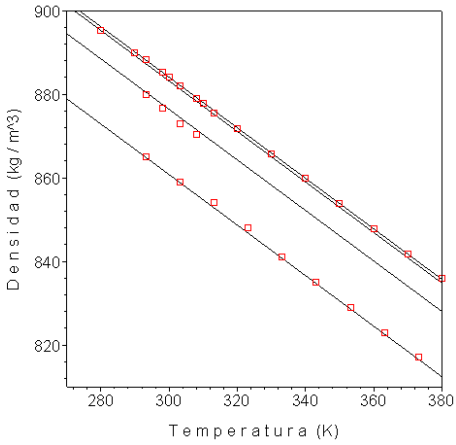

Fig. A2-1: Variación de la densidad (ρ) de los diferentes aceites en función de la temperatura

En la figura A2-1 se pueden apreciar de arriba a abajo y de derecha a izquierda: ρ_{P320}, ρ_{O288}, ρ_{P150} y ρ_{P22} (rectas negras). Los puntos (rojos) son valores tabulados correspondientes a ρ de P320, O288, P150 y P22.

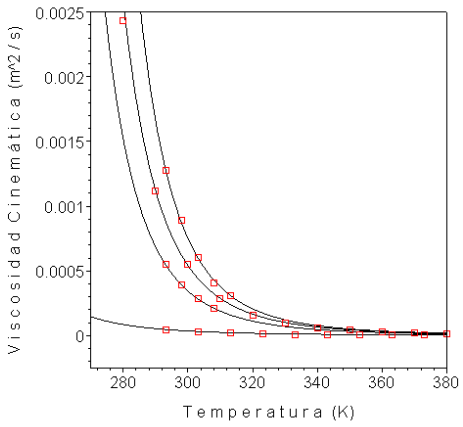

Fig. A2-2: Variación de la viscosidad cinemática (ν) de los diferentes aceites en función de la temperatura

En la figura A2-2 se pueden apreciar de arriba a abajo y de derecha a izquierda: ν_{P320}, ν_{O288}, ν_{P150} y ν_{P22} (curvas negras). Los puntos (rojos) son valores tabulados correspondientes a ν de P320, O288, P150 y P22.

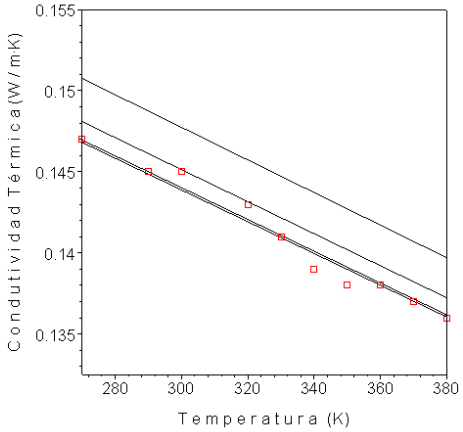

Fig. A2-3: Variación de la conductividad térmica (k) de los diferentes aceites en función de la temperatura

En la figura A2-3 se pueden apreciar de arriba a abajo y de derecha a izquierda: k_{P22}, k_{P150}, k_{O288} y k_{P320} (rectas negras). Los puntos (rojos) son valores tabulados correspondientes a k de O288 con los que se mejoró el ajuste de todas las rectas. La dispersión de los puntos se debe a la poca resolución en los valores tabulados, sin embargo, no es tan grande como aparenta.

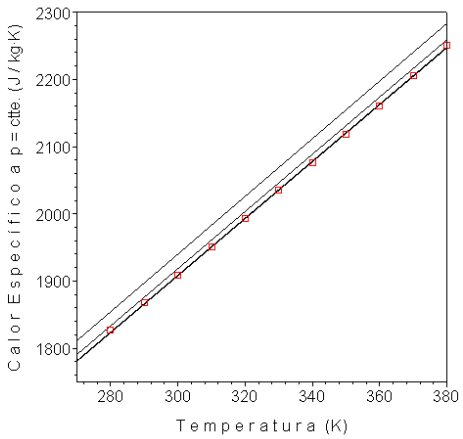

Fig. A2-4: Variación del calor específico (C_p) de los diferentes aceites en función de la temperatura

En la figura A2-4 se pueden apreciar de arriba a abajo y de izquierda a derecha: C_{p_P22}, C_{p_P150}, C_{p_O288} y $Cp_{\_P320}$ (rectas negras, las dos últimas casi superpuestas). Los puntos (rojos) son valores tabulados correspondientes a C_p de O288 con los que se mejoró el ajuste de todas las rectas. Note que es la única propiedad termofísica que aumenta con el aumento de la temperatura.

Partiendo de esta data, se obtuvieron las ecuaciones presentadas en la Tabla 3-1 (ecuaciones para las propiedades dependientes de la temperatura), cuyos coeficientes se muestran a continuación para los cuatro aceites empleados en este trabajo, a saber: P22, P150, O288 y P320.

Tabla A2-1: Coeficientes de las ecuaciones para las propiedades termofísicas de los aceites

	P22	**P150**	**O288**	**P320**
A_ρ	1042.2	1057.8	1064.6	1065.5
B_ρ	-0.6048	-0.6048	-0.6048	-0.6048
A_ν	7.7921	8.5392	8.4089	8.7372
B_ν	-3.0665	-3.2840	-3.2183	-3.3419
A_k	0.1779	0.1748	0.1734	0.1732
B_k	$-1.0058.10^{-4}$	$-0.9880.10^{-4}$	$-0.9805.10^{-4}$	$-0.9795.10^{-4}$
A_{Cp}	653.64	642.53	637.68	637.04
B_{Cp}	4.2884	4.2532	4.2378	4.2357

- **Propiedades termofísicas del agua:**

Para el fluido empleado al realizar la validación, en este caso, el agua, también se consideró el caso de que sus propiedades termofísicas varían en función de la temperatura. Para éste fluido, las ecuaciones de dichas propiedades cambian con respecto a las de los aceites, por ejemplo, las ecuaciones de densidad y de calor específico, vienen dadas por polinomios de grado 5 y 6 respectivamente, y las ecuaciones de viscosidad dinámica y de conductividad térmica vienen dadas por una función exponencial. A continuación se exponen dichas ecuaciones y sus respectivos coeficientes.

Tabla A2-2: Ecuaciones para las propiedades termofísicas del agua

Propiedad	Función Aproximada
Densidad (ρ) [kg/m^3]	$\rho = A_\rho T^5 + B_\rho T^4 + C_\rho T^3 + D_\rho T^2 + E_\rho T + F_\rho$
Viscosidad Dinámica (μ) [Pa.s]	$\mu = \frac{A_\mu}{e^{\frac{B_\mu}{T}}}$
Conductividad Térmica (*k*) [W/m.K]	$k = T^{A_k} e^{(B_k - C_k T)}$
Calor Específico (*Cp*) [J/kg.K]	$C_p = A_{Cp} T^5 + B_{Cp} T^4 + C_{Cp} T^3 + D_{Cp} T^2 + E_{Cp} T + F_{Cp} T + G_{Cp}$
Viscosidad Cinemática (ν) [m^2/s]	$\nu = \frac{\mu}{\rho}$
Difusividad Térmica (α) [m^2/s]	$\alpha = \frac{k}{\rho\, Cp}$

Tabla A2-3: Coeficientes de las ecuaciones para las propiedades termofísicas de los aceites

AGUA	
A_ρ	6.5900 10-9
B_ρ	-1.1300 10-6
C_ρ	1.0100 10-4
D_ρ	-9.1100 10-3
E_ρ	6.8000 10-3
F_ρ	999.84
A_μ	5.3933 10-5
B_μ	856.59
A_k	2.7651
B_k	-14.215
C_k	6.8165 10-3
A_{Cp}	$1.0826\ 10^{-10}$
B_{Cp}	$-2.7266\ 10^{-7}$
C_{Cp}	$2.8301\ 10^{-4}$
D_{Cp}	-0.1548
E_{Cp}	47.049
F_{Cp}	-7531.3
G_{Cp}	500220

ANEXO 3
ALGORITMO DE THOMAS (TDMA)

MÉTODOS ITERATIVOS. MÉTODO LÍNEA A LÍNEA. TDMA

Los métodos iterativos son aquellos que requieren una estimación inicial para dar continuación al proceso de iteración. Son clasificados, en general, como punto a punto, línea a línea o plano a plano. El método línea a línea es un método directo cuando el problema es unidimensional.

El método más conocido de ésta naturaleza (línea a línea) es el algoritmo de Thomas o TDMA (TriDiagonal Matriz Algorithm). Como se deduce por su nombre, los métodos línea a línea resuelven directamente una línea, es decir, un problema unidimensional. Para problemas bi y tridimensionales ellos son iterativos, se procesa con un barrido línea a línea y columna a columna. Para ejemplificar, vamos a considerar un problema bidimensional sin considerar los términos en la dirección z. Considere la Fig. A3-1, donde se muestra una línea en la cual será aplicado el método TDMA. La ecuación a resolver está dad por:

$$A_P T_P = A_e T_E + A_w T_W + A_n T_N + A_s T_S + B_P \qquad \text{(A3.1)}$$

Escribiendo la ec. (A3.1) en una forma más conveniente para procedimientos recursivos, tenemos,

$$A_m T_m + B_m T_{m+1} + C_m T_{m-1} = D_m \qquad \text{(A3.2)}$$

Interesa determinar una relación recursiva de la forma

$$T_m = P_m T_{m+1} + Q_m \qquad \text{(A3.3)}$$

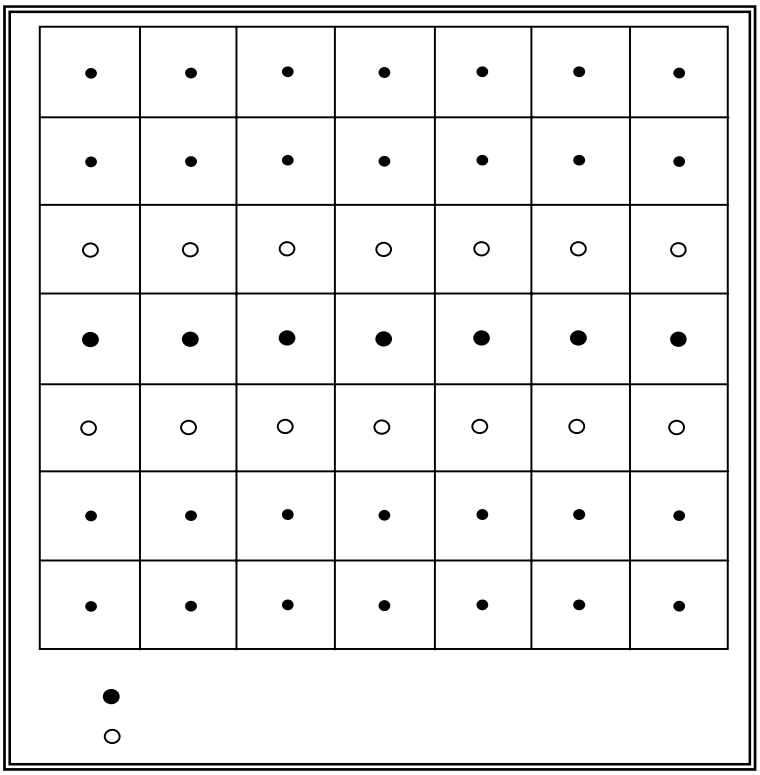

Fig. A3-1: Línea donde es aplicado el método TDMA en un problema 2D

que permita, con el uso de las condiciones de contorno, barrer línea a línea en un sentido, determinando los coeficientes *P* y *Q*, y regresar, determinando los valores de la variable, que para este caso la llamamos *T*. Bajando un índice de la ec. (A3.3), encontramos

$$T_{m-1} = P_{m-1} T_m + Q_{m-1} \tag{A3.4}$$

Sustituyendo la ec. (A3.4) en la ecuación (A3.2) y comparando el resultado con la ec. (A3.3), encontramos las siguientes expresiones para los coeficientes *P* y *Q*:

$$P_m = \frac{-B_m}{A_m + C_m P_{m-1}} \tag{A3.5}$$

$$Q_m = \frac{D_m - C_m Q_{m-1}}{A_m + C_m P_{m-1}} \tag{A3.6}$$

Las ecs. (A3.5) y (A3.6) son relaciones recursivas que permiten, después de conocidos P_1 y Q_1, determinar todos los valores de P y Q. Para nuestro problema, dado por la ec. (A3.1), cuando el desarrollo va de acuerdo con lo mostrado en la Fig.A3-1, tenemos

$$A_m = A_P \; ; \; B_m = -A_e \; ; \; C_m = -A_w \tag{A3.7}$$

y

$$D_m = A_n T_N + A_s T_S + B_P \tag{A3.8}$$

Cuando el barrido es por columnas tendremos las siguientes expresiones para los coeficientes

$$A_m = A_P \; ; \; B_m = -A_n \; ; \; C_m = -A_s \tag{A3.9}$$

y

$$D_m = A_w T_W + A_e T_E + B_P \tag{A3.10}$$

La determinación de P_1 y Q_1 es fácil de inferir, revisando las ecs. (A3.5) y (A3.6). Imaginando que los índices crecen según lo mostrado en la Fig.A3-1, la ecuación la ecuación aproximada para el volumen de frontera (volumen 1) no podrá depender de los valores de la variable de la izquierda. Luego, C_1 deberá ser cero, resultando en

$$P_1 = \frac{-B_1}{A_1} \; ; \; Q_1 = \frac{D_1}{A_1} \tag{A3.11}$$

Para el otro volumen de frontera (volumen N), sabemos que la ecuación aproximada no podrá depender de la variable de la derecha. Luego, B_N deberá ser cero, por la ec. (A3.5), lo que resulta, por la ec. (A3.3), en

$$T_N = Q_N \tag{A3.12}$$

El TDMA es un método utilizado intensamente en el cálculo numérico, dadas su facilidad de implementación y sus buenas características de convergencia. El algoritmo para aplicar el método TDMA puede ser resumido por:

- Estimar el campo de las variables con el que se va ha iniciar el cálculo (iterado inicial)
- Calcular *P1* y *Q1* a través de la ec. (A3.11)
- Calcular todos los *Pm* y *Qm* con m de *2* hasta *N* usando las ecs. (A3.5) y (A3.6)
- Hacer $T_N = Q_N$
- Calcular las variables para los puntos *N-1* hasta *1* usando la ec. (A3.4)
- Revisar la convergencia. Si no ha sido satisfecho el criterio, repetir o alternar la dirección