6
Referências bibliográficas

ALSOY, S.; DUDA, J. L. **Modeling of multicomponent drying of polymeric systems**. AIChE J., 45, 896, (1999)

ALSOY, S. **Predicting drying in multiple zone ovens.** Ind. Eng. Chem. Res., 40, 2995, (2001)

AUST, R.; DURST, F.; RASZILLIER, H**. Modeling of multiple-zone air impingiment dryer.** Chem. Eng. and Process, 36, 469, (1997)

BEARMAN, R. J. **On the molecular basis of some current theories of diffusion**. J. Phys. Chem., 65, 1961, (1961)

BEJAN, A. **Convection heat transfer**. John Wiley & Sons, (1984)

CAIRNCROSS, R. A**. Solidification phenomena during drying of sol-to-gel coatings**. PhD thesis, University of Minnesota, (1994)

CARUTHERS, J. M. et al. **Handbook of diffusion and thermal properties of polymers and polymer solutions.** AIChE

CARVALHO, M. S. **Elementos finitos**. Notas de aula, PUC-Rio, (2002)

CHILTON, T. H.; COLBURN, A. P. **Mass transfer coefficients.** Industrial Engineering Chemistry, 26, 1183, (1934)

COHEN, E. D.; GUTOFF, E. B. **Modern coating and drying technology.** VCH publishers, New York, (1992)

CRANK, J.; PARK, G. S. **Diffusion in polymers.** Academic press Inc., (1968)

CUSSLER, E. J. **Diffusion and mass transfer in fluid systems.** Cambridge University Press, New York, (1984)

DOUMENC, F.; GUERRIER, B. **Estimating polymer-solvent diffusion coefficient by optimization procedure**. AIChE J., 47-5, 984, (2001)

DUDA, J. L.; NI, Y. C.; VRENTAS, J. S. **An equation relating self-diffusion and mutual diffusion coeficients in polymer-solvent systems**. Macromolecules, 12, 459, (1979) DUDA, J. L.; VRENTAS, J. S. **Molecular diffusion in polymer solutions.** AIChE J., 25-1, 1, (1979)

DUDA, J. L. et al**. Prediction of diffusion coeficients for polymer-solvent systems.** AIChE J., 28, 285, (1982)

FAVRE, E. et al. **Aplication of Flory-Huggins theory to ternary polymer-solvent equilibria**: a case study. Eur. Polym. J., 32, 303, (1996)

FERZIGER, J. H. **Numerical methods for engineering application.** John Wiley & Sons, (1998)

FLORY, P. J. **Principles of polymer chemistry**. Cornell Univ. Press., Ithaca, NY (1953)

GUERRIER, B. et al. **Drying kinetics of polymer films**. AIChE J., 44-4, 791, (1998)

HUGGINS, M. L. **Theory of solution of high polymers.** J. Am. Chem. Soc., 64, 1712, (1942)

LOGAN, E. **Thermodynamics – processes and applications.** Marcel Dekker Inc., (1999)

MARK; OVERBERGER; MENGES; BIKALES**. Encyclopedia of polymer science and engineering**. John Wiley & Sons, (1995)

MARTIN, H. **Heat and mass transfer rates between impinging gas jets and solid surfaces.** Adv. Heat Transfer, 13, 1, (1977)

NAUMAN, E. B.; SAVOCA, J. **An engineering approach to an unsolved problem in multicomponent diffusion**. AIChE J., 47-5, 1016, (2001)

PRICE, P.; CAIRNCROSS, R. A. **Optimization of single-zone drying of polymer solution coatings using mathematical modeling**. J. Appl. Poly. Sci., 78, 149, (2000)

PRICE, P.; ROMDHANE, I. H. **Multicomponent diffusion theory and its application to polymer-solvent systems**. AIChE J., 49-2, 309, (2003)

RAMESH, N.; DUDA, J. L**. Analisys of a gap dryer used to produce polymer films and coatings**. AIChE J., 47-5, 972 , (2001)

SAURE, R.; WAGNER, G. R.; SCHLUNDER, E. U**. Drying of solvent-borne polymeric coating:** I. Modeling the drying process. Surface & Coatings Technology, 99, 253, (1998)

STASTNA, J.; KEE, D. **Transport properties in polymers.** Technomic Publishing Company, (1995) SCRIVEN, L. E**. Intermediate fluid mechanics.** Course notes, (1989)

VAN NESS, H. C.; SMITH, J. M. **Introduction to chemical engineering thermodynamics.** McGraw-Hill, (1975)

VERNERET, H. **Solventes industriais - propriedades e aplicações.** Toledo, (1984)

VINJAMUR, M.; CAIRNCROSS, R. A**. Non Fickian nonisothermal model for drying of polymer coatings.** AIChE J., 48-11, 2444, (2002)

VRENTAS, J. S.; DUDA, J. L**. Diffusion of small molecules in amorphous polymers.** Macromolecules, 9, 785, (1976)

VRENTAS, J. S.; DUDA, J. L.; LING, H. C**. Enhancement of impurity removal from polymer films.** J. Appl. Poly. Sci., 30, 4499, (1985)

VRENTAS, J. S.; VRENTAS, C. M**. A new equation relating self-diffusion and mutual diffusion coeficients in polymer solvent systems.** Macromolecules, 26, 6129, (1993)

VRENTAS, J. S.; VRENTAS, C. M. **Drying of solvent coated polymer films**. J. Poly. Sci. Part B; Poly. Phys., 32, 187, (1994)

VRENTAS, J. S.; VRENTAS, C. M. **Prediction of mutual diffusion coeficients for polymer-solvent systems.** J. Appl. Poly. Sci., 77, 3195, (2000)

YAPEL, R. A. **A physical model of the drying of coated films.** MS thesis, University of Minnesota, (1988)

YAPEL, R. A. et al. **Mutual and self-diffusion of water in gelatin:** experimental measurement and predictive test of free-volume theory. Polymer, vol 35, 11, 2411, (1994)

ZIELINSKI, J. M.; ALSOY, S. **Onsager consistency checks to multicomponent diffusion models.** J. Poly. Sci. Part B; Poly. Phys., 39, 1496, (2001)

ZIELINSKI, J. M.; HANLEY, B. F. **Pratical friction based approach to modelling multicomponent diffusion.** AIChE J., 45, 1, (1999)

7 Apêndice

Nesta seção serão apresentados a estrutura de dados usada para a programação do simulador e o código *Fortran* que geraram os resultados apresentados nesta dissertação.

7.1 Estrutura de dados

$\underline{\underline{D_i}}$ =

Nó	D_1	D_2
1	$D_{1,1}$	$D_{2,1}$
2		
.		
n		

$\underline{\underline{D}}$ =

Nó	D_{11}	D_{12}	D_{21}	D_{22}
1	$D_{11,1}$	$D_{12,1}$	$D_{21,1}$	$D_{22,1}$
2				
.				
n				

$\underline{\underline{D_m}}$ =

Nó	D_{m11}	D_{m12}	D_{m21}	D_{m22}
1	$D_{m11,1}$	$D_{m12,1}$	$D_{m21,1}$	$D_{m22,1}$
2				
.				
n				

$D_{m,ij}=0,5(D_{i+1,}j+D_{ij})$

Di = coef. difusão espontânea
Dij = coef. difusão mútua
ρ *= concentração*

dDdX =

	$\rho_{1,1}$	$\rho_{1,2}$	$\cdots$	$\rho_{1,n}$	$\rho_{2,1}$	$\rho_{2,2}$	$\cdots$	$\rho_{2,n}$	T
$D_{11,1}$	$\frac{\partial D_{11,1}}{\partial \rho_{1,1}}$	0	0	0	$\frac{\partial D_{11,1}}{\partial \rho_{2,1}}$	0	0	0	$\frac{\partial D_{11,1}}{\partial T}$
$D_{11,2}$									
...									
$D_{11,n}$									
$D_{12,1}$									
$D_{12,2}$									
...									
$D_{12,n}$									
$D_{21,1}$									
$D_{21,2}$									
...									
$D_{21,n}$									
$D_{22,1}$									
$D_{22,2}$									
...									
$D_{22,n}$									

$$dDidX = \begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,n} & \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,n} & T \\ \hline
D_{1,1} & \frac{\partial D_{1,1}}{\partial \rho_{1,1}} & 0 & 0 & 0 & \frac{\partial D_{1,1}}{\partial \rho_{2,1}} & 0 & 0 & 0 & \frac{\partial D_{1,1}}{\partial T} \\ \hline
D_{1,2} & & & & & & & & & \\ \hline
\ldots & & & & & & & & & \\ \hline
D_{1,n} & & & & & & & & & \\ \hline
D_{2,1} & & & & & & & & & \\ \hline
D_{2,2} & & & & & & & & & \\ \hline
\ldots & & & & & & & & & \\ \hline
D_{2,n} & & & & & & & & & \\ \hline
\end{array}$$

$$dDmdX = \begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,n} & \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,n} & T \\ \hline
D_{m11,1} & \frac{\partial D_{m11,1}}{\partial \rho_{1,1}} & \frac{\partial D_{m11,1}}{\partial \rho_{1,2}} & 0 & 0 & \frac{\partial D_{m11,1}}{\partial \rho_{2,1}} & \frac{\partial D_{m11,1}}{\partial \rho_{2,2}} & 0 & 0 & \frac{\partial D_{m11,1}}{\partial T} \\ \hline
D_{m11,2} & & & & & & & & & \\ \hline
\ldots & & & & & & & & & \\ \hline
D_{m11,n} & & & & & & & & & \\ \hline
D_{m12,1} & & & & & & & & & \\ \hline
D_{m12,2} & & & & & & & & & \\ \hline
\ldots & & & & & & & & & \\ \hline
D_{m12,n} & & & & & & & & & \\ \hline
D_{m21,1} & & & & & & & & & \\ \hline
D_{m21,2} & & & & & & & & & \\ \hline
\ldots & & & & & & & & & \\ \hline
D_{m21,n} & & & & & & & & & \\ \hline
D_{m22,1} & & & & & & & & & \\ \hline
D_{m22,2} & & & & & & & & & \\ \hline
\ldots & & & & & & & & & \\ \hline
D_{m22,n} & & & & & & & & & \\ \hline
\end{array}$$

$R =$

Nó	
1	Eq. Resíduo 1
2	Eq. Resíduo 2
...	...
n	Eq. Resíduo n
1	Eq. Resíduo n+1
2	Eq. Resíduo n+2
...	...
n	Eq. Resíduo 2n
1	Eq. Resíduo 2n+1
2	Eq. Resíduo 2n+2
...	...
n	Eq. Resíduo 3n
Temp	Eq. Resíduo 3n+1

$X =$

Nó	Var	*Tempo anterior*	*Tempo atual*
1	*1*	$\rho^{k}_{1,1}$	$\rho_{1,1}$
2	*2*	$\rho^{k}_{1,2}$	$\rho_{1,2}$
...	...	...	...
n	*n*	$\rho^{k}_{1,n}$	$\rho_{1,n}$
1	*n+1*		
2	*n+2*		
...	...		
n	*2n*		
1	*2n+1*	Z^{K}_{1}	Z_{1}
2	*2n+2*		
...	...		
n	*3n*		
Temp	*3n+1*	T^{k}	T

$C =$

Nó	Var	*Tempo 1*			*Tempo final*
1	*1*	$\rho^{1}_{1,1}$	$\rho^{2}_{1,1}$		$\rho^{t}_{1,1}$
2	*2*	$\rho^{1}_{1,2}$	$\rho^{2}_{1,2}$		$\rho^{t}_{1,2}$
...	...	...	...		...
n	*n*	$\rho^{1}_{1,n}$	$\rho^{2}_{1,n}$		$\rho^{t}_{1,n}$
1	*n+1*				
2	*n+2*				
...	...				
n	*2n*				
1	*2n+1*	Z^{1}_{1}	Z^{2}_{1}		Z^{t}_{1}
2	*2n+2*				
...	...				
n	*3n*				
Tem	*3n+1*	T^{1}	T^{2}		T^{t}

z_i = *posição do nó i*

$J =$

Eq. Res,	*1*	*2*	...	*n*	*n+1*	*n+2*	...	*2n*	*2n+1*	*2n+2*	...	*3n*	*3n+1*
1	$\frac{\partial R_1}{\partial \rho_{1,1}}$	$\frac{\partial R_1}{\partial \rho_{1,2}}$			$\frac{\partial R_1}{\partial \rho_{2,1}}$	$\frac{\partial R_1}{\partial \rho_{2,2}}$			$\frac{\partial R_1}{\partial z_1}$	$\frac{\partial R_1}{\partial z_2}$			$\frac{\partial R_1}{\partial T}$
2													
...													
n													
n+1													
n+2													
...													
2n													
2n+1													
2n+2													
...													
3n													
3n+1													

$$dPdX = \begin{array}{|c|c|}\hline \frac{\partial P_1}{\partial \rho_{1,n}} & \frac{\partial P_2}{\partial \rho_{1,n}} \\ \hline \frac{\partial P_1}{\partial \rho_{2,n}} & \frac{\partial P_2}{\partial \rho_{2,n}} \\ \hline \frac{\partial P_1}{\partial T} & \frac{\partial P_2}{\partial T} \\ \hline \end{array} \qquad dPdT = \begin{array}{|c|c|}\hline \frac{\partial P_1}{\partial T} & \frac{\partial P_2}{\partial T} \\ \hline \end{array}$$

Pi = pressão parcial do solvente i

$$dKdT = \begin{array}{|c|c|}\hline \frac{dK_1}{dT} & \frac{dK_2}{dT} \\ \hline \end{array}$$

Ki = coeficiente de transferência de massa de i

SolRes =

Passo de tempo	*Solv. Residual 1*	*Solv. Residual 2*
1		
2		
.		
t		

Tbb =

Passo de tempo	*Temp. bolha da solução*
1	
2	
.	
t	

dmidro = *a = atividade*

Nó	$\frac{\partial a_1}{\partial \rho_1}$	$\frac{\partial a_1}{\partial \rho_2}$	$\frac{\partial a_2}{\partial \rho_1}$	$\frac{\partial a_2}{\partial \rho_2}$	$\frac{\partial a_1}{\partial T}$	$\frac{\partial a_1}{\partial T}$
1						
2						
...						
n						

XChi =

Nó	χ_{13}	χ_{23}	χ_{12}
1			
2			
...			
n			

χ_{13} =*parâmetro de interação solvente 1 – polímero*

χ_{23} =*parâmetro de interação solvente 2 – polímero*

χ_{12} =*parâmetro de interação solvente 1 – solvente 2*

Xchib =

Nó	χ_{13}	χ_{23}	χ_{12}
1			

$dXChidX =$

Nó	$\frac{\partial\chi_{13}}{\partial\rho_1}$	$\frac{\partial\chi_{13}}{\partial\rho_2}$	$\frac{\partial\chi_{13}}{\partial T}$	$\frac{\partial\chi_{23}}{\partial\rho_1}$	$\frac{\partial\chi_{23}}{\partial\rho_2}$	$\frac{\partial\chi_{23}}{\partial T}$	$\frac{\partial\chi_{12}}{\partial\rho_1}$	$\frac{\partial\chi_{12}}{\partial\rho_2}$	$\frac{\partial\chi_{12}}{\partial T}$
1									
2									
...									
n									

$dXChibdX =$

Nó	$\frac{\partial\chi_{13}}{\partial\rho_1}$	$\frac{\partial\chi_{13}}{\partial\rho_2}$	$\frac{\partial\chi_{13}}{\partial T}$	$\frac{\partial\chi_{23}}{\partial\rho_1}$	$\frac{\partial\chi_{23}}{\partial\rho_2}$	$\frac{\partial\chi_{23}}{\partial T}$	$\frac{\partial\chi_{12}}{\partial\rho_1}$	$\frac{\partial\chi_{12}}{\partial\rho_2}$	$\frac{\partial\chi_{12}}{\partial T}$
1									

$dactdro =$

$\frac{\partial a_{1,n}}{\partial\rho_{1,n}}$	$\frac{\partial a_{2,n}}{\partial\rho_{1,n}}$
$\frac{\partial a_{1,n}}{\partial\rho_{2,n}}$	$\frac{\partial a_{2,n}}{\partial\rho_{2,n}}$
$\frac{\partial a_{1,n}}{\partial T}$	$\frac{\partial a_{2,n}}{\partial T}$

$dPvdT =$

$\frac{dP_{v1}}{dT}$	$\frac{dP_{v2}}{dT}$

Pv = pressão de vapor

$dmi2dX2 =$

Nó	$\frac{\partial A_{11}}{\partial\rho_1}$	$\frac{\partial A_{11}}{\partial\rho_2}$	$\frac{\partial A_{11}}{\partial T}$	$\frac{\partial A_{12}}{\partial\rho_1}$	$\frac{\partial A_{12}}{\partial\rho_2}$	$\frac{\partial A_{12}}{\partial T}$	$\frac{\partial A_{21}}{\partial\rho_1}$	$\frac{\partial A_{21}}{\partial\rho_2}$	$\frac{\partial A_{21}}{\partial T}$	$\frac{\partial A_{22}}{\partial\rho_1}$	$\frac{\partial A_{22}}{\partial\rho_2}$	$\frac{\partial A_{22}}{\partial T}$
1												
2												
...												
n												

$$A_{ij} = \frac{\partial \ln a_i}{\partial \rho_j}$$

$Alfa =$

Nó	α_1	α_2
1		
2		
...		
n		

α_1 *e* α_2 *= parâmetros do modelo proposto por Price*

$dAldro =$

Nó	$\frac{\partial\alpha_1}{\partial\rho_1}$	$\frac{\partial\alpha_1}{\partial\rho_2}$	$\frac{\partial\alpha_2}{\partial\rho_1}$	$\frac{\partial\alpha_2}{\partial\rho_2}$
1				
2				
...				
n				

dwdro =

Nó	$\frac{\partial w_1}{\partial \rho_1}$	$\frac{\partial w_1}{\partial \rho_2}$	$\frac{\partial w_2}{\partial \rho_1}$	$\frac{\partial w_2}{\partial \rho_2}$	$\frac{\partial w_3}{\partial \rho_1}$	$\frac{\partial w_3}{\partial \rho_2}$
1						
2						
...						
n						

w = fração em massa

dVfhdX =

Nó	$\frac{\partial \frac{V_{FH}}{r}}{\partial \rho_1}$	$\frac{\partial \frac{V_{FH}}{r}}{\partial \rho_2}$	$\frac{\partial \frac{V_{FH}}{r}}{\partial T}$
1			
2			
...			
n			

$\frac{V_{FH}}{r}$ *= parâmetro do modelo do volume livre de Vrentas*

7.2 Código Fortran

```
c******************************************************************
c
      Program SmartDrying
c
c******************************************************************
c
c     Programmer: Eduardo de B. Perez
c     Revision: Oct, 2003
c     email: ebperez@mmm.com
c
c******************************************************************
c
c     Objective:
c
c     Simulate drying process in films of polymeric solution consisting
c      of 1 polymer in 1 or 2 solvents.Substrate is considered
c      impermeable and at maximum of 6 drying zones can be used.
c
c     Newton Method is used to solve the nonlinear algebric equations
c        emerging from mass and energy conservation laws and the finite
c        diferences method is used to domain discretization .
c
c******************************************************************
c-----------------------------------------------------------------------

c.....Variables declaration
c-----------------------------------------------------------------------
      parameter (nnodesmax = 300,tnodesmax = 40000)
c
      integer nnodes,tnodes,DifModel,itime,ttnodes
      real*8 V1,V2,V3,MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,
     +         X23,X12,c01,c02,e0,T0,HSub,RoSub,cpSub,RoAir,cpAir,
     +       t1,hs1,hi1,Ta1s,Ta1i,P11,P21,t2,hs2,hi2,Ta2s,Ta2i,P12,P22,
     +       t3,hs3,hi3,Ta3s,Ta3i,P13,P23,t4,hs4,hi4,Ta4s,Ta4i,P14,P24,
     +       t5,hs5,hi5,Ta5s,Ta5i,P15,P25,t6,hs6,hi6,Ta6s,Ta6i,P16,P26,
     +       A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,a,a11,a12,a13,D01,D02,b21,
     +       b22,b23,V1crt,V2crt,V3crt,eps13,eps23,hs,hi,Tas,Tai,P1a,
     +       P2a,P1,P2,VP1,VP2,act1,act2,NR,K1,K2,X12_0,X12_1,X12_2,
     +       X12_3,X13_0,X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +       A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,An,Eact1,Eact2

c
      real*8, allocatable, dimension(:,:):: J,Dm,X,Di,dmidro,C,
     +                                 SolRes,Alpha,dXChidX,XChi
      real*8, allocatable, dimension(:) :: tm,R,z,Tbbmin,Tbb
c
c-----------------------------------------------------------------------
c...Vi            :Specific Volume Component i          [cm3/g]
c...MMi           :Molar Mass Component i               [g/mol]
c...DelHi         :Heat of Vaporization Solvent i         [J/g]
c...RoBar         :Average Density of Solution          [g/cm3]
c...cpLBar        :Average Specific Heat of Solution      [J/g.K]
c...Xij           :Interaction paramenter between i and j
c...c0i           :Initial Concentration Component i      [g/cm3]
c...e0,T0         :Initial Thickness and Temperature      [cm],[K]
c...HSub          :Thickness of Substrate                         [cm]
```

```
c...RoSub         :Density of Substrate                    [g/cm3]
c...cpSub         :Specific Heat of Substrate              [J/g.K]
c...RoAir,cpAir   :Density and Specific Heat of air    [g/cm3],[J/g.K]
c...ti            :Residence Time on Zone i                [s]
c...hsj,hij       :Heat Transfer Coef (Top/Bottom] zone j  [W/cm2.K]
c...Tai           :Air Temperature zone i                  [K]
c...P1i,P2i       :Partial Pressure Solvent 1/2 on zone i    [g/cm.s2]
c...Ai,Bi,Ci,Di,Ei:Antoine Coef for Vapor Pressure
c...Pi            :Current Partial Pressure Solvent i      [g/cm.s2]
c...VPi           :Vapor Pressure Solvent i                [g/cm.s2]
c...acti          :Activity on Interface Solvent i
c...Ki            :Mass Transfer Coef Solvent i            [s/cm]
c...a,b           :Distribution parameters for spatial and time meshs
c...a11,a12,a13,
c...D01,D02,b21,
c...b22,b23,
c...V1crt,V2crt,
c...V3crt,eps13,
c...eps23       : Free Volume Parameters
c----------------------------------------------------------------------
c.....Arrays
c----------------------------------------------------------------------
c.....J : Jacobian of equations
c.....tm: Time mesh
c.....X : Accumulates solution of last time step and guessing of current
c          time step
c.....Dm: Mean Mutual Diffusion Coef
c.....R : Residues of equations
c.....z : Spatial mesh
c.....Di: Self Diffusion Coef
c.....C : Solution for each time step
c....Tbb: Bubble point temperature to each solvent
c.SolRes: Residual solvent
c----------------------------------------------------------------------
c----------------------------------------------------------------------
c....InputData read a txt file with all input data needed including
c.... coating properties, process parameters and diffusion parameters.
c----------------------------------------------------------------------
c
      write(*,*) 'SMARTDRYING '
c
      call InputData(V1,V2,V3,MM1,MM2,DelH1,DelH2,RoBar,
     +                   cpLBar,X13,X23,X12,c01,c02,e0,T0,HSub,RoSub,
     +                   cpSub,RoAir,cpAir,t1,hs1,hi1,Ta1s,Ta1i,P11,
     +                   P21,t2,hs2,hi2,Ta2s,Ta2i,P12,P22,t3,hs3,hi3,
     +                   Ta3s,Ta3i,P13,P23,t4,hs4,hi4,Ta4s,Ta4i,P14,
     +                   P24,t5,hs5,hi5,Ta5s,Ta5i,P15,P25,t6,hs6,hi6,
     +                   Ta6s,Ta6i,P16,P26,A1,B1,C1,D1,E1,A2,
     +                   B2,C2,D2,E2,nnodes,a,tnodes,a11,a12,a13,
     +                   D01,D02,b21,b22,b23,V1crt,V2crt,V3crt,eps13,
     +                   eps23,DifModel,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                   X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                   A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,An,Eact1,Eact2)
c
      allocate(J(3*nnodes+1,3*nnodes+1))
      allocate(Dm(nnodes-1,4))
      allocate(X(3*nnodes+1,2))
      allocate(Di(nnodes,2))
      allocate(dmidro(nnodes,6))
      allocate(Alpha(nnodes,2))
      allocate(dXChidX(nnodes,9))
      allocate(XChi(nnodes,3))
      allocate(C(3*nnodes+1,40000))
      allocate(Tbb(40000))
      allocate(SolRes(40000,2))
      allocate(tm(6*tnodes+1))
      allocate(R(3*nnodes+1))
      allocate(z(nnodes))
      allocate(Tbbmin(2))
c
c-----------------------------------------------------------------------
c.....Nmesh sets the first spatial mesh
c-----------------------------------------------------------------------
      call nmesh(e0,nnodes,a,
     +           z)
c
c-----------------------------------------------------------------------
c.....Tmesh sets the time mesh
c-----------------------------------------------------------------------
      call tmesh (t1,t2,t3,t4,t5,t6,tnodes,
      +           tm,ttnodes)
c
c-----------------------------------------------------------------------
c.....Icond saves initial conditions on solution array C
c-----------------------------------------------------------------------
      call Icond(nnodes,ttnodes,DifModel,itime,c01,c02,z,T0,V1,
     +            V2,e0,MM1,MM2,X13,X23,X12,A1,B1,C1,D1,E1,A2,B2,
     +             C2,D2,E2,X,XChi,X12_0,X12_1,X12_2,X12_3,X13_0,
     +             X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +             C,Tbb,SolRes,Tbbmin)
c
c-----------------------------------------------------------------------
c.....Start of time loop
c-----------------------------------------------------------------------
      do itime=2,ttnodes+1
c
c-----------------------------------------------------------------------
c.....ZoneSettgs sets the current drying process parameters
c-----------------------------------------------------------------------
      call ZoneSettgs(itime,hs1,hs2,hs3,hs4,hs5,hs6,hi1,hi2,hi3,
     +                    hi4,hi5,hi6,Ta1s,Ta2s,Ta3s,Ta4s,Ta5s,Ta6s,
     +                    Ta1i,Ta2i,Ta3i,Ta4i,Ta5i,Ta6i,P11,
     +                    P21,P12,P22,P13,P23,P14,P24,P15,P25,P16,
     +                    P26,tnodes,
     +                    hs,hi,Tas,Tai,P1a,P2a)
c
c-----------------------------------------------------------------------
c.....Guess takes the solution of last time step as guessing of current
c......time step
c-----------------------------------------------------------------------
      call guess (itime,nnodes,ttnodes,C,tm,
     +            X)
c
c-----------------------------------------------------------------------
c.....FormR calculates the residues of equations
c-----------------------------------------------------------------------
      call FormR(nnodes,itime,X,tm,hs,hi,Tas,Tai,P1a,P2a,V1,V2,V3,MM1,
     +               MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,X12,
     +               HSub,RoSub,cpSub,RoAir,cpAir,A1,B1,C1,D1,E1,A2,
     +               B2,C2,D2,E2,a11,a12,a13,D01,D02,b21,b22,b23,
     +               V1crt,V2crt,V3crt,eps13,eps23,DifModel,a,tnodes,
     +               X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3,
     +               X23_0,X23_1,X23_2,X23_3,XChi,dXChidX,Alpha,An,
     +               A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,Eact1,Eact2,
     +               R,P1,P2,VP1,VP2,act1,act2,Dm,Di,dmidro,K1,K2)

c
c-----------------------------------------------------------------------
c.....Start of Newton loop
c-----------------------------------------------------------------------
      NR=1
      iter=1
c
      do while ((NR>0.000001).and.(iter.lt.50))
c
c-----------------------------------------------------------------------
c.....FormJ calculates the Jacobian of equations
c-----------------------------------------------------------------------
      call FormJnew(nnodes,itime,X,tm,hs,hi,Tas,Tai,P1a,P2a,V1,V2,V3,
     +               MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,X12,A1_1,
     +               A1_2,A2_1,A2_2,An,HSub,RoSub,cpSub,A1,B1,C1,D1,E1,
     +               A2,B2,C2,D2,E2,P1,P2,a11,a12,a13,D01,D02,b21,b22,
     +               b23,V1crt,V2crt,V3crt,eps13,eps23,DifModel,a,Di,
     +               dmidro,VP1,VP2,act1,act2,K1,K2,Dm,tnodes,RoAir,
     +               cpAir,dXChidX,XChi,Alpha,Eact1,Eact2,
     +               J)
c
c-----------------------------------------------------------------------
c.....Solver improves the guessing of solution
c-----------------------------------------------------------------------

      call Solver(nnodes,J,R,
     +            X)
c
c
      call FormR(nnodes,itime,X,tm,hs,hi,Tas,Tai,P1a,P2a,V1,V2,V3,MM1,
     +               MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,X12,
     +               HSub,RoSub,cpSub,RoAir,cpAir,A1,B1,C1,D1,E1,A2,
     +               B2,C2,D2,E2,a11,a12,a13,D01,D02,b21,b22,b23,
     +               V1crt,V2crt,V3crt,eps13,eps23,DifModel,a,tnodes,
     +               X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3,
     +               X23_0,X23_1,X23_2,X23_3,XChi,dXChidX,Alpha,An,
     +               A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,Eact1,Eact2,
     +               R,P1,P2,VP1,VP2,act1,act2,Dm,Di,dmidro,K1,K2)
c
      NR = dnrm2(3*nnodes+1, R, 1)
c
      iter=iter+1
c
c-----------------------------------------------------------------------
c.....End of Newton loop
c-----------------------------------------------------------------------
      end do
c
      if (iter.gt.49) then
      write(*,*) "Did not converge"
      write(*,*) "Time step:",itime
      write(*,*) "Newton Iteraction:",iter
      pause
      stop
      endif
c
c-----------------------------------------------------------------------
```

```
c.....TBubble calculates bubble point temperature to each solvent
c-----------------------------------------------------------------------
c
      call TBubble(itime,DifModel,ttnodes,nnodes,A1,B1,C1,D1,E1,
     +                   A2,B2,C2,D2,E2,T0,c01,c02,X,V1,V2,MM1,MM2,
     +                   X13,X23,X12,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                   X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                   Tbb,Tbbmin)

c-----------------------------------------------------------------------
c.....StoreSolution saves the solution of current step and print to
screen
c-----------------------------------------------------------------------
      call StoreSolution(nnodes,itime,ttnodes,X,
     +                   C,Tbb,SolRes)
c-----------------------------------------------------------------------
c.....End of time loop
c-----------------------------------------------------------------------
      enddo
c
c-----------------------------------------------------------------------
c.....PostPro formats the data to generate reports and graphics
c-----------------------------------------------------------------------
      call  PostPro(C,nnodes,ttnodes,tm,Tbb,SolRes,Tbbmin,Ta1s,Ta1i)
c
      end program





c***********************************************************************
c
      subroutine InputData(V1,V2,V3,MM1,MM2,DelH1,DelH2,RoBar,
     +                  cpLBar,X13,X23,X12,c01,c02,e0,T0,HSub,RoSub,
     +                    cpSub,RoAir,cpAir,t1,hs1,hi1,Ta1s,Ta1i,P11,
     +                    P21,t2,hs2,hi2,Ta2s,Ta2i,P12,P22,t3,hs3,hi3,
     +                    Ta3s,Ta3i,P13,P23,t4,hs4,hi4,Ta4s,Ta4i,P14,
     +                    P24,t5,hs5,hi5,Ta5s,Ta5i,P15,P25,t6,hs6,hi6,
     +                    Ta6s,Ta6i,P16,P26,A1,B1,C1,D1,E1,A2,
     +                    B2,C2,D2,E2,nnodes,a,tnodes,a11,a12,a13,
     +                    D01,D02,b21,b22,b23,V1crt,V2crt,V3crt,eps13,
     +                    eps23,DifModel,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                    X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                    A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,An,Eact1,Eact2)

c
c************************************************************************
c***********************************************************************
c
c.....Reads all parameters from a txt file
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer nnodes,tnodes,DifModel
      real*8 V1,V2,V3,MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,
     +      X12,c01,c02,e0,T0,HSub,RoSub,cpSub,RoAir,cpAir,
     +      t1,hs1,hi1,Ta1s,Ta1i,P11,P21,t2,hs2,hi2,Ta2s,Ta2i,P12,P22,
     +      t3,hs3,hi3,Ta3s,Ta3i,P13,P23,t4,hs4,hi4,Ta4s,Ta4i,P14,P24,
     +       t5,hs5,hi5,Ta5s,Ta5i,P15,P25,t6,hs6,hi6,Ta6s,Ta6i,P16,P26,
     +       A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,a,a11,a12,a13,
     +       D01,D02,b21,b22,b23,V1crt,V2crt,V3crt,eps13,eps23,
     +       X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3,X23_0,
     +       X23_1,X23_2,X23_3,A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,An,Eact1,
     +       Eact2
c
c-----------------------------------------------------------------------
c.....File
c-----------------------------------------------------------------------
      open(UNIT=1,FILE='InputData.txt',STATUS='OLD')
c
c-----------------------------------------------------------------------
c.....Thermodynamic Properties of Components
c-----------------------------------------------------------------------
      read(1,10) V1,V2,V3
   10 format(7(/),63x,F8.3,2(/,63x,F8.3))
c
      read(1,20) MM1,MM2,DelH1,DelH2
   20 format(2(/,63x,F9.3),2(/),63x,F8.3,/,63x,F8.3)
c
c-----------------------------------------------------------------------

c.....Vapor Pressure Model
c-----------------------------------------------------------------------
      read(1,30) A1,B1,C1,D1,E1
   30 format(5(/),63x,F10.5,4(/,63x,F10.5))
c
      read(1,40) A2,B2,C2,D2,E2
   40 format(3(/),63x,F10.5,4(/,63x,F10.5))
c
c-----------------------------------------------------------------------
c.....Substrate Properties
c-----------------------------------------------------------------------
      read(1,50) HSub,RoSub,cpSub
   50 format(3(/),63x,F8.3,2(/,63x,F8.3))
c
c-----------------------------------------------------------------------

c.....Air Properties
c-----------------------------------------------------------------------
      read(1,60) RoAir,CpAir
   60 format(3(/),63x,F8.3,/,63x,F8.3)
c
c-----------------------------------------------------------------------
c.....Thermodynamic Properties of Solution
c-----------------------------------------------------------------------
      read(1,70) RoBar,cpLBar
   70 format(3(/),63x,F8.3,/,63x,F8.3)
c
c-----------------------------------------------------------------------
c.....Interaction Factors to Diffusion Models 1 to 5
c-----------------------------------------------------------------------
      read(1,80) X13,X23,X12
   80 format(3(/),63x,F8.3,2(/,63x,F8.3))
c
c-----------------------------------------------------------------------
c.....Parameters to calculate Interaction Factors to Diffusion Models 6
c-----------------------------------------------------------------------
      read(1,90) X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3
   90 format(3(/),63x,F10.6,3(/,63x,F10.6),2(/),63x,F10.6,
    +       3(/,63x,F10.6))
```

```
      read(1,100) X23_0,X23_1,X23_2,X23_3
  100 format(/,63x,F10.6,3(/,63x,F10.6))
c
c-----------------------------------------------------------------------
c.....Free volume parameters
c-----------------------------------------------------------------------
c
      read(1,110) a11,a12,a13,D01,D02,b21,b22,b23
  110 format(3(/),63x,F9.7,4(/,63x,F9.7),3(/,63x,F9.7))
c
      read(1,120) V1crt,V2crt,V3crt,eps13,eps23,Eact1,Eact2
  120 format(63x,F9.7,6(/,63x,F9.7))
c
      read(1,130) A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,An
  130 format(3(/),63x,F10.6,2(/,63x,F10.6),2(/),63x,F10.6,
     +       2(/,63x,F10.6),2(/),63x,F6.3)

c
c-----------------------------------------------------------------------

c.....Initial Conditions
c-----------------------------------------------------------------------
      read(1,140) c01,c02,e0,T0
  140 format(3(/),63x,F12.10,1(/,63x,F12.10),/,63x,F8.6,/,63x,F8.3)
c
c-----------------------------------------------------------------------
c.....Process parameters
c-----------------------------------------------------------------------
      read(1,150) t1,hs1,hi1,Ta1s,Ta1i,P11,P21
  150 format(5(/),63x,F8.3,2(/,63x,F10.9),4(/,63x,F8.3))
c
      read(1,160) t2,hs2,hi2,Ta2s,Ta2i,P12,P22
  160 format(3(/),63x,F8.3,2(/,63x,F10.9),4(/,63x,F8.3))
c
      read(1,170) t3,hs3,hi3,Ta3s,Ta3i,P13,P23
  170 format(3(/),63x,F8.3,2(/,63x,F10.9),4(/,63x,F8.3))
c
      read(1,180) t4,hs4,hi4,Ta4s,Ta4i,P14,P24
  180 format(3(/),63x,F8.3,2(/,63x,F10.9),4(/,63x,F8.3))
c
      read(1,190)t5,hs5,hi5,Ta5s,Ta5i,P15,P25
  190 format(3(/),63x,F8.3,2(/,63x,F10.9),4(/,63x,F8.3))
c
      read(1,200)t6,hs6,hi6,Ta6s,Ta6i,P16,P26
  200 format(3(/),63x,F8.3,2(/,63x,F10.9),4(/,63x,F8.3))
c
c-----------------------------------------------------------------------
c.....Spatial and time meshs
c-----------------------------------------------------------------------

      read(1,210) nnodes,a,tnodes
  210 format(3(/),63x,i4,/,63x,F8.3,/,63x,i5)
c
c-----------------------------------------------------------------------
c.....Diffusion Model
c-----------------------------------------------------------------------
      read(1,220) DifModel
  220 format(/,63x,i1)
c-----------------------------------------------------------------------
      endsubroutine
```

```
c*********************************************************************
c
      subroutine nmesh(e0,nnodes,a,
     +                   z)
c
c*********************************************************************
c*********************************************************************
c
c.....Initial spatial mesh
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 a,e0,
     +       z(nnodes)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 b,c
c
c---------------------------------------------------------------------
      z(nnodes)=0
c
      do inode=1,nnodes
c
      b=nnodes-inode
      c=nnodes-1
      z(inode)= e0*(1-(b/c)**a)
c
      enddo
c
      endsubroutine




c*********************************************************************
c
      subroutine tmesh (t1,t2,t3,t4,t5,t6,tnodes,
     +                  tm,ttnodes)
c
c*********************************************************************
c*********************************************************************
c
c.....Calculate the time mesh
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer tnodes,ttnodes
      real*8 t1,t2,t3,t4,t5,t6,
     +       tm(6*tnodes+1)
c
c---------------------------------------------------------------------
      tm=0
```

```
c
      do inode=2,tnodes+1
c
      tm(inode)=(inode-1)*t1/tnodes
c
      enddo
c
      do inode=tnodes+2,2*tnodes+1
c
      tm(inode)=tm(tnodes+1)+(inode-tnodes-1)*t2/tnodes
c
      enddo
c
      do inode=2*tnodes+2,3*tnodes+1
c
      tm(inode)=tm(2*tnodes+1)+(inode-2*tnodes-1)*t3/tnodes
c
      enddo
c
      do inode=3*tnodes+2,4*tnodes+1
c
      tm(inode)=tm(3*tnodes+1)+(inode-3*tnodes-1)*t4/tnodes
c
      enddo
c
      do inode=4*tnodes+2,5*tnodes+1
c
      tm(inode)=tm(4*tnodes+1)+(inode-4*tnodes-1)*t5/tnodes
c
      enddo
c
      do inode=5*tnodes+2,6*tnodes+1
c
      tm(inode)=tm(5*tnodes+1)+(inode-5*tnodes-1)*t6/tnodes
c
      enddo
c
      ttnodes=tnodes
c
      if ((t2.gt.0).and.(t3.eq.0)) then
c
      ttnodes=2*tnodes
c
      else if ((t3.gt.0).and.(t4.eq.0)) then
c
      ttnodes=3*tnodes
c
      else if ((t4.gt.0).and.(t5.eq.0)) then
c
      ttnodes=4*tnodes
c
      else if ((t5.gt.0).and.(t6.eq.0)) then
c
      ttnodes=5*tnodes
c
      else if (t6.gt.0)then
c
      ttnodes=6*tnodes
c
      endif
```

```
c
      endsubroutine






c*******************************************************************
c
      subroutine Icond(nnodes,ttnodes,DifModel,itime,c01,c02,z,T0,V1,
     +                 V2,e0,MM1,MM2,X13,X23,X12,A1,B1,C1,D1,E1,A2,B2,
     +               C2,D2,E2,X,XChi,X12_0,X12_1,X12_2,X12_3,X13_0,
     +               X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +               C,Tbb,SolRes,Tbbmin)
c
c*******************************************************************
c*******************************************************************
c
c.....Saves the initial condition on solution matrix
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes,ttnodes,itime,DifModel
      real*8 c01,c02,z(nnodes),T0,V1,V2,MM1,MM2,X13,X23,X12,A1,B1,C1,D1,
     +       E1,A2,B2,C2,D2,E2,e0,SolRes(ttnodes,2),XChi(nnodes,3),
     +         X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,
     +       X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +       X(3*nnodes+1,2),C(3*nnodes+1,ttnodes),Tbb(ttnodes),
     +       Tbbmin(2)
c
c-------------------------------------------------------------------
c.....Local variables
c-------------------------------------------------------------------
      real*8 Phi1,Phi2
c
c-------------------------------------------------------------------
c
      itime=1
c
c-------------------------------------------------------------------
c.....Initial residual solvent         [g/cm²]
c-------------------------------------------------------------------
c
      SolRes(itime,1)=c01*e0
      SolRes(itime,2)=c02*e0
c
c
c-------------------------------------------------------------------
c.....Initial coating temperature and solvent concentrations
c-------------------------------------------------------------------
c
      C(3*nnodes+1,1)=T0
c
      do inode=1,nnodes
c
      C(inode,1)=c01
      C(nnodes+inode,1)=c02
      C(2*nnodes+inode,1)=z(inode)
```

```
c
      enddo
c
     do inode=1,3*nnodes+1
c
      X(inode,2)=C(inode,1)
c
      enddo
c
      Phi1=X(nnodes,2)*V1
      Phi2=X(2*nnodes,2)*V2

c-----------------------------------------------------------------------
c.....Chi calculates interaction factors X12,X13,X23
c-----------------------------------------------------------------------
      call Chi(X,nnodes,X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,
     +        X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +        XChi)
cc-----------------------------------------------------------------------
c.....Initial bubble point temperature     [K]
c-----------------------------------------------------------------------
c
      call TBubble(itime,DifModel,ttnodes,nnodes,A1,B1,C1,D1,E1,
     +                   A2,B2,C2,D2,E2,T0,c01,c02,X,V1,V2,MM1,MM2,
     +                   X13,X23,X12,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                   X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                   Tbb,Tbbmin)
c
      endsubroutine




c***********************************************************************
c
      subroutine Chi(X,nnodes,X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,
     +              X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +              XChi)
c
c***********************************************************************
c
c.... Chi calculates the interaction factors X12,X13,X23
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,
     +      X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +      XChi(nnodes,3)
c
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c-----------------------------------------------------------------------
c
      do inode=1,nnodes
```

```
c
      XChi(inode,1)=X13_0+X13_1/X(3*nnodes+1,2)+X13_2*Phi1+X13_3*Phi2
      XChi(inode,2)=X23_0+X23_1/X(3*nnodes+1,2)+X23_2*Phi1+X23_3*Phi2
      XChi(inode,3)=X12_0+X12_1/X(3*nnodes+1,2)+X12_2*Phi1+X12_3*Phi2
c
      enddo
c
      endsubroutine






c***********************************************************************
c
      subroutine TBubble(itime,DifModel,ttnodes,nnodes,A1,B1,C1,D1,E1,
     +                   A2,B2,C2,D2,E2,T0,c01,c02,X,V1,V2,MM1,MM2,
     +                   X13,X23,X12,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                   X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                   Tbb,Tbbmin)
c
c***********************************************************************
c.....TBubble calculates the temperature that will lead partial pressure
c      to reach atmosphere pressure to each solvent.
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer itime,ttnodes,nnodes,DifModel
      real*8 A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,T0,c01,c02,V1,V2,MM1,MM2,X13,
     +       X23,X12,X12_0,X12_1,X12_2,X12_3,X13_0,
     +       X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +       X(3*nnodes+1,2),Tbb(ttnodes),Tbbmin(2)
c
c-----------------------------------------------------------------------
c.....Local variables
c-----------------------------------------------------------------------
      integer iter
      real*8     NR,act1,act2,CrtPhi1,CrtPhi2,XChib(3),
     +        Rtbb,Jtbb,Rtbbmin(2),Jtbbmin(2),VP1bb,VP2bb
c-----------------------------------------------------------------------
c.....Guessing solution bubble point temperature Tbb
c.....Pure solvents bubble point temperature Tbbmin(1) and Tbbmin(2)
c-----------------------------------------------------------------------
      if (itime.eq.1) then
c
      Tbb(itime)=T0
      Tbbmin(1)=T0
      Tbbmin(2)=T0
      CrtPhi1=c01
      CrtPhi2=c02
c-----------------------------------------------------------------------
c.....Activity of each solvent at substrate
c-----------------------------------------------------------------------
      if (DifModel.eq.6) then
c
      call IniActivityVarX(CrtPhi1,CrtPhi2,V1,V2,MM1,MM2,Tbb,ttnodes,
     +                         itime,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                         X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                       XChib,act1,act2)
c
      else
c
      call IniActivity(c01,c02,V1,V2,MM1,MM2,X13,X23,X12,
     +                act1,act2)
c
      endif
c
      else
c
      Tbb(itime)=Tbb(itime-1)
c
c----------------------------------------------------------------------
c.....Assuming that node close to substrate is critical to bubble
formation
c----------------------------------------------------------------------
      CrtPhi1=X(1,2)
      CrtPhi2=X(nnodes+1,2)
c---------------------------------------------------------------------
      if (DifModel.eq.6) then
c
      call IniActivityVarX(CrtPhi1,CrtPhi2,V1,V2,MM1,MM2,Tbb,ttnodes,
     +                      itime,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                       X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                       XChib,act1,act2)
c
      else
c
      call IniActivity(CrtPhi1,CrtPhi2,V1,V2,MM1,MM2,X13,
     +                  X23,X12,
     +                act1,act2)
c
      endif
c
      endif
c
c----------------------------------------------------------------------
c.....FormRtbb calculates the residues of equations to solution bubble
c.....temperature and boiling temperature of pure solvents
c----------------------------------------------------------------------
c
      call FormRtbb(itime,ttnodes,Tbb,A1,B1,C1,D1,E1,A2,B2,C2,D2,
     +               E2,act1,act2,
     +                Rtbb,Rtbbmin,Tbbmin,VP1bb,VP2bb)
c
c----------------------------------------------------------------------
c.....Start of Newton loop
c----------------------------------------------------------------------
      NR=1
      iter=1
c
      do while ((NR>0.000001).and.(iter.lt.20))
c
c----------------------------------------------------------------------
c.....FormJtbb calculates the Jacobian of equations
c----------------------------------------------------------------------
      call FormJTbb (nnodes,ttnodes,itime,X,V1,V2,MM1,MM2,A1,B1,
     +                  C1,D1,E1,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                 X12_1,X12_2,X12_3,A2,B2,C2,D2,E2,VP1bb,VP2bb,
     +                 act1,act2,XChib,DifModel,Tbb,
     +                 Jtbb,Tbbmin,Jtbbmin)
c
c-----------------------------------------------------------------------
c.....Improving the guessing of solution
c-----------------------------------------------------------------------

      Tbb(itime)=(-Rtbb/Jtbb)+Tbb(itime)
c
      if (DifModel.eq.6) then
c
c-----------------------------------------------------------------------
c.....Recalculating activity when its depends on temperature
c-----------------------------------------------------------------------
      call IniActivityVarX(CrtPhi1,CrtPhi2,V1,V2,MM1,MM2,Tbb,ttnodes,
     +                      itime,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                       X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                       XChib,act1,act2)
c
      endif
c
c-----------------------------------------------------------------------
      call FormRtbb(itime,ttnodes,Tbb,A1,B1,C1,D1,E1,A2,B2,C2,D2,
     +               E2,act1,act2,
     +                Rtbb,Rtbbmin,Tbbmin,VP1bb,VP2bb)
c
      NR = DABS(Rtbb)
      iter=iter+1
c-----------------------------------------------------------------------
c.....End of Newton loop
c-----------------------------------------------------------------------
      end do
c
      if (iter.gt.19) then
      write(*,*) "Bubble Temperature did not converge"
      write(*,*) "Time step:",itime
      write(*,*) "Newton Iteraction:",iter
      pause
      stop
      endif
c
      if (itime.eq.1) then
c-----------------------------------------------------------------------
c.....Start of Newton loop for boil temperature of pure solvents
c-----------------------------------------------------------------------
      NR=1
      iter=1
c
      do while ((NR>0.000001).and.(iter.lt.20))
c
c-----------------------------------------------------------------------
c.....FormJtbb calculates the Jacobian of equations
c-----------------------------------------------------------------------
      call FormJTbb (nnodes,ttnodes,itime,X,V1,V2,MM1,MM2,A1,B1,
     +                  C1,D1,E1,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                 X12_1,X12_2,X12_3,A2,B2,C2,D2,E2,VP1bb,VP2bb,
     +                 act1,act2,XChib,DifModel,Tbb,
     +                 Jtbb,Tbbmin,Jtbbmin)
c
c-----------------------------------------------------------------------
```

```
c.....Improving the guessing of solution
c-----------------------------------------------------------------------

      if (itime.eq.1) then
c
      Tbbmin(1)=(-Rtbbmin(1)/Jtbbmin(1))+Tbbmin(1)
      Tbbmin(2)=(-Rtbbmin(2)/Jtbbmin(2))+Tbbmin(2)
c
      endif
c
      call FormRtbb(itime,ttnodes,Tbb,A1,B1,C1,D1,E1,A2,B2,C2,D2,
     +               E2,act1,act2,
     +                Rtbb,Rtbbmin,Tbbmin,VP1bb,VP2bb)
c
      NR = dnrm2(2, Rtbbmin, 1)
      iter=iter+1
c
c-----------------------------------------------------------------------
c.....End of Newton loop
c-----------------------------------------------------------------------
      end do
c
      if (iter.gt.19) then
      write(*,*) "Minimum Bubble Temperature did not converge"
      write(*,*) "Time step:",itime
      write(*,*) "Newton Iteraction:",iter
      pause
      stop
      endif
c
      endif
c
      endsubroutine







c***********************************************************************
c
      subroutine IniActivityVarX(c01,c02,V1,V2,MM1,MM2,Tbb,ttnodes,
     +                       itime,X12_0,X12_1,X12_2,X12_3,X13_0,
     +                       X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                       XChib,act1,act2)
c
c***********************************************************************
c.....Flory-Huggins is used to determinate the activity of solvents.
c
c.....Assumptions:
c......1-Molar volume of polymer >> Molar volume of solvents;
c......2-Variable interaction parameters;
c......3-No volume contraction during mixing ( ideal solution ).
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer ttnodes,itime
      real*8 c01,c02,V1,V2,MM1,MM2,
     +       act1,act2,XChib(3),Tbb(ttnodes),
     +       X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3,X23_0,
     +       X23_1,X23_2,X23_3
c
c-----------------------------------------------------------------------
c.....Local variables
c-----------------------------------------------------------------------
      real*8 Phi1,Phi2,Phip,MV1,MV2
c
c-----------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c-----------------------------------------------------------------------
      act1=0
      act2=0
c
      Phi1=c01*V1
      Phi2=c02*V2
      Phip=1-Phi1-Phi2
c
      MV1=MM1*V1
      MV2=MM2*V2
c-----------------------------------------------------------------------
c.....Determination of interaction coeficients at guessed temperature
c.....XChib at Tbb
c-----------------------------------------------------------------------
c
      call Chib(Tbb,ttnodes,itime,X12_0,X12_1,X12_2,X12_3,X13_0,
     +         X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +         XChib)
c-----------------------------------------------------------------------

      if (Phi1.eq.0) then
c
      act1=0
c
      else
c
      act1=exp(log(Phi1)+(1-Phi1)-(MV1/MV2)*Phi2+((XChib(3)*Phi2+
     +            XChib(1)*Phip)*(Phi2+Phip))-
     +        XChib(2)*(MV1/MV2)*Phi2*Phip)
c
      endif
c
      if (Phi2.eq.0) then
c
      act2=0
c
      else
c
      act2=exp(log(Phi2)+(1-Phi2)-(MV2/MV1)*Phi1+((XChib(3)*Phi1+
     +      XChib(2)*Phip)*(Phi1+Phip))-
     +        XChib(1)*(MV2/MV1)*Phi1*Phip)
c
      endif
c
      end subroutine
```

```
c*******************************************************************
c
      subroutine IniActivity(c01,c02,V1,V2,MM1,MM2,X13,X23,X12,
     +                    iniact1,iniact2)
c
c*******************************************************************
c.....Flory-Huggins is used to determinate the activity of solvents.
c
c.....Assumptions:
c......1-Molar volume of polymer >> Molar volume of solvents;
c......2-Constant interaction parameters;
c......3-No volume contraction during mixing ( ideal solution ).
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      real*8 c01,c02,V1,V2,MM1,MM2,X12,X13,X23,
     +       iniact1,iniact2
c
c-------------------------------------------------------------------
c.....Local variables
c-------------------------------------------------------------------
      real*8 Phi1,Phi2,Phip,MV1,MV2
c
c-------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c-------------------------------------------------------------------
      iniact1=0
      iniact2=0
c
      Phi1=c01*V1
      Phi2=c02*V2
      Phip=1-Phi1-Phi2
c
      MV1=MM1*V1
      MV2=MM2*V2
c
      if (Phi1.eq.0) then
c
      iniact1=0
c
      else
c
      iniact1=exp(log(Phi1)+(1-Phi1)-(MV1/MV2)*Phi2+
     +        ((X12*Phi2+X13*Phip)*(Phi2+Phip))-X23*
     +        (MV1/MV2)*Phi2*Phip)
c
      endif
c
      if (Phi2.eq.0) then
c
      iniact2=0
c
      else
c
      iniact2=exp(log(Phi2)+(1-Phi2)-(MV2/MV1)*Phi1+
     +        ((X12*(MV2/MV1)*Phi1+X23*Phip)*(Phi1+Phip))-X13*
     +        (MV2/MV1)*Phi1*Phip)
```

```
c
      endif
c
      end subroutine









c**********************************************************************
c
      subroutine FormRtbb(itime,ttnodes,Tbb,A1,B1,C1,D1,E1,A2,B2,C2,D2,
     +              E2,act1,act2,
     +                Rtbb,Rtbbmin,Tbbmin,VP1bb,VP2bb)
c
c**********************************************************************
c**********************************************************************
c
c.....Residue of equation to get bubble temperature of solution.
c
c**********************************************************************
c----------------------------------------------------------------------
c.....External & returning variables
c----------------------------------------------------------------------
      integer itime,ttnodes
      real*8 A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,act1,act2,
     +       Tbb(ttnodes),Rtbb,Tbbmin(2),Rtbbmin(2),VP1bb,VP2bb
c
c----------------------------------------------------------------------
c.....Local variables
c----------------------------------------------------------------------
      real*8 VP1min,VP2min
c
c----------------------------------------------------------------------
c.....Tbb      : Solution bubble point temperature
[K]
c.....VPibb    : Vapor pressure that causes partial pressure of solvent
c               i reach atmospheric pressure (760mmHg )        [mmHg]
c----------------------------------------------------------------------
c
      Rtbb=0
      Rtbbmin=0
c
c----------------------------------------------------------------------
c.....Residues to determine bubble temperature of each pure solvent
c.....( boil temperature )
c----------------------------------------------------------------------
c
      if (itime.eq.1) then
c
      VP1min=760
      VP2min=760
c
      Rtbbmin(1)=log10(VP1min)-A1-B1/Tbbmin(1)-C1*log10(Tbbmin(1))-
     +      D1*Tbbmin(1)-E1*(Tbbmin(1)**2)
```

```
c
      Rtbbmin(2)=log10(VP2min)-A2-B2/Tbbmin(2)-C2*log10(Tbbmin(2))-
     +          D2*Tbbmin(2)-E2*(Tbbmin(2)**2)
c
      endif
c
c-----------------------------------------------------------------------
c.....Residue to determine bubble temperature of solution
c-----------------------------------------------------------------------
c
      VP1bb=10**(A1+B1/Tbb(itime)+C1*
     +           log10(Tbb(itime))+D1*Tbb(itime)+
     +        E1*(Tbb(itime)**2))
c
      VP2bb=10**(A2+B2/Tbb(itime)+C2*
     +           log10(Tbb(itime))+D2*Tbb(itime)+
     +        E2*(Tbb(itime)**2))
c
      Rtbb=760-act1*VP1bb-act2*VP2bb
c
      end subroutine







c**********************************************************************
c


      subroutine FormJTbb (nnodes,ttnodes,itime,X,V1,V2,MM1,MM2,A1,B1,
     +                 C1,D1,E1,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                 X12_1,X12_2,X12_3,A2,B2,C2,D2,E2,VP1bb,VP2bb,
     +                 act1,act2,XChib,DifModel,Tbb,
     +                 Jtbb,Tbbmin,Jtbbmin)
c**********************************************************************
c**********************************************************************
c
c.....Jacobian of each equation to determinate solution bubble
c......temperature and boiling temperature of each pure solvent.
c
c**********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer itime,ttnodes,nnodes,DifModel
      real*8 X,V1,V2,MM1,MM2,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +      X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,X12_1,
     +      X12_2,X12_3,VP1bb,VP2bb,act1,act2,
     +      JTbb,Tbb(ttnodes),Tbbmin(2),Jtbbmin(2),dPdT,XChib(3)
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
c.....Jacobian to get boil temperature of pure solvents
c-----------------------------------------------------------------------
      if (itime.eq.1) then
c
      Jtbbmin(1)=B1/(Tbbmin(1)**2)-C1/(Tbbmin(1)*log(10.0))-
     +      D1-2*E1*Tbbmin(1)
```

```
c
      Jtbbmin(2)=B2/(Tbbmin(2)**2)-C2/(Tbbmin(2)*log(10.0))-
     +      D2-2*E2*Tbbmin(2)
c
      endif
c
c-----------------------------------------------------------------------
c.....Jacobian to get solution bubble temperature
c-----------------------------------------------------------------------
c
      call FormdPdT(nnodes,ttnodes,itime,X,V1,V2,MM1,MM2,A1,B1,
     +                C1,D1,E1,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                X12_1,X12_2,X12_3,A2,B2,C2,D2,E2,VP1bb,VP2bb,
     +                act1,act2,XChib,DifModel,Tbb,
     +                dPdT)
c
      Jtbb=-dPdT
c
      endsubroutine









c***********************************************************************
c
      subroutine Chib(Tbb,ttnodes,itime,X12_0,X12_1,X12_2,X12_3,X13_0,
     +           X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +           XChib)
c
c***********************************************************************
c
c.... Chib calculates the interaction factors X12,X13,X23 at guessed
c.....temperature
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer itime,ttnodes
      real*8 X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,
     +       X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +       Tbb(ttnodes),XChib(3)
c
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c-----------------------------------------------------------------------
c
      XChib(1)=X13_0+X13_1/Tbb(itime)+X13_2*Phi1+X13_3*Phi2
      XChib(2)=X23_0+X23_1/Tbb(itime)+X23_2*Phi1+X23_3*Phi2
      XChib(3)=X12_0+X12_1/Tbb(itime)+X12_2*Phi1+X12_3*Phi2
c
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine  FormdPdT(nnodes,ttnodes,itime,X,V1,V2,MM1,MM2,A1,B1,
     +                 C1,D1,E1,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                 X12_1,X12_2,X12_3,A2,B2,C2,D2,E2,VP1bb,VP2bb,
     +                 act1,act2,XChib,DifModel,Tbb,
     +                 dPdT)
c
c*********************************************************************
c*********************************************************************
c
c...Derivative of solution pressure to determine solution bubble
temperature.
c
c   J= - (act1*dVP1/dT + dact1/dT * VP1 + act2*dVP2/dT + dact2/dT * VP2
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,DifModel,ttnodes,itime
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,A1,B1,C1,D1,E1,
     +       A2,B2,C2,D2,E2,VP1bb,VP2bb,act1,act2,dactdT(2),dPvbdT(2),
     +       X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,
     +       XChib(3),dPdT,Tbb(ttnodes)
c---------------------------------------------------------------------
c.....FormdactdT calculates derivatives of activity of each solvent
c......at base.
c---------------------------------------------------------------------
      call FormdactdT (DifModel,nnodes,ttnodes,itime,X,V1,V2,MM1,
     +                MM2,XChib,Tbb,X13_1,X13_2,X13_3,
     +                X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,
     +                dactdT)
c
c---------------------------------------------------------------------
c.....FormdPvbdT calculates devivatives of vapor pressure of each solvent
c---------------------------------------------------------------------
      call FormdPvbdT(itime,ttnodes,Tbb,A1,B1,C1,D1,E1,A2,B2,C2,
     +               D2,E2,
     +                 dPvbdT)
c
c---------------------------------------------------------------------
c
      dPdT=act1*dPvbdT(1)+dactdT(1)*VP1bb+act2*dPvbdT(2)+dactdT(2)*VP2bb
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine FormdactdT (DifModel,nnodes,ttnodes,itime,X,V1,V2,MM1,
     +                    MM2,XChib,Tbb,X13_1,X13_2,X13_3,
     +                    X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,
     +                    dactdT)
c
c*********************************************************************
c*********************************************************************
c
c.....Derivatives of activity of each solvent at base.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,DifModel,ttnodes,itime
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X13_1,X13_2,X13_3,
     +       X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,
     +       XChib(3),dactdT(2),Tbb(ttnodes)

c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 dXChibdX(9),MV1,MV2,Phi1,Phi2,Phip

c---------------------------------------------------------------------
c     Phii = initial volume fraction of component i [cm3/cm3]
c     MVi = molar volume of component i   [cm3/mol]
c---------------------------------------------------------------------
c
      dactdT=0
c
      Phi1=X(1,2)*V1
      Phi2=X(nnodes+1,2)*V2
      Phip=1-Phi1-Phi2
c
      MV1=MM1*V1
      MV2=MM2*V2
c
      if (DifModel.eq.6) then
c
c---------------------------------------------------------------------
c.....Derivative of activity to temperature at base
c---------------------------------------------------------------------
c
      call FormdXChibdX (ttnodes,itime,V1,V2,X13_1,X13_2,X13_3,
     +                  X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,Tbb,
     +                  dXChibdX)
c---------------------------------------------------------------------
      dactdT(1)=Phi1*exp((1-Phi1)-(MV1/MV2)*Phi2+(XChib(3)*
     +     Phi2+XChib(1)*Phip)*(Phi2+Phip)-XChib(2)*
     +     (MV1/MV2)*Phi2*Phip)*((Phi2+Phip)*(Phi2*
     +     dXChibdX(9)+Phip*dXChibdX(3))-(MV1/MV2)*
     +     Phi2*Phip*dXChibdX(6))
c
      dactdT(2)=Phi2*exp((1-Phi2)-(MV2/MV1)*Phi1+(XChib(3)*
     +     Phi1*(MV2/MV1)+XChib(2)*Phip)*(Phi1+Phip)-
     +     XChib(1)*(MV2/MV1)*Phi1*Phip)*((Phi1+Phip)*(Phi1*
     +     (MV2/MV1)*dXChibdX(9)+Phip*dXChibdX(6))-
     +     (MV2/MV1)*Phi1*Phip*dXChibdX(3))
c
      else
c
      dactdT(1)= 0
c
      dactdT(2)= 0
c
      endif
c
      endsubroutine








c*********************************************************************
c
      subroutine FormdPvbdT(itime,ttnodes,Tbb,A1,B1,C1,D1,E1,A2,B2,C2,
     +                      D2,E2,
     +                        dPvbdT)
c
c*********************************************************************
c*********************************************************************
c
c.....Vapor Pressure derivatives of Solvents at guessed temperature.
c
c     VP=10**( A+ B/T + ClogT + DT + ET2 )         [mmHg]
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer itime,ttnodes
      real*8 A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +       dPvbdT(2),Tbb(ttnodes)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 T

c---------------------------------------------------------------------
c.....T  : Guessed temperature            [K]
c---------------------------------------------------------------------
      T=Tbb(itime)
c---------------------------------------------------------------------
c
      dPvbdT(1)=log(10.0)*(-B1/(T**2)+C1/
     +    (T*log(10.0))+
     +    D1+2*E1*T)*(10**(A1+B1/T+C1*
     +      log10(T)+D1*T+
     +    E1*(T**2)))
     +
c
      dPvbdT(2)=log(10.0)*(-B2/(T**2)+C2/
     +    (T*log(10.0))+D2+2*E2*T)*
     +    (10**(A2+B2/T+C2*
     +      log10(T)+D2*T+
     +    E2*(T**2)))
c
       endsubroutine




c*******************************************************************
c
      subroutine FormdXChibdX (ttnodes,itime,V1,V2,X13_1,X13_2,X13_3,
     +                        X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,Tbb,
     +                        dXChibdX)
c
c*******************************************************************
c*******************************************************************
c
c.....Derivatives of interaction factors
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer itime,ttnodes
      real*8 V1,V2,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +       X12_1,X12_2,X12_3,
     +       Tbb(ttnodes),dXChibdX(9)
c
c-------------------------------------------------------------------
c
      dXChibdX(1)=X13_2*V1
      dXChibdX(2)=X13_3*V2
      dXChibdX(3)=-X13_1/(Tbb(itime)**2)
c
      dXChibdX(4)=X23_2*V1
      dXChibdX(5)=X23_3*V2
      dXChibdX(6)=-X23_1/(Tbb(itime)**2)
c
      dXChibdX(7)=X12_2*V1
      dXChibdX(8)=X12_3*V2
      dXChibdX(9)=-X12_1/(Tbb(itime)**2)
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine ZoneSettgs(itime,hs1,hs2,hs3,hs4,hs5,hs6,hi1,hi2,hi3,
     +                     hi4,hi5,hi6,Ta1s,Ta2s,Ta3s,Ta4s,Ta5s,Ta6s,
     +                     Ta1i,Ta2i,Ta3i,Ta4i,Ta5i,Ta6i,P11,
     +                     P21,P12,P22,P13,P23,P14,P24,P15,P25,P16,
     +                     P26,tnodes,
     +                     hs,hi,Tas,Tai,P1a,P2a)
c
c*********************************************************************
c*********************************************************************
c
c.....Sets the current drying process parameters
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer itime,tnodes
      real*8 hs1,hs2,hs3,hs4,hs5,hs6,hi1,hi2,hi3,hi4,hi5,hi6,Ta1s,Ta2s,
     +       Ta3s,Ta4s,Ta5s,Ta6s,Ta1i,Ta2i,Ta3i,Ta4i,Ta5i,Ta6i,
     +       P11,P21,P12,P22,P13,P23,P14,P24,P15,P25,
     +       P16,P26,hs,hi,Tas,Tai,P1a,P2a
c
c---------------------------------------------------------------------
c     conversion factor Bar to g/cm.s2
c---------------------------------------------------------------------
      cf=10E06
c---------------------------------------------------------------------
      if (itime<tnodes+1) THEN
             hs=hs1
               hi=hi1
              Tas=Ta1s
            Tai=Ta1i
            P1a=P11*cf
            P2a=P21*cf
c
      else if(itime>tnodes.and.itime<2*tnodes+1) THEN
             hs=hs2
               hi=hi2
              Tas=Ta2s
            Tai=Ta2i
            P1a=P12*cf
            P2a=P22*cf
c
      else if(itime>2*tnodes.and.itime<3*tnodes+1) THEN
               hs=hs3
             hi=hi3
              Tas=Ta3s
            Tai=Ta3i
            P1a=P13*cf
            P2a=P23*cf
c
      else if(itime>3*tnodes.and.itime<4*tnodes+1)THEN
               hs=hs4
               hi=hi4
              Tas=Ta4s
            Tai=Ta4i
            P1a=P14*cf
            P2a=P24*cf
c
      else if(itime>4*tnodes.and.itime<5*tnodes+1)THEN
               hs=hs5
               hi=hi5
              Tas=Ta5s
            Tai=Ta5i
            P1a=P15*cf
            P2a=P25*cf
c
      else if(itime>5*tnodes) THEN
             hs=hs6
               hi=hi6
              Tas=Ta6s
            Tai=Ta6i
            P1a=P16*cf
            P2a=P26*cf
c
      end if
c
      end subroutine






c*********************************************************************
c
      subroutine guess (itime,nnodes,ttnodes,C,tm,
     +                  X)
c
c*********************************************************************
c*********************************************************************
c
c.....Guessing the solution for current time step
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,itime,tnodes,ttnodes
      real*8 X(3*nnodes+1,2),C(3*nnodes+1,ttnodes)
c
c---------------------------------------------------------------------

c
      do inode=1,3*nnodes+1
c
      X(inode,1)=C(inode,itime-1)
c
      X(inode,2)=X(inode,1)
c
      enddo
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine FormR(nnodes,itime,X,tm,hs,hi,Tas,Tai,P1a,P2a,V1,V2,
     +                  V3,MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,X12,
     +                HSub,RoSub,cpSub,RoAir,cpAir,A1,B1,C1,D1,E1,A2,
     +                B2,C2,D2,E2,a11,a12,a13,D01,D02,b21,b22,b23,
     +                V1crt,V2crt,V3crt,eps13,eps23,DifModel,a,tnodes,
     +                X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3,
     +                X23_0,X23_1,X23_2,X23_3,XChi,dXChidX,Alpha,An,
     +                A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,Eact1,Eact2,
     +                  R,P1,P2,VP1,VP2,act1,act2,Dm,Di,dmidro,K1,K2)

c
c*********************************************************************
c*********************************************************************
c
c.....Residue of each equation.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,itime,DifModel,tnodes
      real*8 X(3*nnodes+1,2),tm(6*tnodes+1),hs,hi,Tas,Tai,P1a,P2a,V1,
     +    V2,V3,MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,X12,HSub,RoSub,
     +    cpSub,RoAir,cpAir,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,a11,a12,
     +    a13,D01,D02,b21,b22,b23,V1crt,V2crt,V3crt,eps13,eps23,a,
     +    X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,X13_2,X13_3,X23_0,
     +    A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,Eact1,Eact2,
     +    X23_1,X23_2,X23_3,P1,P2,VP1,VP2,act1,act2,K1,K2,An,
     +    D(nnodes,4),Dm(nnodes-1,4),XChi(nnodes,3),dXChidX(nnodes,9),
     +    R(3*nnodes+1),Di(nnodes,2),dmidro(nnodes,6),Alpha(nnodes,2)

c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 aux1,aux2
c
c---------------------------------------------------------------------
c.....PartialPressure calculates the partial pressure of each sonvent
c......at interface.
c---------------------------------------------------------------------
      call PartialPressure(X,nnodes,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +                       V1,V2,MM1,MM2,X13,X23,X12,DifModel,
     +                        X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,
     +                        X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                        P1,P2,VP1,VP2,act1,act2,XChi)

      if (P1a.gt.P1) then
c
      P1a=P1
c
      endif
c
      if(P2a.gt.P2) then
c
      P2a=P2
c
```

```
      endif
c
c-----------------------------------------------------------------------
c.....MassTransfCoef calculates the mass transfer coef for each solvent
c-----------------------------------------------------------------------

      call MassTransfCoef  (X,nnodes,Tas,hs,MM1,RoAir,CpAir,MM2,
     +                     K1,K2)
c
c-----------------------------------------------------------------------
c.....MutDiffCoef calculates the mutual diffusion coeficients.
c-----------------------------------------------------------------------
      call MutDiffCoef(X,nnodes,a11,b21,a12,b22,a13,b23,D01,D02,
     +                DifModel,V1crt,V2crt,V3crt,eps13,eps23,V1,V2,V3,
     +                MM1,MM2,X12,X13,X23,X13_1,X13_2,X13_3,X23_1,
     +                X23_2,X23_3,X12_1,X12_2,X12_3,An,Eact1,Eact2,
     +                A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,
     +                D,Di,dmidro,XChi,dXChidX,Alpha)
c
c-----------------------------------------------------------------------
c.....MeanDiffCoef calculates averages to mutual diffusion coeficients.
c-----------------------------------------------------------------------
      call MeanDiffCoef (nnodes,D,
     +                   Dm)
c
c-----------------------------------------------------------------------
      R(1)=(X(2,2)-X(1,2))
c
      R(nnodes+1)=(X(nnodes+2,2)-X(nnodes+1,2))
c
      R(2*nnodes+1)=X(2*nnodes+1,2)
c
      do inode=2,nnodes-1
c
c.....Mass conservation equations
c
c
      R(inode)=(X(inode,2)-X(inode,1))/(tm(itime)-tm(itime-1))-
     +        (X(2*nnodes+inode,2)/X(3*nnodes,2))*((X(3*nnodes,2)-
     +        X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*((X(inode+1,2)-
     +        X(inode-1,2))/(X(2*nnodes+1+inode,2)-
     +        X(2*nnodes-1+inode,2)))-(2/(X(2*nnodes+1+inode,2)-
     +        X(2*nnodes-1+inode,2)))*
     +        (Dm(inode,1)*(X(inode+1,2)-X(inode,2))/
     +        (X(2*nnodes+1+inode,2)-X(2*nnodes+inode,2))-
     +        Dm(inode-1,1)*(X(inode,2)-X(inode-1,2))/
     +        (X(2*nnodes+inode,2)-X(2*nnodes-1+inode,2))+Dm(inode,2)*
     +        (X(nnodes+inode+1,2)-X(nnodes+inode,2))/
     +        (X(2*nnodes+1+inode,2)-X(2*nnodes+inode,2))-
     +        Dm(inode-1,2)*(X(nnodes+inode,2)-X(nnodes+inode-1,2))/
     +        (X(2*nnodes+inode,2)-X(2*nnodes-1+inode,2)))
c
      R(nnodes+inode)=(X(nnodes+inode,2)-X(nnodes+inode,1))/
     +        (tm(itime)-tm(itime-1))-(X(2*nnodes+inode,2)/
     +        X(3*nnodes,2))*((X(3*nnodes,2)-X(3*nnodes,1))/
     +        (tm(itime)-tm(itime-1)))*((X(nnodes+inode+1,2)-
     +        X(nnodes+inode-1,2))/(X(2*nnodes+1+inode,2)-
     +        X(2*nnodes-1+inode,2)))-(2/(X(2*nnodes+1+inode,2)-
     +        X(2*nnodes-1+inode,2)))*
     +        (Dm(inode,3)*(X(inode+1,2)-X(inode,2))/
     +        (X(2*nnodes+1+inode,2)-X(2*nnodes+inode,2))-
     +        Dm(inode-1,3)*(X(inode,2)-X(inode-1,2))/
     +        (X(2*nnodes+inode,2)-X(2*nnodes-1+inode,2))+
     +        Dm(inode,4)*(X(nnodes+inode+1,2)-X(nnodes+inode,2))/
     +        (X(2*nnodes+1+inode,2)-X(2*nnodes+inode,2))-
     +        Dm(inode-1,4)*(X(nnodes+inode,2)-X(nnodes+inode-1,2))/
     +        (X(2*nnodes+inode,2)-X(2*nnodes-1+inode,2)))
c
      aux1=nnodes-inode
      aux2=nnodes-1
c
c.....Mesh equation
c
      R(2*nnodes+inode)=X(2*nnodes+inode,2)-X(3*nnodes,2)*(1-
     +                    (aux1/aux2)**a)
c
      end do
c
c.....Interface
c
      R(nnodes)=-Dm(nnodes-1,1)*(X(nnodes,2)-X(nnodes-1,2))/
     +          (X(3*nnodes,2)-X(3*nnodes-1,2))-Dm(nnodes-1,2)*
     +          (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-
     +          X(3*nnodes-1,2))-X(nnodes,2)*(X(3*nnodes,2)-
     +          X(3*nnodes,1))/(tm(itime)-tm(itime-1))-K1*(P1-P1a)
c
      R(2*nnodes)=-Dm(nnodes-1,3)*(X(nnodes,2)-X(nnodes-1,2))/
     +            (X(3*nnodes,2)-X(3*nnodes-1,2))-Dm(nnodes-1,4)*
     +            (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-
     +            X(3*nnodes-1,2))-X(2*nnodes,2)*(X(3*nnodes,2)-
     +                  X(3*nnodes,1))/(tm(itime)-tm(itime-1))-K2*(P2-
P2a)
c
      R(3*nnodes)=(X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1))+
     +            K1*V1*(P1-P1a)+K2*V2*(P2-P2a)
c
      R(3*nnodes+1)=(X(3*nnodes+1,2)-X(3*nnodes+1,1))/(tm(itime)-
     +              tm(itime-1))+(1/(RoBar*cpLBar*X(3*nnodes,2)+
     +              RoSub*cpSub*HSub))*(hs*(X(3*nnodes+1,2)-Tas)+K1*
     +              DelH1*(P1-P1a)+K2*DelH2*(P2-P2a)+hi*
     +              (X(3*nnodes+1,2)-Tai))
c
      end subroutine
```

```
c*********************************************************************
c
      subroutine PartialPressure(X,nnodes,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +                          V1,V2,MM1,MM2,X13,X23,X12,DifModel,
     +                          X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,
     +                          X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +                          P1,P2,VP1,VP2,act1,act2,XChi)
c
c*********************************************************************
c*********************************************************************
c
c.....Partial pressure of each solvent at interface.
c
c.....P=act.VP
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,V1,V2,
     +       MM1,MM2,X13,X23,X12,X12_0,X12_1,X12_2,X12_3,X13_0,
     +       X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +       P1,P2,VP1,VP2,act1,act2,XChi(nnodes,3)
c
c---------------------------------------------------------------------
c.....Vapor pressure for each solvent
c---------------------------------------------------------------------
      call VaporPressure(X,nnodes,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +                   VP1,VP2)
c
      if (DifModel.eq.6) then
c---------------------------------------------------------------------
c.....Activity of solvents at interface (Variable interaction factor)
c---------------------------------------------------------------------
      call ActivityVarX(X,nnodes,V1,V2,MM1,MM2,X12_0,X12_1,X12_2,
     +                        X12_3,X13_0,X13_1,X13_2,X13_3,X23_0,
     +                    X23_1,X23_2,X23_3,
     +                    act1,act2,XChi)
c---------------------------------------------------------------------
      else
c---------------------------------------------------------------------
c.....Activity of solvents at interface (Constant interaction factor)
c---------------------------------------------------------------------
      call Activity(X,nnodes,V1,V2,MM1,MM2,X13,X23,X12,
     +             act1,act2)
c
      endif
c
      P1=act1*VP1
      P2=act2*VP2
c
      endsubroutine
```

```
c*******************************************************************
c
      subroutine VaporPressure(X,nnodes,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +                         VP1,VP2)
c
c*******************************************************************
c*******************************************************************
c
c.....Calculates the vapor pressure of each solvent
c
c     VP(T)=10**(A-(B/T)+C*logT+D*T+E*T²))
[g/cm.s2]
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +       VP1,VP2
c
c-------------------------------------------------------------------
c.....Conversion factor mmHg to g/cm.s2
c-------------------------------------------------------------------
      cf=1333.0
c-------------------------------------------------------------------
c.....Solvent 1

            VP1=cf*10**(A1+B1/X(3*nnodes+1,2)+C1*
     +            log10(X(3*nnodes+1,2))+D1*X(3*nnodes+1,2)+
     +        E1*(X(3*nnodes+1,2)**2))

c.....Solvent 2

            VP2=cf*10**(A2+B2/X(3*nnodes+1,2)+C2*
     +            log10(X(3*nnodes+1,2))+D2*X(3*nnodes+1,2)+
     +        E2*(X(3*nnodes+1,2)**2))
c
      endsubroutine
```

```
c******************************************************************
c
      subroutine ActivityVarX(X,nnodes,V1,V2,MM1,MM2,X12_0,X12_1,X12_2,
     +                        X12_3,X13_0,X13_1,X13_2,X13_3,X23_0,
     +                        X23_1,X23_2,X23_3,
     +                        act1,act2,XChi)
c
c******************************************************************
c.....Flory-Huggins is used to determinate the activity of solvents.
c
c.....Assumptions:
c......1-Molar volume of polymer >> Molar volume of solvents;
c......2-Variable interaction parameters;
c......3-No volume contraction during mixing ( ideal solution ).
c
c******************************************************************
c------------------------------------------------------------------
c.....External & returning variables
c------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X12_0,X12_1,X12_2,X12_3,
     +       X13_0,X13_1,X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,
     +       act1,act2,XChi(nnodes,3)
c
c------------------------------------------------------------------
c.....Local variables
c------------------------------------------------------------------
      real*8 Phi1,Phi2,Phip,MV1,MV2
c
c------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c------------------------------------------------------------------
      act1=0
      act2=0
c
      Phi1=X(nnodes,2)*V1
      Phi2=X(2*nnodes,2)*V2
      Phip=1-Phi1-Phi2
c
      MV1=MM1*V1
      MV2=MM2*V2
c
c------------------------------------------------------------------
c.....Chi calculates interaction factors X12,X13,X23
c------------------------------------------------------------------
      call Chi(X,nnodes,X12_0,X12_1,X12_2,X12_3,X13_0,X13_1,
     +        X13_2,X13_3,X23_0,X23_1,X23_2,X23_3,Phi1,Phi2,
     +        XChi)
c------------------------------------------------------------------
c
      if (Phi1.eq.0) then
c
      act1=0
c
      else
c
      act1=exp(log(Phi1)+(1-Phi1)-(MV1/MV2)*Phi2+((XChi(nnodes,3)*Phi2+
     +            XChi(nnodes,1)*Phip)*(Phi2+Phip))-
     +        XChi(nnodes,2)*(MV1/MV2)*Phi2*Phip)
c
      endif
c
      if (Phi2.eq.0) then
c
      act2=0
c
      else
c
      act2=exp(log(Phi2)+(1-Phi2)-(MV2/MV1)*Phi1+((XChi(nnodes,3)*Phi1+
     +      XChi(nnodes,2)*Phip)*(Phi1+Phip))-
     +        XChi(nnodes,1)*(MV2/MV1)*Phi1*Phip)
c
      endif
c
      end subroutine




c***********************************************************************
c
      subroutine Activity(X,nnodes,V1,V2,MM1,MM2,X13,X23,X12,
     +                    act1,act2)
c
c***********************************************************************
c.....Flory-Huggins is used to determinate the activity of solvents.
c
c.....Assumptions:
c......1-Molar volume of polymer >> Molar volume of solvents;
c......2-Constant interaction parameters;
c......3-No volume contraction during mixing ( ideal solution ).
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X12,X13,X23,
     +       act1,act2
c
c-----------------------------------------------------------------------
c.....Local variables
c-----------------------------------------------------------------------
      real*8 Phi1,Phi2,Phip,MV1,MV2
c
c-----------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c-----------------------------------------------------------------------
      act1=0
      act2=0
c
      Phi1=X(nnodes,2)*V1
      Phi2=X(2*nnodes,2)*V2
      Phip=1-Phi1-Phi2
c
      MV1=MM1*V1
      MV2=MM2*V2
c
      if (Phi1.eq.0) then
c
      act1=0
c
      else
c
      act1=exp(log(Phi1)+(1-Phi1)-(MV1/MV2)*Phi2+((X12*Phi2+X13*Phip)*
     +   (Phi2+Phip))-X23*(MV1/MV2)*Phi2*Phip)
c
      endif
c
      if (Phi2.eq.0) then
c
      act2=0
c
      else
c
      act2=exp(log(Phi2)+(1-Phi2)-(MV2/MV1)*Phi1+((X12*(MV2/MV1)*Phi1+
     +      X23*Phip)*(Phi1+Phip))-X13*(MV2/MV1)*Phi1*Phip)
c
      endif
c
      end subroutine




c***********************************************************************
c
      subroutine MassTransfCoef(X,nnodes,Ta,hs,MM1,RoAir,CpAir,MM2,
     +                          K1,K2)
c
c***********************************************************************
c***********************************************************************
c
c.....Calculates mass transfer coeficients using Chilton-Colburn analo-
c......gy to heat transfer.    [s/cm]
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),Ta,hs,MM1,RoAir,CpAir,MM2,
     +       K1,K2
c
c-----------------------------------------------------------------------
c.....Local variables
c-----------------------------------------------------------------------
      real*8 Tmd
c
c-----------------------------------------------------------------------
c.....Tmd:Average temperature between liquid and air  [K]
c-----------------------------------------------------------------------
c
c-----------------------------------------------------------------------
```

```
c.....Constants
c-----------------------------------------------------------------------
c.....R: Universal gas constant                      [(g/cm.s2).cm3/mol.K]
c
      R=8.31451E07
c
c.....a: Thermal conductivity of air                 [W/cm.K]
c
      a=0.00026
c-----------------------------------------------------------------------
c
      DAir1= 0.086
      DAir2= 0.086
c
      Tmd=(X(3*nnodes+1,2)+Ta)/2
c
      if (X(nnodes,1).gt.0) then
c
      K1= hs*MM1*((RoAir*CpAir*DAir1/a)**(0.67))/(RoAir*CpAir*R*Tmd)
c
      else
c
      K1=0
c
      endif
c
      if (X(2*nnodes,1).gt.0) then
c
      K2= hs*MM2*((RoAir*CpAir*DAir2/a)**(0.67))/(RoAir*CpAir*R*Tmd)
c
      else
c
      K2=0
c
      endif
c
      endsubroutine
```

```
c*******************************************************************
c
      subroutine MutDiffCoef(X,nnodes,a11,b21,a12,b22,a13,b23,D01,D02,
     +                 DifModel,V1crt,V2crt,V3crt,eps13,eps23,V1,V2,V3,
     +                 MM1,MM2,X12,X13,X23,X13_1,X13_2,X13_3,X23_1,
     +                 X23_2,X23_3,X12_1,X12_2,X12_3,An,Eact1,Eact2,
     +                 A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,
     +                 D,Di,dmidro,XChi,dXChidX,Alpha)
c
c*******************************************************************
c*******************************************************************
c
c.....Calculates the mutual diffusion coeficients using 5 different
c......models
c
c......Initial four models by Alsoy,S.,AIChe Jornal,Vol.45 n°4 (1999)
c......Fifth model by Zielinski.J., AIChe Jornal,Vol.45 n°1 (1999)
c......Sixth model by Price.P.and Romdhane,I.,AIChe Jornal,Vol.49 n°2
(2003)
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),a11,b21,a12,b22,a13,b23,D01,D02,V1crt,
     +       V2crt,V3crt,eps13,eps23,V1,V2,V3,MM1,MM2,X12,X13,X23,
     +       X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,X12_1,X12_2,X12_3,
     +       A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,Eact1,Eact2,
     +       Di(nnodes,2),dmidro(nnodes,6),dXChidX(nnodes,9),
     +       XChi(nnodes,3),D(nnodes,4),Alpha(nnodes,2),An
c
c-------------------------------------------------------------------
c.....Self diffusion coeficients
c-------------------------------------------------------------------
      call SelfDiffCoef(X,nnodes,a11,b21,a12,b22,a13,b23,D01,D02,
     +               V1crt,V2crt,V3crt,eps13,eps23,V1,V2,V3,Eact1,Eact2,
     +               Di)
c
c-------------------------------------------------------------------
c
      if (DifModel.eq.6) then
c
c-------------------------------------------------------------------
c.....Chemical potential gradients for variable interaction factors
c-------------------------------------------------------------------
      call ChemPotVarX(X,nnodes,V1,V2,MM1,MM2,dXChidX,XChi,
     +                 X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                 X12_1,X12_2,X12_3,
     +                 dmidro)
c
      else
c
c-------------------------------------------------------------------
c.....Chemical potential gradients for constant interaction factors
c-------------------------------------------------------------------
      call ChemPot(X,nnodes,V1,V2,MM1,MM2,X12,X13,X23,
     +            dmidro)
c -----------------------------------------------------------------
```

```
c
      endif
c
c-----------------------------------------------------------------------
c.....Mutual diffusion coeficients
c-----------------------------------------------------------------------
      if (DifModel.eq.1) THEN
c
      do inode=1,nnodes
c
      D(inode,1)=Di(inode,1)*X(inode,2)*dmidro(inode,1)
c
      D(inode,2)=X(inode,2)*Di(inode,1)*dmidro(inode,2)
c
      D(inode,3)=X(nnodes+inode,2)*Di(inode,2)*dmidro(inode,3)
c
      D(inode,4)=Di(inode,2)*X(nnodes+inode,2)*dmidro(inode,4)
c
      end do
c
      else if(DifModel.eq.2) THEN
c
      do inode=1,nnodes
c
      D(inode,1)=Di(inode,1)*X(inode,2)*dmidro(inode,1)
c
      D(inode,2)=0
c
      D(inode,3)=0
c
      D(inode,4)=Di(inode,2)*X(nnodes+inode,2)*dmidro(inode,4)
c
      end do
c
      else if(DifModel.eq.3) THEN
c
      do inode=1,nnodes
c
      D(inode,1)=Di(inode,1)
c
      D(inode,2)=0
c
      D(inode,3)=0
c
      D(inode,4)=Di(inode,2)

      end do
c
      else if(DifModel.eq.4) THEN
c
      do inode=1,nnodes
c
      D(inode,1)=X(inode,2)*(1-X(inode,2)*V1)*Di(inode,1)*
     +          dmidro(inode,1)-X(inode,2)*X(nnodes+inode,2)*V2*
     +           Di(inode,2)*dmidro(inode,3)
c
      D(inode,2)=X(inode,2)*(1-X(inode,2)*V1)*Di(inode,1)*
     +           dmidro(inode,2)-X(inode,2)*X(nnodes+inode,2)*V2*
     +           Di(inode,2)*dmidro(inode,4)
```

```
c
      D(inode,3)=X(nnodes+inode,2)*(1-X(nnodes+inode,2)*V2)*
     +           Di(inode,2)*dmidro(inode,3)-X(inode,2)*
     +           X(nnodes+inode,2)*V1*Di(inode,1)*dmidro(inode,1)
c
      D(inode,4)=X(nnodes+inode,2)*(1-X(nnodes+inode,2)*V2)*
     +           Di(inode,2)*dmidro(inode,4)-X(inode,2)*
     +           X(nnodes+inode,2)*V1*Di(inode,1)*dmidro(inode,2)
c
      enddo
c
      else if(DifModel.eq.5) THEN

c
      do inode=1,nnodes
c
      D(inode,1)=X(inode,2)*(1-X(inode,2)*V1+X(inode,2)*V3)*
     +             Di(inode,1)*dmidro(inode,1)+X(inode,2)*
     +           X(nnodes+inode,2)*(V3-V2)*Di(inode,2)*dmidro(inode,3)
c
      D(inode,2)=X(inode,2)*(1-X(inode,2)*V1+X(inode,2)*V3)*
     +           Di(inode,1)*dmidro(inode,2)+X(inode,2)*
     +           X(nnodes+inode,2)*(V3-V2)*Di(inode,2)*dmidro(inode,4)
c
      D(inode,3)=X(nnodes+inode,2)*(1-X(nnodes+inode,2)*V2+
     +               X(nnodes+inode,2)*V3)*Di(inode,2)*dmidro(inode,3)+
     +           X(inode,2)*X(nnodes+inode,2)*(V3-V1)*Di(inode,1)*
     +           dmidro(inode,1)
c
      D(inode,4)=X(nnodes+inode,2)*(1-X(nnodes+inode,2)*V2+
     +           X(nnodes+inode,2)*V3)*Di(inode,2)*dmidro(inode,4)+
     +           X(inode,2)*X(nnodes+inode,2)*(V3-V1)*Di(inode,1)*
     +           dmidro(inode,2)
c
      enddo
c
      else if(DifModel.eq.6) THEN
c
c-----------------------------------------------------------------------
c.....FormAlpha calculates parameter Alpha of model 6
c-----------------------------------------------------------------------
      call FormAlpha(X,nnodes,A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,V1,V2,
     +              Alpha)
c-----------------------------------------------------------------------
c
      do inode=1,nnodes
c
      D(inode,1)=(1-X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*X(inode,2)*
     +           Di(inode,1)*dmidro(inode,1)-(1-(Alpha(inode,2)/An))*
     +           X(nnodes+inode,2)*V2*X(inode,2)*Di(inode,2)*
     +           dmidro(inode,3)
c
      D(inode,2)=(1-X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*X(inode,2)*
     +           Di(inode,1)*dmidro(inode,2)-(1-(Alpha(inode,2)/An))*
     +           X(nnodes+inode,2)*V2*X(inode,2)*Di(inode,2)*
     +           dmidro(inode,4)
c
      D(inode,3)=(1-X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An)))*
     +           X(nnodes+inode,2)*
     +          Di(inode,2)*dmidro(inode,3)-(1-(Alpha(inode,1)/An))*
     +          X(inode,2)*V1*X(nnodes+inode,2)*Di(inode,1)*
     +          dmidro(inode,1)
c
      D(inode,4)=(1-X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An)))*
     +          X(nnodes+inode,2)*
     +          Di(inode,2)*dmidro(inode,4)-(1-(Alpha(inode,1)/An))*
     +          X(inode,2)*V1*X(nnodes+inode,2)*Di(inode,1)*
     +          dmidro(inode,2)
c
      enddo
c
      else
c
      do inode=1,nnodes
c
      D(inode,1)=1.0E-05
c
      D(inode,2)=0
c
      D(inode,3)=0
c
      D(inode,4)=1.0E-05
c
      enddo
c
      endif
c
      endsubroutine



c************************************************************************
c
      subroutine SelfDiffCoef(X,nnodes,a11,b21,a12,b22,a13,b23,D01,D02,
     +             V1crt,V2crt,V3crt,eps13,eps23,V1,V2,V3,Eact1,Eact2,
     +             Di)
c
c************************************************************************
c************************************************************************
c
c.....Calculates Self Diffusion Coeficients using free volume theory
c......of Vrentas and Duda.
c
c************************************************************************
c------------------------------------------------------------------------
c.....External & returning variables
c------------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),a11,b21,a12,b22,a13,b23,D01,D02,V1crt,
     +       V2crt,V3crt,eps13,eps23,V1,V2,V3,Eact1,Eact2,
     +       Di(nnodes,2)
c
c------------------------------------------------------------------------
c.....Local variables
c------------------------------------------------------------------------
      real*8 ro1,ro2,ro3,w1,w2,w3,T,Vfh_Gama
c
```

```
c----------------------------------------------------------------------
c.....T        : Current liquid temperature            [K]
c.....roi      : Concentration of component i      [g/cm3]
c.....wi       : Mass fraction of component i
c.....Vfh_Gama: Free volume paramenter            [cm3/g]
c----------------------------------------------------------------------
c----------------------------------------------------------------------
c.....Parameters
c----------------------------------------------------------------------
c.....a11=K11/Gama                                          [cm3/g.K]
c.....b21=K21-Tg1                                           [K]
c.....a12=K12/gama                                          [cm3/g.K]
c.... b22=K22-Tg2                                           [K]
c.... a13=K13/Gama                                          [cm3/g.K]
c.... b23=K23-Tg3                                           [K]
c.....Eact1,Eact2                                 [cal/mol]
c----------------------------------------------------------------------
c----------------------------------------------------------------------
c.....Gas Constant                                [cal/mol.K]
c----------------------------------------------------------------------
      R=2.0
c----------------------------------------------------------------------
      do inode=1,nnodes
c
      ro1=X(inode,2)
      ro2=X(nnodes+inode,2)
      ro3=(1-(V1*ro1+V2*ro2))/V3
c
      w1=ro1/(ro1+ro2+ro3)
      w2=ro2/(ro1+ro2+ro3)
      w3=ro3/(ro1+ro2+ro3)
c
      T=X(3*nnodes+1,2)
c
      Vfh_Gama=w1*a11*(b21+T)+w2*a12*(b22+T)+w3*a13*(b23+T)
c
      Di(inode,1)=D01*(exp(Eact1/(R*T)))*exp(-(w1*V1crt+w2*(eps13/
     +          eps23)*V2crt+w3*V3crt*eps13)/Vfh_Gama)
c
      Di(inode,2)=D02*(exp(Eact2/(R*T)))*exp(-(w1*V1crt*(eps23/
     +            eps13)+w2*V2crt+w3*V3crt*eps23)/Vfh_Gama)
c
      enddo
c
      endsubroutine
```

```
c*******************************************************************
c
      subroutine ChemPotVarX(X,nnodes,V1,V2,MM1,MM2,dXChidX,XChi,
     +                      X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +                      X12_1,X12_2,X12_3,
     +                      dmidro)
c
c*******************************************************************
c*******************************************************************
c
c.....Derivatives of chemical potencial of each solvent.
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X13_1,X13_2,X13_3,X23_1,
     +       X23_2,X23_3,X12_1,X12_2,X12_3,
     +       dmidro(nnodes,6),dXChidX(nnodes,9),XChi(nnodes,3)
c
c-------------------------------------------------------------------
c.....Local variables
c-------------------------------------------------------------------
      real*8 MV1,MV2,Phi1,Phi2,Phip
c
c-------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c-------------------------------------------------------------------
c
      dmidro=0
c
      MV1=MM1*V1
      MV2=MM2*V2
c
c-------------------------------------------------------------------
c.....Derivatives of interaction factors
c-------------------------------------------------------------------
      call FormdXChidX (X,nnodes,V1,V2,X13_1,X13_2,X13_3,X23_1,
     +                  X23_2,X23_3,X12_1,X12_2,X12_3,
     +                  dXChidX)
c-------------------------------------------------------------------
c
      do inode=1,nnodes
c
      Phi1=X(inode,2)*V1
      Phi2=X(nnodes+inode,2)*V2
      Phip=1-Phi1-Phi2
c
      dmidro(inode,1)=(1/X(inode,2))-V1+(Phi2*dXChidX(nnodes,7)+Phip*
     +    dXChidX(nnodes,1)-XChi(nnodes,1)*V1)*(Phi2+Phip)-
     +    (XChi(nnodes,3)*Phi2+XChi(nnodes,1)*Phip)*V1-(MV1/MV2)*
     +    Phi2*(dXChidX(nnodes,4)*Phip-XChi(nnodes,2)*V1)
c
      dmidro(inode,2)=-(MV1/MV2)*V2+(1-Phi1)*
     +   ((dXChidX(nnodes,8)*Phi2+XChi(nnodes,3)*V2)+
     +    dXChidX(nnodes,2)*Phip-XChi(nnodes,1)*V2)-(MV1/MV2)*
     +    (dXChidX(nnodes,5)*Phi2*Phip+XChi(nnodes,2)*(V2*Phip-
     +    Phi2*V2))
c
      dmidro(inode,3)=-(MV2/MV1)*V1+(1-Phi2)*
     +   ((dXChidX(nnodes,7)*Phi1+XChi(nnodes,3)*V1)*(MV2/MV1)+
     +    dXChidX(nnodes,4)*Phip-XChi(nnodes,2)*V1)-(MV2/MV1)*
     +    (dXChidX(nnodes,1)*Phi1*Phip+XChi(nnodes,1)*(V1*Phip-
     +    Phi1*V1))
c
      dmidro(inode,4)=(1/X(nnodes+inode,2))-V2+
     +    (dXChidX(nnodes,8)*Phi1*(MV2/MV1)+dXChidX(nnodes,5)*Phip-
     +    XChi(nnodes,2)*V2)*(Phi1+Phip)-(XChi(nnodes,3)*Phi1*
     +    (MV2/MV1)+XChi(nnodes,2)*Phip)*V2-(MV2/MV1)*Phi1*
     +    (dXChidX(nnodes,2)*Phip-XChi(nnodes,1)*V2)
c
      dmidro(inode,5)=(Phi2+Phip)*(Phi2*dXChidX(inode,9)+
     +                Phip*dXChidX(inode,3))-
     +                (MV1/MV2)*Phi2*Phip*dXChidX(inode,6)
c
      dmidro(inode,6)=(Phi1+Phip)*((MV2/MV1)*Phi1*dXChidX(inode,9)+
     +                Phip*dXChidX(inode,6))-
     +                (MV2/MV1)*Phi1*Phip*dXChidX(inode,3)
c
      end do
c
      end subroutine




c***********************************************************************
c
      subroutine FormdXChidX (X,nnodes,V1,V2,X13_1,X13_2,X13_3,X23_1,
     +                        X23_2,X23_3,X12_1,X12_2,X12_3,
     +                        dXChidX)
c
c***********************************************************************
c***********************************************************************
c
c.....Derivatives of interaction factors
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,X13_1,X13_2,X13_3,X23_1,X23_2,X23_3,
     +       X12_1,X12_2,X12_3,
     +       dXChidX(nnodes,9)
c
c-----------------------------------------------------------------------
c
      do inode=1,nnodes
c
      dXChidX(inode,1)=X13_2*V1
      dXChidX(inode,2)=X13_3*V2
      dXChidX(inode,3)=-X13_1/(X(3*nnodes+1,2)**2)
c
```

```
      dXChidX(inode,4)=X23_2*V1
      dXChidX(inode,5)=X23_3*V2
      dXChidX(inode,6)=-X23_1/(X(3*nnodes+1,2)**2)
c
      dXChidX(inode,7)=X12_2*V1
      dXChidX(inode,8)=X12_3*V2
      dXChidX(inode,9)=-X12_1/(X(3*nnodes+1,2)**2)
c
      end do
c
      endsubroutine





c*********************************************************************
c
      subroutine ChemPot(X,nnodes,V1,V2,MM1,MM2,X12,X13,X23,
     +                   dmidro)
c
c*********************************************************************
c*********************************************************************
c
c.....Derivatives of chemical potencial of each solvent.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X12,X13,X23,
     +       dmidro(nnodes,6)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 MV1,MV2,Phi1,Phi2,Phip
c
c---------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c---------------------------------------------------------------------
c
      dmidro=0
c
      MV1=MM1*V1
      MV2=MM2*V2
c
      do inode=1,nnodes
c
      Phi1=X(inode,2)*V1
      Phi2=X(nnodes+inode,2)*V2
      Phip=1-Phi1-Phi2
c
      if ((Phi1.gt.0).and.(Phi2.gt.0)) then
c
      dmidro(inode,1)=(1/X(inode,2))-V1-X13*V1*(Phi2+Phip)-(X12*Phi2+
     +                  X13*Phip)*V1+X23*(MV1/MV2)*Phi2*V1
c
      dmidro(inode,2)= -(MV1/MV2)*V2+(1-Phi1)*(X12*V2-X13*V2)-X23*
     +                 (MV1/MV2)*(V2*Phip-Phi2*V2)
c
      dmidro(inode,3)= -(MV2/MV1)*V1+(1-Phi2)*(X12*V1*(MV2/MV1)-
     +                  X23*V1)-X13*(MV2/MV1)*(V1*Phip-Phi1*V1)
c
      dmidro(inode,4)=(1/X(nnodes+inode,2))-V2-X23*V2*(Phi1+Phip)-
     +                (X12*Phi1*(MV2/MV1)+X23*Phip)*V2+X13*(MV2/MV1)*
     +                Phi1*V2

      endif

      if ((Phi1.gt.0).and.(Phi2.eq.0)) then
c
      dmidro(inode,1)=(1/X(inode,2))-V1-X13*V1*(Phi2+Phip)-(X12*Phi2+
     +                  X13*Phip)*V1+X23*(MV1/MV2)*Phi2*V1
c
      dmidro(inode,2)=0
      dmidro(inode,3)=0
      dmidro(inode,4)=0
c
      endif
c
      if ((Phi1.eq.0).and.(Phi2.gt.0)) then
c
      dmidro(inode,1)=0
      dmidro(inode,2)=0
      dmidro(inode,3)=0
c
      dmidro(inode,4)=(1/X(nnodes+inode,2))-V2-X23*V2*(Phi1+Phip)-
     +                (X12*Phi1*(MV2/MV1)+X23*Phip)*V2+X13*(MV2/MV1)*
     +                Phi1*V2
c
      endif
c
      end do
c
      end subroutine
```

```
c***********************************************************************
c
      subroutine FormAlpha(X,nnodes,A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,V1,V2,
     +           Alpha)
c
c***********************************************************************
c***********************************************************************
c
c.....Calculates the parameter alpha of model 6 - see eq (40) of
c
c        Price.P.and Romdhane,I.,AIChe Jornal,Vol.49 n°2 (2003)
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),A1_0,A1_1,A1_2,A2_0,A2_1,A2_2,V1,V2,
     +       Alpha(nnodes,2)
c-----------------------------------------------------------------------
c
      do inode=1,nnodes
c
      Phi1=X(inode,2)*V1
      Phi2=X(nnodes+inode,2)*V2
c
      Alpha(inode,1)=A1_0+A1_1*Phi1+A1_2*Phi2
      Alpha(inode,2)=A2_0+A2_1*Phi1+A2_2*Phi2
c
      enddo
c
      end subroutine




c***********************************************************************
c
      subroutine MeanDiffCoef (nnodes,D,
     +                         Dm)
c
c***********************************************************************
c***********************************************************************
c
c.....Calculates averages of mutual diffusion coeficients
c
c***********************************************************************
c-----------------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------------
      real*8 D(nnodes,4),
     +       Dm(nnodes-1,4)
c
c-----------------------------------------------------------------------
      Dm=0
c
      do inode=1,nnodes-1
```

```
c
      do j=1,4
c
      Dm(inode,j)=(D(inode+1,j)+D(inode,j))/2
c
      enddo
c
      enddo
c

      endsubroutine



c*****************************************************************
c
      subroutine FormJnew(nnodes,itime,X,tm,hs,hi,Tas,Tai,P1a,P2a,V1,
     +               V2,V3,MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,
     +               X12,A1_1,A1_2,A2_1,A2_2,An,HSub,RoSub,cpSub,A1,B1,
     +               C1,D1,E1,A2,B2,C2,D2,E2,P1,P2,a11,a12,a13,D01,D02,
     +               b21,b22,b23,V1crt,V2crt,V3crt,eps13,eps23,
     +               DifModel,a,Di,dmidro,VP1,VP2,act1,act2,K1,K2,Dm,
     +               tnodes,RoAir,cpAir,dXChidX,XChi,Alpha,Eact1,Eact2,
     +               J)
c
c*****************************************************************
c*****************************************************************
c
c.....Jacobian of each equation.
c
c*****************************************************************
c-----------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------
      integer nnodes,itime,DifModel,tnodes
      real*8 X(3*nnodes+1,2),tm(6*tnodes+1),hs,hi,Tas,Tai,P1a,P2a,V1,V2,
     +      V3,MM1,MM2,DelH1,DelH2,RoBar,cpLBar,X13,X23,X12,HSub,RoSub,
     +       cpSub,P1,P2,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,a11,a12,a13,D01,
     +       D02,b21,b22,b23,V1crt,V2crt,V3crt,eps13,eps23,a,
     +       A1_1,A1_2,A2_1,A2_2,An,Eact1,Eact2,
     +       Di(nnodes,2),VP1,VP2,act1,act2,K1,K2,dmidro(nnodes,4),
     +       Dm(nnodes-1,4),RoAir,cpAir,dDmdX(4*nnodes,2*nnodes+1),
     +       dPdX(3,2),dKdT(2,1),
     +       dXChidX(nnodes,9),XChi(nnodes,3),Alpha(nnodes,2),
     +       J(3*nnodes+1,3*nnodes+1)
c
c-----------------------------------------------------------------
c.....Local variables
c-----------------------------------------------------------------
      real*8 aux1,aux2
c
c-----------------------------------------------------------------
c.....FormdDmdX calculates de average derivative of mutual diffusion coef
c-----------------------------------------------------------------
      call FormdDmdX(nnodes,X,DifModel,Di,V1,V2,V3,a11,b21,a12,
     +                 b22,a13,b23,D01,D02,V1crt,V2crt,V3crt,eps13,
     +                 A1_1,A1_2,A2_1,A2_2,An,eps23,dmidro,MM1,MM2,
     +                 dXChidX,XChi,Alpha,X12,X13,X23,Eact1,Eact2,
     +                 dDmdX)
```

```
c
c-----------------------------------------------------------------------
c.....FormdPdX calculates partial pressure derivatives of solvents
c-----------------------------------------------------------------------
      call FormdPdX(nnodes,X,V1,V2,MM1,MM2,X12,X13,X23,A1,B1,C1,
     +                       D1,E1,A2,B2,C2,D2,E2,VP1,VP2,act1,act2,
     +                    dXChidX,XChi,DifModel,
     +                    dPdX)
c
c-----------------------------------------------------------------------
c.....FormdKdT calculates mass trasfer derivatives of solvents
c-----------------------------------------------------------------------
      call FormdKdT(X,nnodes,Tas,hs,MM1,RoAir,CpAir,MM2,
     +              dKdT)
c
c-----------------------------------------------------------------------
c
      J=0
c
      J(1,1)= -1
c
      J(1,2)=     1
c
      J(nnodes+1,nnodes+1)=-1
c
      J(nnodes+1,nnodes+2)= 1
c
      J(2*nnodes+1,2*nnodes+1)=1
c
      do inode=2, nnodes-1
c
c.....Jacobian of first equation ( Mass conservation law )
c
      J(inode,inode-1)=(X(2*nnodes+inode,2)/X(3*nnodes,2))*
      + ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     + (1/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))-(2/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))*
     + (-dDmdX(inode-1,inode-1)*(X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+
     + Dm(inode-1,1)/(X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(nnodes+inode-1,inode-1)*
     + (X(nnodes+inode,2)-X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     +  X(2*nnodes+inode-1,2)))
*
      J(inode,inode)=1/(tm(itime)-tm(itime-1))-(2/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))*(-Dm(inode,1)/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(inode,inode)*(X(inode+1,2)-X(inode,2))/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))-Dm(inode-1,1)/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(inode-1,inode)*(X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+
     + dDmdX(nnodes+inode,inode)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))-
     + dDmdX(nnodes+inode-1,inode)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
*
```

```
      J(inode,inode+1)=-(X(2*nnodes+inode,2)/X(3*nnodes,2))*
     + ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     + (1/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))-
     + (2/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))*
     + (Dm(inode,1)/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(inode,inode+1)*(X(inode+1,2)-X(inode,2))/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(nnodes+inode,inode+1)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2)))
*
      J(inode,nnodes+inode-1)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(-dDmdX(inode-1,nnodes+inode-1)*
     + (X(inode,2)-X(inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))+Dm(inode-1,2)/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(nnodes+inode-1,nnodes+inode-1)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
*
      J(inode,nnodes+inode)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(dDmdX(inode,nnodes+inode)*
     + (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-dDmdX(inode-1,nnodes+inode)*
     + (X(inode,2)-X(inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))-Dm(inode,2)/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(nnodes+inode,nnodes+inode)*(X(nnodes+inode+1,2)-
     + X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))-
     + Dm(inode-1,2)/(X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(nnodes+inode-1,nnodes+inode)*
     + (X(nnodes+inode,2)-X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
*
      J(inode,nnodes+inode+1)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(dDmdX(inode,nnodes+inode+1)*
     + (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))+Dm(inode,2)/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))+dDmdX(nnodes+inode,nnodes+inode+1)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2)))
*
      J(inode,2*nnodes+inode-1)=-(X(2*nnodes+inode,2)/X(3*nnodes,2))*
     + ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     + (X(inode+1,2)-X(inode-1,2))/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2)-(2/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2))*(Dm(inode,1)*(X(inode+1,2)-
     + X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,1)*(X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+Dm(inode,2)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,2)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(-Dm(inode-1,1)*(X(inode,2)-
     + X(inode-1,2))/((X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))**2)-Dm(inode-1,2)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/((X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))**2))
```

```
*
      J(inode,2*nnodes+inode)=-(1/X(3*nnodes,2))*((X(3*nnodes,2)-
     + X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*(X(inode+1,2)-
     + X(inode-1,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(Dm(inode,1)*
     + (X(inode+1,2)-X(inode,2))/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))**2)+Dm(inode-1,1)*
     + (X(inode,2)-X(inode-1,2))/((X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))**2)+
     + Dm(inode,2)*(X(nnodes+inode+1,2)-X(nnodes+inode,2))/
     + ((X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))**2)+
     + Dm(inode-1,2)*(X(nnodes+inode,2)-X(nnodes+inode-1,2))/
     + ((X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))**2))
*
      J(inode,2*nnodes+inode+1)=(X(2*nnodes+inode,2)/X(3*nnodes,2))*
     + ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     + (X(inode+1,2)-X(inode-1,2))/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2)+(2/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2))*(Dm(inode,1)*
     + (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,1)*(X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+Dm(inode,2)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,2)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(-Dm(inode,1)*(X(inode+1,2)-X(inode,2))/
     + ((X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))**2)-Dm(inode,2)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))**2))
*
      J(inode,3*nnodes)=-X(2*nnodes+inode,2)*(X(inode+1,2)-
     + X(inode-1,2))/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2))*
     + (-(1/(X(3*nnodes,2)**2))*(X(3*nnodes,2)-X(3*nnodes,1))/
     + (tm(itime)-tm(itime-1))+1/(X(3*nnodes,2)*(tm(itime)-
     + tm(itime-1))))
*
      J(inode,3*nnodes+1)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(dDmdX(inode,2*nnodes+1)*
     + (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-dDmdX(inode-1,2*nnodes+1)*
     + (X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+
     + dDmdX(nnodes+inode,2*nnodes+1)*(X(nnodes+inode+1,2)-
     + X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-dDmdX(nnodes+inode-1,2*nnodes+1)*
     + (X(nnodes+inode,2)-X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
c
c.....Jacobian of second equation ( Mass conservation law )
*
      J(nnodes+inode,inode-1)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(-dDmdX(3*nnodes+inode-1,inode-1)*
     + (X(nnodes+inode,2)-X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))+Dm(inode-1,3)/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(2*nnodes+inode-1,inode-1)*(X(inode,2)-
     + X(inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
*
      J(nnodes+inode,inode)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(dDmdX(3*nnodes+inode,inode)*
     +  (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-dDmdX(3*nnodes+inode-1,inode)*
     +  (X(nnodes+inode,2)-X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))-Dm(inode,3)/
     +  (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(2*nnodes+inode,inode)*(X(inode+1,2)-
     + X(inode,2))/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))-
     + Dm(inode-1,3)/(X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(2*nnodes+inode-1,inode)*
     +  (X(inode,2)-X(inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
*
      J(nnodes+inode,inode+1)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(dDmdX(3*nnodes+inode,inode+1)*
     +  (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))+Dm(inode,3)/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))+dDmdX(2*nnodes+inode,inode+1)*
     +  (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2)))
*
      J(nnodes+inode,nnodes+inode-1)=(X(2*nnodes+inode,2)/
     + X(3*nnodes,2))*
     +  ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     +  (1/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))-(2/
     +  (X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))*
     +  (-dDmdX(3*nnodes+inode-1,nnodes+inode-1)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/
     +  (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+
     + Dm(inode-1,4)/(X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(2*nnodes+inode-1,nnodes+inode-1)*
     +  (X(inode,2)-X(inode-1,2))/(X(2*nnodes+inode,2)-
     +  X(2*nnodes+inode-1,2)))
*
      J(nnodes+inode,nnodes+inode)=1/(tm(itime)-tm(itime-1))-(2/
     +  (X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))*(-Dm(inode,4)/
     +  (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(3*nnodes+inode,nnodes+inode)*(X(nnodes+inode+1,2)-
     + X(nnodes+inode,2))/
     +  (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))-Dm(inode-1,4)/
     +  (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))-
     + dDmdX(3*nnodes+inode-1,nnodes+inode)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/
     +  (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+
     + dDmdX(2*nnodes+inode,nnodes+inode)*
     +  (X(inode+1,2)-X(inode,2))/
     +  (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))-
     + dDmdX(2*nnodes+inode-1,nnodes+inode)*(X(inode,2)-
     + X(inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
*
      J(nnodes+inode,nnodes+inode+1)=-(X(2*nnodes+inode,2)/
     +  X(3*nnodes,2))*
     +   ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     +  (1/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))-
     +  (2/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode-1,2)))*
     + (Dm(inode,4)/(X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(3*nnodes+inode,nnodes+inode+1)*(X(nnodes+inode+1,2)-
     + X(nnodes+inode,2))/
     + (X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))+
     + dDmdX(2*nnodes+inode,nnodes+inode+1)*
     + (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2)))
*
      J(nnodes+inode,2*nnodes+inode-1)=-(X(2*nnodes+inode,2)/
     +  X(3*nnodes,2))*
     + ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     + (X(nnodes+inode+1,2)-X(nnodes+inode-1,2))/
     + ((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2)-(2/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2))*(Dm(inode,3)*(X(inode+1,2)-
     + X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,3)*(X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+Dm(inode,4)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,4)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(-Dm(inode-1,3)*(X(inode,2)-
     + X(inode-1,2))/((X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))**2)-Dm(inode-1,4)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/((X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))**2))
*
      J(nnodes+inode,2*nnodes+inode)=-(1/X(3*nnodes,2))*
     +((X(3*nnodes,2)-
     + X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*(X(nnodes+inode+1,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(Dm(inode,4)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))**2)+Dm(inode-1,4)*
     + (X(nnodes+inode,2)-X(nnodes+inode-1,2))/((X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2))**2)+
     + Dm(inode,3)*(X(inode+1,2)-X(inode,2))/
     + ((X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))**2)+
     + Dm(inode-1,3)*(X(inode,2)-X(inode-1,2))/
     + ((X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))**2))
*
      J(nnodes+inode,2*nnodes+inode+1)=(X(2*nnodes+inode,2)/
     + X(3*nnodes,2))*
     + ((X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1)))*
     + (X(nnodes+inode+1,2)-X(nnodes+inode-1,2))/
     + ((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2)+(2/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))**2))*(Dm(inode,3)*
     + (X(inode+1,2)-X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,3)*(X(inode,2)-X(inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+Dm(inode,4)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-Dm(inode-1,4)*(X(nnodes+inode,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(-Dm(inode,3)*(X(inode+1,2)-X(inode,2))/
     + ((X(2*nnodes+inode+1,2)-X(2*nnodes+inode,2))**2)-Dm(inode,4)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/((X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))**2))
*
      J(nnodes+inode,3*nnodes)=-X(2*nnodes+inode,2)*
     + (X(nnodes+inode+1,2)-
     + X(nnodes+inode-1,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2))*
     + (-(1/(X(3*nnodes,2)**2))*(X(3*nnodes,2)-X(3*nnodes,1))/
     + (tm(itime)-tm(itime-1))+1/(X(3*nnodes,2)*(tm(itime)-
     + tm(itime-1))))
*
      J(nnodes+inode,3*nnodes+1)=-(2/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode-1,2)))*(dDmdX(3*nnodes+inode,2*nnodes+1)*
     + (X(nnodes+inode+1,2)-X(nnodes+inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-dDmdX(3*nnodes+inode-1,2*nnodes+1)*
     + (X(nnodes+inode,2)-X(nnodes+inode-1,2))/
     + (X(2*nnodes+inode,2)-X(2*nnodes+inode-1,2))+
     + dDmdX(2*nnodes+inode,2*nnodes+1)*(X(inode+1,2)-
     + X(inode,2))/(X(2*nnodes+inode+1,2)-
     + X(2*nnodes+inode,2))-dDmdX(2*nnodes+inode-1,2*nnodes+1)*
     + (X(inode,2)-X(inode-1,2))/(X(2*nnodes+inode,2)-
     + X(2*nnodes+inode-1,2)))
c
c.....Jacobian of mesh
*
      J(2*nnodes+inode,2*nnodes+inode)=1
c
      aux1=nnodes-inode
      aux2=nnodes-1
*
      J(2*nnodes+inode,3*nnodes)=-(1-(aux1/aux2)**a)
c
      enddo
c
c.....Jacobian of boundary conditions
c
c.....1° equation
*
      J(nnodes,nnodes-1)=+Dm(nnodes-1,1)/(X(3*nnodes,2)-
     + X(3*nnodes-1,2))-dDmdX(nnodes-1,nnodes-1)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-dDmdX(2*nnodes-1,nnodes-1)*
     + (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))
*
      J(nnodes,nnodes)=-Dm(nnodes-1,1)/(X(3*nnodes,2)-
     + X(3*nnodes-1,2))-dDmdX(nnodes-1,nnodes)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-dDmdX(2*nnodes-1,nnodes)*
     + (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + (X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1))-
     + K1*dPdX(1,1)
*
      J(nnodes,2*nnodes-1)=-dDmdX(nnodes-1,2*nnodes-1)*
     + (X(nnodes,2)-X(nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))+
     + Dm(nnodes-1,2)/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + dDmdX(2*nnodes-1,2*nnodes-1)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))
*
      J(nnodes,2*nnodes)=-dDmdX(nnodes-1,2*nnodes)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-Dm(nnodes-1,2)/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-
     + dDmdX(2*nnodes-1,2*nnodes)*
     + (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + K1*dPdX(2,1)
*
      J(nnodes,3*nnodes-1)=-Dm(nnodes-1,1)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)-
     + Dm(nnodes-1,2)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)
*
      J(nnodes,3*nnodes)=Dm(nnodes-1,1)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)+
     + Dm(nnodes-1,2)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)-
     + X(nnodes,2)/(tm(itime)-tm(itime-1))
*
      J(nnodes,3*nnodes+1)=-dDmdX(nnodes-1,2*nnodes+1)*
     + (X(nnodes,2)-X(nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + dDmdX(2*nnodes-1,2*nnodes+1)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-K1*dPdX(3,1)-dKdT(1,1)*(P1-P1a)
c
c.....2° equation
*
      J(2*nnodes,nnodes-1)=Dm(nnodes-1,3)/(X(3*nnodes,2)-
     + X(3*nnodes-1,2))-dDmdX(3*nnodes-1,nnodes-1)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-dDmdX(4*nnodes-1,nnodes-1)*
     + (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))
*
      J(2*nnodes,nnodes)=-Dm(nnodes-1,3)/(X(3*nnodes,2)-
     + X(3*nnodes-1,2))-dDmdX(3*nnodes-1,nnodes)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-dDmdX(4*nnodes-1,nnodes)*
     + (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + K2*dPdX(1,2)
*
      J(2*nnodes,2*nnodes-1)=-dDmdX(3*nnodes-1,2*nnodes-1)*
     + (X(nnodes,2)-X(nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))+
     + Dm(nnodes-1,4)/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + dDmdX(4*nnodes-1,2*nnodes-1)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))
*
      J(2*nnodes,2*nnodes)=-dDmdX(3*nnodes-1,2*nnodes)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-Dm(nnodes-1,4)/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-
     + dDmdX(4*nnodes-1,2*nnodes)*
     + (X(2*nnodes,2)-X(2*nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + (X(3*nnodes,2)-X(3*nnodes,1))/(tm(itime)-tm(itime-1))-
     + K2*dPdX(2,2)
*
      J(2*nnodes,3*nnodes-1)=-Dm(nnodes-1,3)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)-
     + Dm(nnodes-1,4)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)
*
      J(2*nnodes,3*nnodes)=Dm(nnodes-1,3)*
     + (X(nnodes,2)-X(nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)+
     + Dm(nnodes-1,4)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + ((X(3*nnodes,2)-X(3*nnodes-1,2))**2)-
     + X(2*nnodes,2)/(tm(itime)-tm(itime-1))
*
      J(2*nnodes,3*nnodes+1)=-dDmdX(3*nnodes-1,2*nnodes+1)*
     + (X(nnodes,2)-X(nnodes-1,2))/(X(3*nnodes,2)-X(3*nnodes-1,2))-
     + dDmdX(4*nnodes-1,2*nnodes+1)*(X(2*nnodes,2)-X(2*nnodes-1,2))/
     + (X(3*nnodes,2)-X(3*nnodes-1,2))-K2*dPdX(3,2)-dKdT(2,1)*(P2-P2a)
c
c.....Jacobian of mass and heat transport on interface
*
      J(3*nnodes,nnodes)=K1*V1*dPdX(1,1)+K2*V2*dPdX(1,2)
*
      J(3*nnodes,2*nnodes)=K1*V1*dPdX(2,1)+K2*V2*dPdX(2,2)
*
      J(3*nnodes,3*nnodes)=1/(tm(itime)-tm(itime-1))
*
      J(3*nnodes,3*nnodes+1)=K1*V1*dPdX(3,1)+K2*V2*dPdX(3,2)+dKdT(1,1)*
     + V1*(P1-P1a)+dKdT(2,1)*V2*(P2-P2a)
*
      J(3*nnodes+1,nnodes)=(K1*DelH1*dPdX(1,1)+K2*DelH2*dPdX(1,2))*
     + (1/(RoBar*cpLBar*X(3*nnodes,2)+RoSub*cpSub*HSub))
*
      J(3*nnodes+1,2*nnodes)=(K1*DelH1*dPdX(2,1)+K2*DelH2*dPdX(2,2))*
     + (1/(RoBar*cpLBar*X(3*nnodes,2)+RoSub*cpSub*HSub))
*
      J(3*nnodes+1,3*nnodes)=-RoBar*cpLBar*(hs*(X(3*nnodes+1,2)-Tas)+
     + K1*DelH1*(P1-P1a)+K2*DelH2*(P2-P2a)+hi*(X(3*nnodes+1,2)-Tai))*
     + (1/((RoBar*cpLBar*X(3*nnodes,2)+RoSub*cpSub*HSub)**2))
*
      J(3*nnodes+1,3*nnodes+1)=1/(tm(itime)-tm(itime-1))+
     + (hs+K1*DelH1*dPdX(3,1)+K2*DelH2*dPdX(3,2)+hi+dKdT(1,1)*
     + DelH1*(P1-P1a)+dKdT(2,1)*DelH2*(P2-P2a))*
     + (1/(RoBar*cpLBar*X(3*nnodes,2)+RoSub*cpSub*HSub))
c
      endsubroutine
```

```
c*******************************************************************
c
      subroutine FormdDmdX(nnodes,X,DifModel,Di,V1,V2,V3,a11,b21,a12,
     +                b22,a13,b23,D01,D02,V1crt,V2crt,V3crt,eps13,
     +                A1_1,A1_2,A2_1,A2_2,An,eps23,dmidro,MM1,MM2,
     +                dXChidX,XChi,Alpha,X12,X13,X23,Eact1,Eact2,
     +                dDmdX)
c
c*******************************************************************
c*******************************************************************
c
c.....Average derivatives of Mutual Diffusion Coeficients.
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),Di(nnodes,2),V1,V2,V3,a11,b21,a12,b22,a13,
     +       b23,D01,D02,V1crt,V2crt,V3crt,eps13,eps23,MM1,MM2,
     +       A1_1,A1_2,A2_1,A2_2,An,X12,X13,X23,Eact1,Eact2,
     +       dmidro(nnodes,4),dDdX(4*nnodes,2*nnodes+1),
     +       dXChidX(nnodes,9),XChi(nnodes,3),Alpha(nnodes,2),
     +       dDmdX(4*nnodes,2*nnodes+1)
c
c-------------------------------------------------------------------
c.....FormdDdX calculates the derivatives of Mutual Diffusion Coef.
c-------------------------------------------------------------------

      call FormdDdX(nnodes,X,DifModel,Di,V1,V2,V3,a11,b21,a12,
     +                 b22,a13,b23,D01,D02,V1crt,V2crt,V3crt,eps13,An,
     +                 eps23,dmidro,MM1,MM2,A1_1,A1_2,A2_1,A2_2,
     +                 dXChidX,XChi,Alpha,X12,X13,X23,Eact1,Eact2,
     +                 dDdX)
c
      dDmdX=0
c
      do inode=1,nnodes-1
c
      do j=1,2*nnodes+1
c
      dDmdX(inode,j)=(dDdX(inode+1,j)+dDdX(inode,j))/2
c
      dDmdX(nnodes+inode,j)=(dDdX(nnodes+inode+1,j)+
     +            dDdX(nnodes+inode,j))/2
c
      dDmdX(2*nnodes+inode,j)=(dDdX(2*nnodes+inode+1,j)+
     +                 dDdX(2*nnodes+inode,j))/2
c
      dDmdX(3*nnodes+inode,j)=(dDdX(3*nnodes+inode+1,j)+
     +                 dDdX(3*nnodes+inode,j))/2
c
      enddo
c
      enddo
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine  FormdDdX(nnodes,X,DifModel,Di,V1,V2,V3,a11,b21,a12,
     +                  b22,a13,b23,D01,D02,V1crt,V2crt,V3crt,eps13,An,
     +                  eps23,dmidro,MM1,MM2,A1_1,A1_2,A2_1,A2_2,
     +                  dXChidX,XChi,Alpha,X12,X13,X23,Eact1,Eact2,
     +                  dDdX)
c
c*********************************************************************
c*********************************************************************
c
c.....Derivatives of Mutual Diffusion Coeficients.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),Di(nnodes,2),V1,V2,V3,a11,b21,a12,b22,a13,
     +      b23,D01,D02,V1crt,V2crt,V3crt,eps13,eps23,MM1,MM2,
     +      A1_1,A1_2,A2_1,A2_2,An,X12,X13,X23,Eact1,Eact2,
     +      dmidro(nnodes,4),dDidX(2*nnodes,2*nnodes+1),Alpha(nnodes,2),
     +      dXChidX(nnodes,9),XChi(nnodes,3),dDdX(4*nnodes,2*nnodes+1)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 MV1,MV2,
     +       dmi2dX2(nnodes,12),dAldro(nnodes,4)
c
c---------------------------------------------------------------------
c.....MVi:  Molar volume of component i   [cm3/mol]
c---------------------------------------------------------------------
c
c---------------------------------------------------------------------
c.....FormdDidX calculates the Self Diffusion Coef. derivatives
c---------------------------------------------------------------------
      call FormdDidX (nnodes,X,V1,V2,V3,a11,b21,a12,b22,a13,b23,
     +               D01,D02,V1crt,V2crt,V3crt,eps13,eps23,Eact1,Eact2,
     +               dDidX)
c---------------------------------------------------------------------
c.....Formdmi2dX2 calculates second derivatives of chemical potencial
c---------------------------------------------------------------------
      call Formdmi2dX2(X,nnodes,DifModel,V1,V2,MM1,MM2,
     +                      dXChidX,XChi,X12,X13,X23,
     +                      dmi2dX2)
c---------------------------------------------------------------------



      MV1=MM1*V1
      MV2=MM2*V2
c
      if (DifModel.eq.1) THEN
c
      do inode=1,nnodes
c
c.....dD11dX
c
```

```
      dDdX(inode,inode)=dDidX(inode,inode)*X(inode,2)*dmidro(inode,1)+
     +    Di(inode,1)*(dmidro(inode,1)+X(inode,2)*dmi2dX2(inode,1))

c
      dDdX(inode,nnodes+inode)=X(inode,2)*(dDidX(inode,nnodes+inode)*
     +    dmidro(inode,1)+Di(inode,1)*dmi2dX2(inode,2))
c
      dDdX(inode,2*nnodes+1)=X(inode,2)*dmidro(inode,1)*
     +      dDidX(inode,2*nnodes+1)
c
c.....dD12dX
c
      dDdX(nnodes+inode,inode)=dDidX(inode,inode)*X(inode,2)*
     +      dmidro(inode,2)+Di(inode,1)*(dmidro(inode,2)*X(inode,2)*
     +    dmi2dX2(inode,4))
c
      dDdX(nnodes+inode,nnodes+inode)=X(inode,2)*
     +    (dDidX(inode,nnodes+inode)*dmidro(inode,2)+Di(inode,1)*
     +    dmi2dX2(inode,5))
c
      dDdX(nnodes+inode,2*nnodes+1)=X(inode,2)*dmidro(inode,2)*
     +    dDidX(inode,2*nnodes+1)
c
c.....dD21dX
c
      dDdX(2*nnodes+inode,inode)=X(nnodes+inode,2)*
     +    (dDidX(nnodes+inode,inode)*dmidro(inode,3)+Di(inode,2)*
     +    dmi2dX2(inode,7))
c
      dDdX(2*nnodes+inode,nnodes+inode)=
     +      dDidX(nnodes+inode,nnodes+inode)*X(nnodes+inode,2)*
     +      dmidro(inode,3)+Di(inode,2)*(dmidro(inode,3)*
     +    X(nnodes+inode,2)*dmi2dX2(inode,8))
c
      dDdX(2*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*
     +    dmidro(inode,3)*dDidX(nnodes+inode,2*nnodes+1)
c
c.....dD22dX
c
      dDdX(3*nnodes+inode,inode)=X(nnodes+inode,2)*
     +   (dDidX(nnodes+inode,inode)*dmidro(inode,4)+Di(inode,2)*
     +    dmi2dX2(inode,10))
c
      dDdX(3*nnodes+inode,nnodes+inode)=
     +      dDidX(nnodes+inode,nnodes+inode)*X(nnodes+inode,2)*
     +      dmidro(inode,4)+Di(inode,2)*(dmidro(inode,4)*
     +    X(nnodes+inode,2)*dmi2dX2(inode,11))
c
      dDdX(3*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*
     +    dmidro(inode,4)*dDidX(nnodes+inode,2*nnodes+1)


c
      end do
c
      else if(DifModel.eq.2) THEN
c
      do inode=1,nnodes
c
```

```
c.....dD11dX
c
      dDdX(inode,inode)=dDidX(inode,inode)*X(inode,2)*dmidro(inode,1)+
     +      Di(inode,1)*(dmidro(inode,1)+X(inode,2)*dmi2dX2(inode,1))

c
      dDdX(inode,nnodes+inode)=X(inode,2)*(dDidX(inode,nnodes+inode)*
     +    dmidro(inode,1)+Di(inode,1)*dmi2dX2(inode,2))
c
      dDdX(inode,2*nnodes+1)=X(inode,2)*dmidro(inode,1)*
     +      dDidX(inode,2*nnodes+1)
c
c.....dD12dX
c
      dDdX(nnodes+inode,inode)=0
c
      dDdX(nnodes+inode,nnodes+inode)=0
c
      dDdX(nnodes+inode,2*nnodes+1)=0
c
c.....dD21dX
c
      dDdX(2*nnodes+inode,inode)=0
c
      dDdX(2*nnodes+inode,nnodes+inode)=0
c
      dDdX(2*nnodes+inode,2*nnodes+1)=0
c
c.....dD22dX
c
      dDdX(3*nnodes+inode,inode)=X(nnodes+inode,2)*
     +   (dDidX(nnodes+inode,inode)*dmidro(inode,4)+Di(inode,2)*
     +    dmi2dX2(inode,10))
c
      dDdX(3*nnodes+inode,nnodes+inode)=
     +      dDidX(nnodes+inode,nnodes+inode)*X(nnodes+inode,2)*
     +      dmidro(inode,4)+Di(inode,2)*(dmidro(inode,4)*
     +    X(nnodes+inode,2)*dmi2dX2(inode,11))
c
      dDdX(3*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*
     +    dmidro(inode,4)*dDidX(nnodes+inode,2*nnodes+1)


c
      end do
c
      else if(DifModel.eq.3) THEN
c
      do inode=1,nnodes
c
c.....dD11dX
c
      dDdX(inode,inode)=dDidX(inode,inode)
c
      dDdX(inode,nnodes+inode)=dDidX(inode,nnodes+inode)
c
      dDdX(inode,2*nnodes+1)=dDidX(inode,2*nnodes+1)
c
c.....dD12dX
```

```
c
      dDdX(nnodes+inode,inode)=0
c
      dDdX(nnodes+inode,nnodes+inode)=0
c
      dDdX(nnodes+inode,2*nnodes+1)=0
c
c.....dD21dX
c
      dDdX(2*nnodes+inode,inode)=0
c
      dDdX(2*nnodes+inode,nnodes+inode)=0
c
      dDdX(2*nnodes+inode,2*nnodes+1)=0
c
c.....dD22dX
c
      dDdX(3*nnodes+inode,inode)=dDidX(nnodes+inode,inode)
c
      dDdX(3*nnodes+inode,nnodes+inode)=
     +     dDidX(nnodes+inode,nnodes+inode)
c
      dDdX(3*nnodes+inode,2*nnodes+1)=dDidX(nnodes+inode,2*nnodes+1)


c
      end do
c
      else if(DifModel.eq.4) THEN
c
      do inode=1,nnodes
c
c.....dD11dX
c
      dDdX(inode,inode)=(1-2*X(inode,2)*V1)*Di(inode,1)*
     +       dmidro(inode,1)+X(inode,2)*(1-X(inode,2)*V1)*
     +     (dDidX(inode,inode)*dmidro(inode,1)+Di(inode,1)*
     +     dmi2dX2(inode,1))-X(nnodes+inode,2)*V2*(Di(inode,2)*
     +     dmidro(inode,3)+X(inode,2)*(dDidX(nnodes+inode,inode)*
     +     dmidro(inode,3)+Di(inode,2)*dmi2dX2(inode,7)))
c
      dDdX(inode,nnodes+inode)=X(inode,2)*(1-X(inode,2)*V1)*
     +     (dDidX(inode,nnodes+inode)*dmidro(inode,1)+
     +     Di(inode,1)*dmi2dX2(inode,2))-X(inode,2)*V2*
     +     (Di(inode,2)*dmidro(inode,3)+X(nnodes+inode,2)*
     +     (dDidX(nnodes+inode,nnodes+inode)*dmidro(inode,3)+
     +     Di(inode,2)*dmi2dx2(inode,8)))
c
      dDdX(inode,2*nnodes+1)=X(inode,2)*(1-X(inode,2)*V1)*
     +        dmidro(inode,1)*dDidX(inode,2*nnodes+1)-X(inode,2)*
     +        X(nnodes+inode,2)*V2*dmidro(inode,3)*
     +        dDidX(nnodes+inode,2*nnodes+1)
c
c.....dD12dX
c
      dDdX(nnodes+inode,inode)=(1-2*X(inode,2)*V1)*Di(inode,1)*
     +       dmidro(inode,2)+X(inode,2)*(1-X(inode,2)*V1)*
     +     (dDidX(inode,inode)*dmidro(inode,2)+Di(inode,1)*
     +     dmi2dX2(inode,4))-X(nnodes+inode,2)*V2*(Di(inode,2)*
     +     dmidro(inode,4)+X(inode,2)*(dDidX(nnodes+inode,inode)*
     +     dmidro(inode,4)+Di(inode,2)*dmi2dX2(inode,10)))
c
      dDdX(nnodes+inode,nnodes+inode)=X(inode,2)*(1-X(inode,2)*V1)*
     +    (dDidX(inode,nnodes+inode)*dmidro(inode,2)+
     +     Di(inode,1)*dmi2dX2(inode,5))-X(inode,2)*V2*
     +     (Di(inode,2)*dmidro(inode,4)+X(nnodes+inode,2)*
     +     (dDidX(nnodes+inode,nnodes+inode)*dmidro(inode,4)+
     +     Di(inode,2)*dmi2dx2(inode,11)))
c
      dDdX(nnodes+inode,2*nnodes+1)=X(inode,2)*(1-X(inode,2)*V1)*
     +       dmidro(inode,2)*dDidX(inode,2*nnodes+1)-X(inode,2)*
     +        X(nnodes+inode,2)*V2*dmidro(inode,4)*
     +        dDidX(nnodes+inode,2*nnodes+1)
c
c.....dD21dX
c
      dDdX(2*nnodes+inode,inode)=X(nnodes+inode,2)*(1-
     +      X(nnodes+inode,2)*V2)*
     +    (dDidX(nnodes+inode,inode)*dmidro(inode,3)+
     +     Di(inode,2)*dmi2dX2(inode,7))-X(nnodes+inode,2)*V1*
     +     (Di(inode,1)*dmidro(inode,1)+X(inode,2)*
     +     (dDidX(inode,inode)*dmidro(inode,1)+
     +     Di(inode,1)*dmi2dx2(inode,1)))
c
      dDdX(2*nnodes+inode,nnodes+inode)=(1-2*X(nnodes+inode,2)*V2)*
     +      Di(inode,2)*dmidro(inode,3)+X(nnodes+inode,2)*(1-
     +    X(nnodes+inode,2)*V2)*(dDidX(nnodes+inode,nnodes+inode)*
     +    dmidro(inode,3)+Di(inode,2)*dmi2dX2(inode,8))-X(inode,2)*V1*
     +    (Di(inode,1)*dmidro(inode,1)+X(nnodes+inode,2)*
     +    (dDidX(inode,nnodes+inode)*
     +     dmidro(inode,1)+Di(inode,1)*dmi2dX2(inode,2)))
c
      dDdX(2*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*(1-
     +      X(nnodes+inode,2)*V2)*dmidro(inode,3)*
     +    dDidX(nnodes+inode,2*nnodes+1)-X(inode,2)*
     +    X(nnodes+inode,2)*V1*dmidro(inode,1)*
     +    dDidX(inode,2*nnodes+1)
c
c.....dD22dX
c
      dDdX(3*nnodes+inode,inode)=X(nnodes+inode,2)*(1-
     +     X(nnodes+inode,2)*V2)*
     +    (dDidX(nnodes+inode,inode)*dmidro(inode,4)+
     +     Di(inode,2)*dmi2dX2(inode,10))-X(nnodes+inode,2)*V1*
     +     (Di(inode,1)*dmidro(inode,2)+X(inode,2)*
     +     (dDidX(inode,inode)*dmidro(inode,2)+
     +     Di(inode,1)*dmi2dx2(inode,4)))
c
      dDdX(3*nnodes+inode,nnodes+inode)=(1-2*X(nnodes+inode,2)*V2)*
     +      Di(inode,2)*dmidro(inode,4)+X(nnodes+inode,2)*(1-
     +    X(nnodes+inode,2)*V2)*(dDidX(nnodes+inode,nnodes+inode)*
     +    dmidro(inode,4)+Di(inode,2)*dmi2dX2(inode,11))-X(inode,2)*V1*
     +    (Di(inode,1)*dmidro(inode,2)+X(nnodes+inode,2)*
     +    (dDidX(inode,nnodes+inode)*
     +     dmidro(inode,2)+Di(inode,1)*dmi2dX2(inode,5)))
c
      dDdX(3*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*(1-
     +    X(nnodes+inode,2)*V2)*dmidro(inode,4)*
     +    dDidX(nnodes+inode,2*nnodes+1)-X(inode,2)*
     +    X(nnodes+inode,2)*V1*dmidro(inode,2)*
     +    dDidX(inode,2*nnodes+1)

c
      enddo
c
      else if(DifModel.eq.5) THEN
c
      do inode=1,nnodes
c
c.....dD11dX
c
      dDdX(inode,inode)=(Di(inode,1)+X(inode,2)*dDidX(inode,inode))*
     +    (1-X(inode,2)*V1+X(inode,2)*V3)*dmidro(inode,1)+X(inode,2)*
     +     Di(inode,1)*((V3-V1)*dmidro(inode,1)+(1-X(inode,2)*V1+
     +    X(inode,2)*V3)*dmi2dX2(inode,1))+X(nnodes+inode,2)*(V3-V2)*
     +    (Di(inode,2)*dmidro(inode,3)+X(inode,2)*
     +    (dDidX(nnodes+inode,inode)*dmidro(inode,3)+Di(inode,2)*
     +    dmi2dX2(inode,7)))
c
      dDdX(inode,nnodes+inode)=X(inode,2)*(1-X(inode,2)*V1+
     +       X(inode,2)*V3)*(dDidX(inode,nnodes+inode)*
     +     dmidro(inode,1)+Di(inode,1)*dmi2dX2(inode,2))+X(inode,2)*
     +     (V3-V2)*(Di(inode,2)*dmidro(inode,3)+X(nnodes+inode,2)*
     +     (dDidX(nnodes+inode,nnodes+inode)*dmidro(inode,3)+
     +     Di(inode,2)*dmi2dX2(inode,8)))
c
      dDdX(inode,2*nnodes+1)=X(inode,2)*(1-X(inode,2)*V1+
     +            X(inode,2)*V3)*dmidro(inode,1)*dDidX(inode,2*nnodes+1)+
     +        X(inode,2)*X(nnodes+inode,2)*(V3-V2)*dmidro(inode,3)*
     +        dDidX(nnodes+inode,2*nnodes+1)
c
c.....dD12dX
c
      dDdX(nnodes+inode,inode)=(Di(inode,1)+X(inode,2)*
     +      dDidX(inode,inode))*
     +    (1-X(inode,2)*V1+X(inode,2)*V3)*dmidro(inode,2)+X(inode,2)*
     +     Di(inode,1)*((V3-V1)*dmidro(inode,2)+(1-X(inode,2)*V1+
     +    X(inode,2)*V3)*dmi2dX2(inode,4))+X(nnodes+inode,2)*(V3-V2)*
     +    (Di(inode,2)*dmidro(inode,4)+X(inode,2)*
     +    (dDidX(nnodes+inode,inode)*dmidro(inode,4)+Di(inode,2)*
     +    dmi2dX2(inode,10)))
c
      dDdX(nnodes+inode,nnodes+inode)=X(inode,2)*(1-X(inode,2)*V1+
     +       X(inode,2)*V3)*(dDidX(inode,nnodes+inode)*
     +     dmidro(inode,2)+Di(inode,1)*dmi2dX2(inode,5))+X(inode,2)*
     +     (V3-V2)*(Di(inode,2)*dmidro(inode,4)+X(nnodes+inode,2)*
     +     (dDidX(nnodes+inode,nnodes+inode)*dmidro(inode,4)+
     +     Di(inode,2)*dmi2dX2(inode,11)))
c
      dDdX(nnodes+inode,2*nnodes+1)=X(inode,2)*(1-X(inode,2)*V1+
     +            X(inode,2)*V3)*dmidro(inode,2)*dDidX(inode,2*nnodes+1)+
     +        X(inode,2)*X(nnodes+inode,2)*(V3-V2)*dmidro(inode,4)*
     +        dDidX(nnodes+inode,2*nnodes+1)
c
c.....dD21dX
c
      dDdX(2*nnodes+inode,inode)=X(nnodes+inode,2)*(1-X(nnodes+inode,2)*
     +      V2+X(nnodes+inode,2)*V3)*(dDidX(nnodes+inode,inode)*
     +      dmidro(inode,3)+Di(inode,2)*dmi2dX2(inode,7))+
     +      X(nnodes+inode,2)*(V3-V1)*(Di(inode,1)*dmidro(inode,1)+
     +      X(inode,2)*(dDidX(inode,inode)*dmidro(inode,1)+
     +      Di(inode,1)*dmi2dX2(inode,1)))

      dDdX(2*nnodes+inode,nnodes+inode)=(Di(inode,2)+X(nnodes+inode,2)*
     +     dDidX(nnodes+inode,nnodes+inode))*
     +     (1-X(nnodes+inode,2)*V2+X(nnodes+inode,2)*V3)*
     +     dmidro(inode,3)+X(nnodes+inode,2)*Di(inode,2)*((V3-V2)*
     +     dmidro(inode,3)+(1-X(nnodes+inode,2)*V2+X(nnodes+inode,2)*
     +     V3)*dmi2dX2(inode,8))+X(inode,2)*(V3-V1)*
     +     (Di(inode,1)*dmidro(inode,1)+X(nnodes+inode,2)*
     +     (dDidX(inode,nnodes+inode)*dmidro(inode,1)+Di(inode,1)*
     +     dmi2dX2(inode,2)))
c
      dDdX(2*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*
     +             (1-X(nnodes+inode,2)*V2+X(nnodes+inode,2)*V3)*
     +         dmidro(inode,3)*dDidX(nnodes+inode,2*nnodes+1)+
     +         X(inode,2)*X(nnodes+inode,2)*(V3-V1)*dmidro(inode,1)*
     +         dDidX(inode,2*nnodes+1)
c
c.....dD22dX
c
      dDdX(3*nnodes+inode,inode)=X(nnodes+inode,2)*(1-X(nnodes+inode,2)*
     +     V2+X(nnodes+inode,2)*V3)*(dDidX(nnodes+inode,inode)*
     +      dmidro(inode,4)+Di(inode,2)*dmi2dX2(inode,10))+
     +      X(nnodes+inode,2)*(V3-V1)*(Di(inode,1)*dmidro(inode,2)+
     +      X(inode,2)*(dDidX(inode,inode)*dmidro(inode,2)+
     +      Di(inode,1)*dmi2dX2(inode,4)))
c
      dDdX(3*nnodes+inode,nnodes+inode)=(Di(inode,2)+X(nnodes+inode,2)*
     +     dDidX(nnodes+inode,nnodes+inode))*
     +     (1-X(nnodes+inode,2)*V2+X(nnodes+inode,2)*V3)*
     +     dmidro(inode,4)+X(nnodes+inode,2)*Di(inode,2)*((V3-V2)*
     +     dmidro(inode,4)+(1-X(nnodes+inode,2)*V2+X(nnodes+inode,2)*
     +     V3)*dmi2dX2(inode,11))+X(inode,2)*(V3-V1)*
     +     (Di(inode,1)*dmidro(inode,2)+X(nnodes+inode,2)*
     +     (dDidX(inode,nnodes+inode)*dmidro(inode,2)+Di(inode,1)*
     +     dmi2dX2(inode,5)))
c
      dDdX(3*nnodes+inode,2*nnodes+1)=X(nnodes+inode,2)*
     +        (1-X(nnodes+inode,2)*V2+X(nnodes+inode,2)*V3)*
     +      dmidro(inode,4)*dDidX(nnodes+inode,2*nnodes+1)+
     +      X(inode,2)*X(nnodes+inode,2)*(V3-V1)*dmidro(inode,2)*
     +      dDidX(inode,2*nnodes+1)
c
      enddo
c
      else if(DifModel.eq.6) THEN
c
c-------------------------------------------------------------------
c.....dAdro calculates derivatives of parameter alpha - model 6
c-------------------------------------------------------------------
      call dAdro(nnodes,A1_1,A1_2,A2_1,A2_2,V1,V2,
     +                dA1dro)
c-------------------------------------------------------------------

c
```

```
      do inode=1,nnodes
c
c.....dD11dX
c
      dDdX(inode,inode)=Di(inode,1)*dmidro(inode,1)*((-V1*(1-
     +         (Alpha(inode,1)/An))-X(inode,2)*V1*((-1/An)*
     +         dAldro(inode,1)))*X(inode,2)+(1-X(inode,2)*V1*(1-
     +         (Alpha(inode,1)/An))))+(dmidro(inode,1)*
     +         dDidX(inode,inode)+dmi2dX2(inode,1)*Di(inode,1))*
     +         (1-X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*X(inode,2)-
     +         X(nnodes+inode,2)*V2*((((-1/An)*dAldro(inode,3))*
     +         X(inode,2)+(1-(Alpha(inode,2)/An)))*Di(inode,2)*
     +         dmidro(inode,3)+(1-(Alpha(inode,2)/An))*X(inode,2)*
     +         (dDidX(nnodes+inode,inode)*dmidro(inode,3)+
     +         Di(inode,2)*dmi2dX2(inode,7)))
c
      dDdX(inode,nnodes+inode)=X(inode,2)*((X(inode,2)*V1*(1/An)*
     +         dAldro(inode,2))*Di(inode,1)*dmidro(inode,1)+(1-
     +         X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*(Di(inode,1)*
     +         dmi2dX2(inode,2)+dDidX(inode,nnodes+inode)*
     +         dmidro(inode,1)))-X(inode,2)*V2*(((-1/An)*
     +         dAldro(inode,4)*X(nnodes+inode,2)+(1-
     +         (Alpha(inode,2)/An)))*Di(inode,2)*dmidro(inode,3)+
     +         (1-(Alpha(inode,2)/An))*X(nnodes+inode,2)*
     +         (Di(inode,2)*dmi2dX2(inode,8)+
     +         dDidX(nnodes+inode,nnodes+inode)*dmidro(inode,3)))
c
      dDdX(inode,2*nnodes+1)=(1-X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*
     +         X(inode,2)*(dmidro(inode,1)*dDidX(inode,2*nnodes+1)+
     +         dmi2dX2(inode,3)*Di(inode,1))-(1-(Alpha(inode,2)/An))*
     +         X(nnodes+inode,2)*V2*X(inode,2)*(dmidro(inode,3)*
     +         dDidX(nnodes+inode,2*nnodes+1)+dmi2dX2(inode,9)*
     +         Di(inode,2))
c
c.....dD12dX
c
      dDdX(nnodes+inode,inode)=Di(inode,1)*dmidro(inode,2)*((-V1*(1-
     +         (Alpha(inode,1)/An))-X(inode,2)*V1*((-1/An)*
     +         dAldro(inode,1)))*X(inode,2)+(1-X(inode,2)*V1*(1-
     +         (Alpha(inode,1)/An))))+(dmidro(inode,2)*
     +         dDidX(inode,inode)+dmi2dX2(inode,4)*Di(inode,1))*
     +         (1-X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*X(inode,2)-
     +         X(nnodes+inode,2)*V2*((((-1/An)*dAldro(inode,3))*
     +         X(inode,2)+(1-(Alpha(inode,2)/An)))*Di(inode,2)*
     +         dmidro(inode,4)+(1-(Alpha(inode,2)/An))*X(inode,2)*
     +         (dDidX(nnodes+inode,inode)*dmidro(inode,4)+
     +         Di(inode,2)*dmi2dX2(inode,10)))
c
      dDdX(inode,nnodes+inode)=X(inode,2)*((X(inode,2)*V1*(1/An)*
     +         dAldro(inode,2))*Di(inode,1)*dmidro(inode,2)+(1-
     +         X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*(Di(inode,1)*
     +         dmi2dX2(inode,5)+dDidX(inode,nnodes+inode)*
     +         dmidro(inode,2)))-X(inode,2)*V2*(((-1/An)*
     +         dAldro(inode,4)*X(nnodes+inode,2)+(1-
     +         (Alpha(inode,2)/An)))*Di(inode,2)*dmidro(inode,4)+
     +         (1-(Alpha(inode,2)/An))*X(nnodes+inode,2)*
     +         (Di(inode,2)*dmi2dX2(inode,11)+
     +         dDidX(nnodes+inode,nnodes+inode)*dmidro(inode,4)))
c
```

```
        dDdX(inode,2*nnodes+1)=(1-X(inode,2)*V1*(1-(Alpha(inode,1)/An)))*
     +          X(inode,2)*(dmidro(inode,2)*dDidX(inode,2*nnodes+1)+
     +          dmi2dX2(inode,6)*Di(inode,1))-(1-(Alpha(inode,2)/An))*
     +          X(nnodes+inode,2)*V2*X(inode,2)*(dmidro(inode,4)*
     +          dDidX(nnodes+inode,2*nnodes+1)+dmi2dX2(inode,12)*
     +          Di(inode,2))

c.....dD21dX
c
        dDdX(2*nnodes+inode,inode)=X(nnodes+inode,2)*((X(nnodes+inode,2)*
     +       V2*(1/An)*dAldro(inode,3))*Di(inode,2)*dmidro(inode,3)+
     +          (1-X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An)))*
     +          (Di(inode,2)*dmi2dX2(inode,7)+dDidX(nnodes+inode,inode)*
     +          dmidro(inode,3)))-X(nnodes+inode,2)*V1*(((-1/An)*
     +          dAldro(inode,1)*X(inode,2)+(1-(Alpha(inode,1)/An)))*
     +          Di(inode,1)*dmidro(inode,1)+(1-(Alpha(inode,1)/An))*
     +          X(inode,2)*(Di(inode,1)*dmi2dX2(inode,1)+
     +          dDidX(inode,inode)*dmidro(inode,1)))

        dDdX(2*nnodes+inode,nnodes+inode)=Di(inode,2)*dmidro(inode,3)*
     +         ((-V2*(1-(Alpha(inode,2)/An))-X(nnodes+inode,2)*V2*
     +         ((-1/An)*dAldro(inode,4)))*X(nnodes+inode,2)+(1-
     +          X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An))))+
     +         (dmidro(inode,3)*dDidX(nnodes+inode,nnodes+inode)+
     +          dmi2dX2(inode,8)*Di(inode,2))*
     +          (1-X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An)))*
     +          X(nnodes+inode,2)-X(inode,2)*V1*((((-1/An)*
     +          dAldro(inode,2))*X(nnodes+inode,2)+(1-
     +          (Alpha(inode,1)/An)))*Di(inode,1)*
     +          dmidro(inode,1)+(1-(Alpha(inode,1)/An))*
     +          X(nnodes+inode,2)*(dDidX(inode,nnodes+inode)*
     +          dmidro(inode,1)+Di(inode,1)*dmi2dX2(inode,2)))
c
        dDdX(2*nnodes+inode,2*nnodes+1)=(1-X(nnodes+inode,2)*V2*
     +         (1-(Alpha(inode,2)/An)))*X(nnodes+inode,2)*
     +         (dmidro(inode,3)*dDidX(nnodes+inode,2*nnodes+1)+
     +          dmi2dX2(inode,9)*Di(inode,2))-(1-(Alpha(inode,1)/An))*
     +          X(inode,2)*V1*X(nnodes+inode,2)*(dmidro(inode,1)*
     +          dDidX(inode,2*nnodes+1)+dmi2dX2(inode,3)*
     +          Di(inode,1))
c
c.....dD22dX
c
        dDdX(3*nnodes+inode,inode)=X(nnodes+inode,2)*((X(nnodes+inode,2)*
     +       V2*(1/An)*dAldro(inode,3))*Di(inode,2)*dmidro(inode,4)+
     +         (1-X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An)))*
     +         (Di(inode,2)*dmi2dX2(inode,10)+dDidX(nnodes+inode,inode)*
     +          dmidro(inode,4)))-X(nnodes+inode,2)*V1*(((-1/An)*
     +          dAldro(inode,1)*X(inode,2)+(1-(Alpha(inode,1)/An)))*
     +          Di(inode,1)*dmidro(inode,2)+(1-(Alpha(inode,1)/An))*
     +          X(inode,2)*(Di(inode,1)*dmi2dX2(inode,4)+
     +          dDidX(inode,inode)*dmidro(inode,2)))
c
        dDdX(3*nnodes+inode,nnodes+inode)=Di(inode,2)*dmidro(inode,4)*
     +        ((-V2*(1-(Alpha(inode,2)/An))-X(nnodes+inode,2)*V2*
     +         ((-1/An)*dAldro(inode,4)))*X(nnodes+inode,2)+(1-
     +          X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An))))+
     +         (dmidro(inode,4)*dDidX(nnodes+inode,nnodes+inode)+
     +          dmi2dX2(inode,11)*Di(inode,2))*
     +        (1-X(nnodes+inode,2)*V2*(1-(Alpha(inode,2)/An)))*
     +        X(nnodes+inode,2)-X(inode,2)*V1*((((-1/An)*
     +        dAldro(inode,2))*X(nnodes+inode,2)+(1-
     +        (Alpha(inode,1)/An)))*Di(inode,1)*
     +        dmidro(inode,2)+(1-(Alpha(inode,1)/An))*
     +        X(nnodes+inode,2)*(dDidX(inode,nnodes+inode)*
     +        dmidro(inode,2)+Di(inode,1)*dmi2dX2(inode,5)))
c
      dDdX(3*nnodes+inode,2*nnodes+1)=(1-X(nnodes+inode,2)*V2*
     +     (1-(Alpha(inode,2)/An)))*X(nnodes+inode,2)*
     +       (dmidro(inode,4)*dDidX(nnodes+inode,2*nnodes+1)+
     +        dmi2dX2(inode,12)*Di(inode,2))-(1-(Alpha(inode,1)/An))*
     +        X(inode,2)*V1*X(nnodes+inode,2)*(dmidro(inode,2)*
     +        dDidX(inode,2*nnodes+1)+dmi2dX2(inode,6)*
     +        Di(inode,1))
c
      enddo
c
      else
c
      do inode=1,nnodes
c
c.....dD11dX
c
      dDdX(inode,inode)=0.0
c
      dDdX(inode,nnodes+inode)=0.0
c
      dDdX(inode,2*nnodes+1)=0.0
c
c.....dD12dX
c
      dDdX(nnodes+inode,inode)=0.0
c
      dDdX(nnodes+inode,nnodes+inode)=0.0
c
      dDdX(nnodes+inode,2*nnodes+1)=0.0
c
c.....dD21dX
c
      dDdX(2*nnodes+inode,inode)=0.0
c
      dDdX(2*nnodes+inode,nnodes+inode)=0.0
c
      dDdX(2*nnodes+inode,2*nnodes+1)=0.0
c
c.....dD22dX
c
      dDdX(3*nnodes+inode,inode)=0.0
c
      dDdX(3*nnodes+inode,nnodes+inode)=0.0
c
      dDdX(3*nnodes+inode,2*nnodes+1)=0.0
c
      enddo
c
      endif
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine FormdDidX(nnodes,X,V1,V2,V3,a11,b21,a12,b22,a13,b23,
     +               D01,D02,V1crt,V2crt,V3crt,eps13,eps23,Eact1,Eact2,
     +               dDidX)
c
c*********************************************************************
c*********************************************************************
c
c.....Derivatives of Self Diffusion Coeficients.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,V3,a11,b21,a12,b22,a13,Eact1,Eact2,
     +       b23,D01,D02,V1crt,V2crt,V3crt,eps13,eps23,
     +       dwdro(nnodes,6),dVfhdX(nnodes,3),
     +       dDidX(2*nnodes,2*nnodes+1)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 T,ro1,ro2,ro3,w1,w2,w3,Vfh_Gama
c
c---------------------------------------------------------------------
c.....T  : Current liquid temperature           [C]
c.....roi: Concentration of component i     [g/cm3]
c.....wi : Mass fraction of component i
c.....Vfh_Gama:Variable of free volume theory
c---------------------------------------------------------------------
c---------------------------------------------------------------------
c.....Gas Constant                          [cal/mol.K]
c---------------------------------------------------------------------
      R=2.0
c---------------------------------------------------------------------
c.....Formdwdro calculates de derivatives of mass fraction of each
c......component
c---------------------------------------------------------------------
      call Formdwdro(nnodes,X,V1,V2,V3,
     +               dwdro)
c
c---------------------------------------------------------------------
c.....FormdVfhdX calculates de derivative of Vfh_Gama of each component
c---------------------------------------------------------------------
      call FormdVfhdX(nnodes,X,V1,V2,V3,a11,b21,a12,b22,dwdro,a13,
     +                b23,
     +                dVfhdX)
c
      T=X(3*nnodes+1,2)
      dDidX=0
c
      do inode=1,nnodes
c
      ro1=X(inode,2)
      ro2=X(nnodes+inode,2)
      ro3=(1-(V1*ro1+V2*ro2))/V3
c
```

```
      w1=ro1/(ro1+ro2+ro3)
      w2=ro2/(ro1+ro2+ro3)
      w3=ro3/(ro1+ro2+ro3)
c
      Vfh_Gama=w1*a11*(b21+T)+w2*a12*(b22+T)+w3*a13*(b23+T)
c
c.....dD1dX
c
      dDidX(inode,inode)=D01*((exp(-(w1*V1crt+w2*(eps13/eps23)*V2crt+
     +     w3*V3crt*eps13)/Vfh_Gama)*(-1)*((1/Vfh_Gama)*(V1crt*
     +     dwdro(inode,1)+(eps13/eps23)*V2crt*dwdro(inode,3)+V3crt*
     +     eps13*dwdro(inode,5))+((-1/(Vfh_Gama**2))*dVfhdX(inode,1))*
     +     (w1*V1crt+w2*(eps13/eps23)*V2crt+w3*V3crt*eps13)))*
     +     (exp(Eact1/(R*T))))
c
      dDidX(inode,nnodes+inode)=D01*((exp(-(w1*V1crt+w2*(eps13/eps23)*
     +      V2crt+w3*V3crt*eps13)/Vfh_Gama)*(-1)*((1/Vfh_Gama)*(V1crt*
     +     dwdro(inode,2)+(eps13/eps23)*V2crt*dwdro(inode,4)+V3crt*
     +     eps13*dwdro(inode,6))+((-1/(Vfh_Gama**2))*dVfhdX(inode,2))*
     +     (w1*V1crt+w2*(eps13/eps23)*V2crt+w3*V3crt*eps13)))*
     +     (exp(Eact1/(R*T))))
c
      dDidX(inode,2*nnodes+1)=D01*((exp(-(w1*V1crt+w2*(eps13/eps23)*
     +     V2crt+w3*V3crt*eps13)/Vfh_Gama)*((-1)*(w1*V1crt+w2*(eps13/
     +     eps23)*V2crt+w3*V3crt*eps13))*((-1/(Vfh_Gama**2))*
     +     dVfhdX(inode,3)))*(exp(Eact1/(R*T)))+(exp(-(w1*V1crt+w2*
     +     (eps13/eps23)*V2crt+w3*V3crt*eps13)/Vfh_Gama))*
     +     (exp(Eact1/(R*T)))*(-Eact1/(R*(T**2))))

c
c.....dD2dX
c
      dDidX(nnodes+inode,inode)=D02*((exp(-(w1*V1crt*(eps23/eps13)+w2*
     +      V2crt+w3*V3crt*eps23)/Vfh_Gama)*(-1)*((1/Vfh_Gama)*(V1crt*
     +    (eps23/eps13)*dwdro(inode,1)+V2crt*dwdro(inode,3)+V3crt*
     +     eps23*dwdro(inode,5))+((-1/(Vfh_Gama**2))*dVfhdX(inode,1))*
     +     (w1*V1crt*(eps23/eps13)+w2*V2crt+w3*V3crt*eps23)))*
     +     (exp(Eact2/(R*T))))
c
      dDidX(nnodes+inode,nnodes+inode)=D02*((exp(-(w1*V1crt*(eps23/
     +      eps13)+w2*V2crt+w3*V3crt*eps23)/Vfh_Gama)*(-1)*((1/Vfh_Gama)*
     +    (V1crt*(eps23/eps13)*dwdro(inode,2)+V2crt*dwdro(inode,4)+
     +    V3crt*eps23*dwdro(inode,6))+((-1/(Vfh_Gama**2))*
     +    dVfhdX(inode,2))*(w1*V1crt*(eps23/eps13)+w2*V2crt+w3*V3crt*
     +    eps23)))*(exp(Eact2/(R*T))))
c
      dDidX(nnodes+inode,2*nnodes+1)=D02*((exp(-(w1*V1crt*(eps23/eps13)+
     +      w2*V2crt+w3*V3crt*eps23)/Vfh_Gama)*((-1)*(w1*V1crt*
     +    (eps23/eps13)+w2*V2crt+w3*V3crt*eps23))*((-1/(Vfh_Gama**2))*
     +    dVfhdX(inode,3)))*(exp(Eact2/(R*T)))+(exp(-(w1*V1crt*(eps23/
     +    eps13)+w2*V2crt+w3*V3crt*eps23)/Vfh_Gama))*(exp(Eact2/
     +    (R*T)))*(-Eact2/(R*(T**2))))
c
      enddo
c
      endsubroutine
```

```
c******************************************************************
c
      subroutine Formdwdro(nnodes,X,V1,V2,V3,
     +                     dwdro)
c
c******************************************************************
c******************************************************************
c
c.....Derivatives of mass fraction of each component
c
c     wi=roi/(ro1+ro2+ro3)     ( mass fraction )
c
c******************************************************************
c------------------------------------------------------------------
c.....External & returning variables
c------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,V3,
     +       dwdro(nnodes,6)
c
c------------------------------------------------------------------
c.....Local variables
c------------------------------------------------------------------
      real*8 ro1,ro2,ro3

c------------------------------------------------------------------
c.....roi: Concentration of component i     [g/cm3]
c------------------------------------------------------------------
c
      do inode=1,nnodes
c
      ro1=X(inode,2)
      ro2=X(nnodes+inode,2)
      ro3=(1-(V1*ro1+V2*ro2))/V3
c
      dwdro(inode,1)=ro1*(-1/((ro1+ro2+ro3)**2))+(1/(ro1+ro2+ro3))
c
      dwdro(inode,2)=ro1*(-1/((ro1+ro2+ro3)**2))
c
      dwdro(inode,3)=ro2*(-1/((ro1+ro2+ro3)**2))
c
      dwdro(inode,4)=ro2*(-1/((ro1+ro2+ro3)**2))+(1/(ro1+ro2+ro3))
c
      dwdro(inode,5)=ro3*(-1/((ro1+ro2+ro3)**2))
c
      dwdro(inode,6)=ro3*(-1/((ro1+ro2+ro3)**2))
c
      enddo
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine FormdVfhdX(nnodes,X,V1,V2,V3,a11,b21,a12,b22,dwdro,a13,
     +                      b23,
     +                      dVfhdX)
c
c*********************************************************************
c*********************************************************************
c
c.....Derivatives of Vfh_Gama (parameter of free volume theory)
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),V1,V2,V3,a11,b21,a12,b22,a13,b23,
     +       dwdro(nnodes,6),
     +       dVfhdX(nnodes,3)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 T,ro1,ro2,ro3,w1,w2,w3
c
c---------------------------------------------------------------------
c.....T  : Current liquid temperature           [C]
c.....roi: Concentration of component i     [g/cm3]
c.....wi : Mass fraction of component i
c---------------------------------------------------------------------
c
      T=X(3*nnodes+1,2)
c
      do inode=1,nnodes
c
      ro1=X(inode,2)
      ro2=X(nnodes+inode,2)
      ro3=(1-(V1*ro1+V2*ro2))/V3
c
      w1=ro1/(ro1+ro2+ro3)
      w2=ro2/(ro1+ro2+ro3)
      w3=ro3/(ro1+ro2+ro3)
c
      dVfhdX(inode,1)=a11*(b21+T)*dwdro(inode,1)+a12*(b22+T)*
     +      dwdro(inode,3)+a13*(b23+T)*dwdro(inode,5)
c
      dVfhdX(inode,2)=a11*(b21+T)*dwdro(inode,2)+a12*(b22+T)*
     +      dwdro(inode,4)+a13*(b23+T)*dwdro(inode,6)
c
      dVfhdX(inode,3)=w1*a11+w2*a12+w3*a13
c
      enddo
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine Formdmi2dX2(X,nnodes,DifModel,V1,V2,MM1,MM2,
     +                        dXChidX,XChi,X12,X13,X23,
     +                         dmi2dX2)
c
c*********************************************************************
c*********************************************************************
c
c.....Second derivatives of chemical potencial of each solvent.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,dXChidX(nnodes,9),X12,X13,
     +       X23,XChi(nnodes,3),dmi2dX2(nnodes,12)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 MV1,MV2,Phi1,Phi2,Phip
c
c---------------------------------------------------------------------
c.....Phii: Initial volume fraction of component i [cm3/cm3]
c.....MVi:  Molar volume of component i   [cm3/mol]
c---------------------------------------------------------------------
c
      MV1=MM1*V1
      MV2=MM2*V2
c
      if (DifModel.eq.6) then

      do inode=1,nnodes
c
      Phi1=X(inode,2)*V1
      Phi2=X(nnodes+inode,2)*V2
      Phip=1-Phi1-Phi2
c
      dmi2dX2(inode,1)=-1/(X(inode,2)**2)-2*dXChidX(inode,1)*V1*
     +          (Phi2+Phip)-(dXChidX(inode,7)*Phi2+dXChidX(inode,1)*
     +          Phip-XChi(inode,1)*V1)*V1-V1*(dXChidX(inode,7)*Phi2+
     +          dXChidX(inode,1)*Phip-XChi(inode,1)*V1)+(MV1/MV2)*
     +          Phi2*2*dXChidX(inode,4)*V1
c
      dmi2dX2(inode,2)=(1-Phi1)*(dXChidX(inode,7)*V2-dXChidX(inode,1)*
     +          V2-dXChidX(inode,2)*V1)-V1*(dXChidX(inode,8)*Phi2+
     +              XChi(inode,3)*V2+dXChidX(inode,2)*Phip-XChi(inode,1)*
     +          V2)-(MV1/MV2)*(V2*(dXChidX(inode,4)*Phip-XChi(inode,2)*
     +          V1)+Phi2*(-dXChidX(inode,4)*V2-dXChidX(inode,5)*V1))
c
      dmi2dX2(inode,3)=-dXChidX(inode,3)*V1*(Phi2+Phip)-V1*
     +          (dXChidX(inode,9)*Phi2+dXChidX(inode,3)*Phip)+
     +          (MV1/MV2)*Phi2*dXChidX(inode,6)*V1
c
      dmi2dX2(inode,4)=(1-Phi1)*(dXChidX(inode,7)*V2-dXChidX(inode,2)*
     +         V1-dXChidX(inode,1)*V2)-V1*(dXChidX(inode,8)*Phi2+
     +              XChi(inode,3)*V2+dXChidX(inode,2)*Phip-XChi(inode,1)*
     +          V2)-(MV1/MV2)*(-dXChidX(inode,5)*Phi2*V1+
     +          dXChidX(inode,4)*(V2*Phip-Phi2*V2)-XChi(inode,2)*
     +          V1*V2)

c
      dmi2dX2(inode,5)=(1-Phi1)*(2*dXChidX(inode,8)*V2-2*
     +          dXChidX(inode,2)*V2)-(MV1/MV2)*(dXChidX(inode,5)*
     +          (V2*Phip-Phi2*V2)+dXChidX(inode,5)*(V2*Phip-
     +          Phi2*V2)-2*XChi(inode,2)*(V2**2))
c
      dmi2dX2(inode,6)=(1-Phi1)*(dXChidX(inode,9)*V2-dXChidX(inode,3)*
     +          V2)-(MV1/MV2)*dXChidX(inode,6)*(V2*Phip-Phi2*V2)
c
      dmi2dX2(inode,7)=(1-Phi2)*(2*dXChidX(inode,7)*V1*(MV2/MV1)-2*
     +          dXChidX(inode,4)*V1)-(MV2/MV1)*(dXChidX(inode,1)*
     +          (V1*Phip-Phi1*V1)+dXChidX(inode,1)*(V1*Phip-
     +          Phi1*V1)-2*XChi(inode,1)*(V1**2))
c
      dmi2dX2(inode,8)=(1-Phi2)*(dXChidX(inode,8)*V1*(MV2/MV1)-
     +          dXChidX(inode,4)*V2-dXChidX(inode,5)*V1)-V2*
     +          ((dXChidX(inode,7)*Phi1+XChi(inode,3)*V1)*(MV2/MV1)+
     +          dXChidX(inode,4)*Phip-XChi(inode,2)*
     +          V1)-(MV2/MV1)*(-dXChidX(inode,1)*Phi1*V2+
     +          dXChidX(inode,2)*(V1*Phip-Phi1*V1)-XChi(inode,1)*
     +          V1*V2)
c
      dmi2dX2(inode,9)=(1-Phi2)*(dXChidX(inode,9)*V1*(MV2/MV1)-
     +          dXChidX(inode,6)*V1)-(MV2/MV1)*dXChidX(inode,3)*
     +          (V1*Phip-Phi1*V1)
c
      dmi2dX2(inode,10)=(1-Phi2)*(dXChidX(inode,8)*V1*(MV2/MV1)-
     +      dXChidX(inode,5)*V1-dXChidX(inode,4)*V2)-V2*
     +        (dXChidX(inode,7)*Phi1*(MV2/MV1)+XChi(inode,3)*V1*
     +        (MV2/MV1)+dXChidX(inode,4)*Phip-XChi(inode,2)*
     +          V1)-(MV2/MV1)*(V1*(dXChidX(inode,2)*Phip-XChi(inode,1)*
     +          V2)+Phi1*(-dXChidX(inode,2)*V1-dXChidX(inode,1)*V2))
c
      dmi2dX2(inode,11)=-1/(X(nnodes+inode,2)**2)-2*dXChidX(inode,5)*
     +          V2*(Phi1+Phip)-(dXChidX(inode,8)*Phi1*(MV2/MV1)+
     +          dXChidX(inode,5)*Phip-XChi(inode,2)*V2)*V2-V2*
     +          (dXChidX(inode,7)*Phi1*(MV2/MV1)+dXChidX(inode,5)*
     +          Phip-XChi(inode,2)*V2)+(MV2/MV1)*
     +          Phi1*2*dXChidX(inode,2)*V2
c
      dmi2dX2(inode,12)=-dXChidX(inode,6)*V2*(Phi1+Phip)-V2*
     +          (dXChidX(inode,9)*Phi1*(MV2/MV1)+dXChidX(inode,6)*
     +          Phip)+(MV2/MV1)*Phi1*dXChidX(inode,3)*V2
c
      end do
c
      else
c
      do inode=1,nnodes
c
      Phi1=X(inode,2)*V1
      Phi2=X(nnodes+inode,2)*V2
      Phip=1-Phi1-Phi2
c
      dmi2dX2(inode,1)=-1/(X(inode,2)**2)+2*X13*(V1**2)
```

```
c
      dmi2dX2(inode,2)=-V1*(X12*V2-X13*V2)+(MV1/MV2)*V1*V2*X23
c
      dmi2dX2(inode,3)=0
c
      dmi2dX2(inode,4)=-V1*(X12*V2-X13*V2)+(MV1/MV2)*V1*V2*X23
c
      dmi2dX2(inode,5)=X23*(MV1/MV2)*2*(V2**2)
c
      dmi2dX2(inode,6)=0
c
      dmi2dX2(inode,7)=X13*(MV2/MV1)*2*(V1**2)
c
      dmi2dX2(inode,8)=-V2*(X12*V1*(MV2/MV1)-X23*V1)+(MV2/MV1)*V1*V2*X13
c
      dmi2dX2(inode,9)=0
c
      dmi2dX2(inode,10)=-V2*(X12*V1*(MV2/MV1)-X23*V1)+(MV2/MV1)*V1*V2*
     +                  X13
c
      dmi2dX2(inode,11)=-1/(X(nnodes+inode,2)**2)+2*X23*(V2**2)
c
      dmi2dX2(inode,12)=0
c
      end do
c
      endif
c
      end subroutine



c*********************************************************************
c
      subroutine dAdro(nnodes,A1_1,A1_2,A2_1,A2_2,V1,V2,
     +                dA1dro)
c
c*********************************************************************
c*********************************************************************
c
c.....Calculates derivatives of the parameter alpha model 6-see eq (40)
of
c
c        Price.P.and Romdhane,I.,AIChe Jornal,Vol.49 n°2 (2003)
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 A1_1,A1_2,A2_1,A2_2,V1,V2,
     +       dA1dro(nnodes,4)
c---------------------------------------------------------------------
c
      do inode=1,nnodes
c
      dA1dro(inode,1)=A1_1*V1
      dA1dro(inode,2)=A1_2*V2
      dA1dro(inode,3)=A2_1*V1
      dAldro(inode,4)=A2_2*V2
c
      enddo
c
      end subroutine



c*******************************************************************
c
      subroutine  FormdPdX(nnodes,X,V1,V2,MM1,MM2,X12,X13,X23,A1,B1,C1,
     +                       D1,E1,A2,B2,C2,D2,E2,VP1,VP2,act1,act2,
     +                   dXChidX,XChi,DifModel,
     +                   dPdX)
c
c*******************************************************************
c*******************************************************************
c
c.....Partial Pressure derivatives of Solvents.
c
c     P=act*VP
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X12,X13,X23,A1,B1,C1,D1,E1,
     +       A2,B2,C2,D2,E2,VP1,VP2,act1,act2,dactdro(3,2),dPvdT(2),
     +       dXChidX(nnodes,9),XChi(nnodes,3),dPdX(3,2)
c
c-------------------------------------------------------------------
c
c-------------------------------------------------------------------
c.....Formdactdro calculates derivatives of activity of each solvente
c......at interface.
c-------------------------------------------------------------------
      call Formdactdro (DifModel,nnodes,X,V1,V2,MM1,MM2,X12,X13,
     +                  X23,dXChidX,XChi,
     +                  dactdro)
c
c-------------------------------------------------------------------
c.....FormdPvdT calculates devivatives of vapor pressure of each solvent
c-------------------------------------------------------------------
      call FormdPvdT(nnodes,X,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +                dPvdT)
c
      dPdX(1,1)=VP1*dactdro(1,1)
c
      dPdX(1,2)=VP2*dactdro(1,2)
c
      dPdX(2,1)=VP1*dactdro(2,1)
c
      dPdX(2,2)=VP2*dactdro(2,2)
c
      dPdX(3,1)=act1*dPvdT(1)+dactdro(3,1)*VP1
c
      dPdX(3,2)=act2*dPvdT(2)+dactdro(3,2)*VP2
c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine Formdactdro (DifModel,nnodes,X,V1,V2,MM1,MM2,X12,X13,
     +                        X23,dXChidX,XChi,
     +                        dactdro)
c
c*********************************************************************
c*********************************************************************
c
c.....Derivatives of activity of each solvent at inteface.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,DifModel
      real*8 X(3*nnodes+1,2),V1,V2,MM1,MM2,X12,X13,X23,
     +       dXChidX(nnodes,9),XChi(nnodes,3),
     +       dactdro(3,2)

c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 MV1,MV2,Phi1,Phi2,Phip

c---------------------------------------------------------------------
c     Phii = initial volume fraction of component i [cm3/cm3]
c     MVi = molar volume of component i   [cm3/mol]
c---------------------------------------------------------------------
c
      dactdro=0
c
      Phi1=X(nnodes,2)*V1
      Phi2=X(2*nnodes,2)*V2
      Phip=1-Phi1-Phi2
c
      MV1=MM1*V1
      MV2=MM2*V2
c
      if (DifModel.eq.6) then
c
      dactdro(1,1)=V1*exp((1-Phi1)-(MV1/MV2)*Phi2+(XChi(nnodes,3)*Phi2+
     +    XChi(nnodes,1)*Phip)*(Phi2+Phip)-XChi(nnodes,2)*(MV1/MV2)*
     +    Phi2*Phip)+Phi1*(exp((1-Phi1)-(MV1/MV2)*Phi2+(XChi(nnodes,3)*
     +    Phi2+XChi(nnodes,1)*Phip)*(Phi2+Phip)-XChi(nnodes,2)*
     +    (MV1/MV2)*Phi2*Phip))*(-V1+(Phi2*dXChidX(nnodes,7)+Phip*
     +    dXChidX(nnodes,1)-XChi(nnodes,1)*V1)*(Phi2+Phip)-
     +    (XChi(nnodes,3)*Phi2+XChi(nnodes,1)*Phip)*V1-(MV1/MV2)*
     +    Phi2*(dXChidX(nnodes,4)*Phip-XChi(nnodes,2)*V1))
c
      dactdro(1,2)=Phi2*exp((1-Phi2)-(MV2/MV1)*Phi1+(XChi(nnodes,3)*
     +    Phi1*(MV2/MV1)+XChi(nnodes,2)*Phip)*(Phi1+Phip)-
     +    XChi(nnodes,1)*(MV2/MV1)*Phi1*Phip)*(-(MV2/MV1)*V1+(1-Phi2)*
     +    ((dXChidX(nnodes,7)*Phi1+XChi(nnodes,3)*V1)*(MV2/MV1)+
     +    dXChidX(nnodes,4)*Phip-XChi(nnodes,2)*V1)-(MV2/MV1)*
     +    (dXChidX(nnodes,1)*Phi1*Phip+XChi(nnodes,1)*(V1*Phip-
     +    Phi1*V1)))
c
```

```
      dactdro(2,1)=Phi1*exp((1-Phi1)-(MV1/MV2)*Phi2+(XChi(nnodes,3)*
     +     Phi2+XChi(nnodes,1)*Phip)*(Phi2+Phip)-
     +     XChi(nnodes,2)*(MV1/MV2)*Phi2*Phip)*(-(MV1/MV2)*V2+(1-Phi1)*
     +     ((dXChidX(nnodes,8)*Phi2+XChi(nnodes,3)*V2)+
     +     dXChidX(nnodes,2)*Phip-XChi(nnodes,1)*V2)-(MV1/MV2)*
     +     (dXChidX(nnodes,5)*Phi2*Phip+XChi(nnodes,2)*(V2*Phip-
     +     Phi2*V2)))
c
      dactdro(2,2)=V2*exp((1-Phi2)-(MV2/MV1)*Phi1+(XChi(nnodes,3)*Phi1*
     +     (MV2/MV1)+XChi(nnodes,2)*Phip)*(Phi1+Phip)-XChi(nnodes,1)*
     +     (MV2/MV1)*Phi1*Phip)+Phi2*(exp((1-Phi2)-(MV2/MV1)*Phi1+
     +     (XChi(nnodes,3)*Phi1*(MV2/MV1)+XChi(nnodes,2)*Phip)*
     +     (Phi1+Phip)-XChi(nnodes,1)*(MV2/MV1)*Phi1*Phip))*(-V2+
     +     (dXChidX(nnodes,8)*Phi1*(MV2/MV1)+dXChidX(nnodes,5)*Phip-
     +     XChi(nnodes,2)*V2)*(Phi1+Phip)-(XChi(nnodes,3)*Phi1*
     +     (MV2/MV1)+XChi(nnodes,2)*Phip)*V2-(MV2/MV1)*Phi1*
     +     (dXChidX(nnodes,2)*Phip-XChi(nnodes,1)*V2))
c
      dactdro(3,1)=Phi1*exp((1-Phi1)-(MV1/MV2)*Phi2+(XChi(nnodes,3)*
     +      Phi2+XChi(nnodes,1)*Phip)*(Phi2+Phip)-XChi(nnodes,2)*
     +      (MV1/MV2)*Phi2*Phip)*((Phi2+Phip)*(Phi2*
     +      dXChidX(nnodes,9)+Phip*dXChidX(nnodes,3))-(MV1/MV2)*
     +      Phi2*Phip*dXChidX(nnodes,6))
c
      dactdro(3,2)=Phi2*exp((1-Phi2)-(MV2/MV1)*Phi1+(XChi(nnodes,3)*
     +      Phi1*(MV2/MV1)+XChi(nnodes,2)*Phip)*(Phi1+Phip)-
     +      XChi(nnodes,1)*(MV2/MV1)*Phi1*Phip)*((Phi1+Phip)*(Phi1*
     +      (MV2/MV1)*dXChidX(nnodes,9)+Phip*dXChidX(nnodes,6))-
     +      (MV2/MV1)*Phi1*Phip*dXChidX(nnodes,3))
c
      else
c
      dactdro(1,1)=V1*exp((1-Phi1)-(MV1/MV2)*Phi2+((X12*Phi2+X13*Phip)*
     +    (Phi2+Phip))-X23*(MV1/MV2)*Phi2*Phip)+Phi1*(-V1-X13*V1*
     +    (Phi2+Phip)-(X12*Phi2+X13*Phip)*V1+X23*(MV1/MV2)*Phi2*V1)*
     +    exp((1-Phi1)-(MV1/MV2)*Phi2+((X12*Phi2+X13*Phip)*(Phi2+Phip))-
     +    X23*(MV1/MV2)*Phi2*Phip)
c
      dactdro(1,2)=Phi2*exp((1-Phi2)-(MV2/MV1)*Phi1+((X12*Phi1*
     +     (MV2/MV1)+X23*Phip)*(Phi1+Phip))-X13*(MV2/MV1)*Phi1*Phip)*
     +     (-(MV2/MV1)*V1+(1-Phi2)*(X12*V1*(MV2/MV1)-X23*V1)-X13*
     +     (MV2/MV1)*(V1*Phip-Phi1*V1))
c
      dactdro(2,1)=Phi1*exp((1-Phi1)-(MV1/MV2)*Phi2+((X12*Phi2+
     +     X13*Phip)*(Phi2+Phip))-X23*(MV1/MV2)*Phi2*Phip)*(-(MV1/MV2)*
     +     V2+(1-Phi1)*(X12*V2-X13*V2)-X23*(MV1/MV2)*(V2*Phip-Phi2*V2))

c
      dactdro(2,2)=V2*exp((1-Phi2)-(MV2/MV1)*Phi1+((X12*Phi1*(MV2/MV1)+
     +       X23*Phip)*(Phi1+Phip))-X13*(MV2/MV1)*Phi1*Phip)+Phi2*(-V2-
     +     X23*V2*(Phi1+Phip)-(X12*Phi1*(MV2/MV1)+X23*Phip)*V2+X13*
     +     (MV2/MV1)*Phi1*V2)*exp((1-Phi2)-(MV2/MV1)*Phi1+((X12*Phi1*
     +     (MV2/MV1)+X23*Phip)*(Phi1+Phip))-X13*(MV2/MV1)*Phi1*Phip)
c
      endif
c
      endsubroutine
```

```
c*****************************************************************
c
      subroutine FormdPvdT(nnodes,X,A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +                       dPvdT)
c
c*****************************************************************
c*****************************************************************
c
c.....Vapor Pressure derivatives of Solvents.
c
c     VP=10**( A+ B/T + ClogT + DT + ET2 )         [g/cm.s2]
c
c*****************************************************************
c-----------------------------------------------------------------
c.....External & returning variables
c-----------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),A1,B1,C1,D1,E1,A2,B2,C2,D2,E2,
     +       dPvdT(2)
c
c-----------------------------------------------------------------
c.....Local variables
c-----------------------------------------------------------------
      real*8 T

c-----------------------------------------------------------------
c.....T  : Current liquid temperature           [K]
c-----------------------------------------------------------------
      T=X(3*nnodes+1,2)
c-----------------------------------------------------------------
c.....Conversion factor mmHg to g/cm.s2
c-----------------------------------------------------------------
      cf=1333.0
c-----------------------------------------------------------------
c
      dPvdT(1)=cf*log(10.0)*(-B1/(X(3*nnodes+1,2)**2)+C1/
     +    (X(3*nnodes+1,2)*log(10.0))+
     +    D1+2*E1*X(3*nnodes+1,2))*(10**(A1+B1/X(3*nnodes+1,2)+C1*
     +      log10(X(3*nnodes+1,2))+D1*X(3*nnodes+1,2)+
     +    E1*(X(3*nnodes+1,2)**2)))
     +
c
      dPvdT(2)=cf*log(10.0)*(-B2/(X(3*nnodes+1,2)**2)+C2/
     +    (X(3*nnodes+1,2)*log(10.0))+D2+2*E2*X(3*nnodes+1,2))*
     +    (10**(A2+B2/X(3*nnodes+1,2)+C2*
     +      log10(X(3*nnodes+1,2))+D2*X(3*nnodes+1,2)+
     +    E2*(X(3*nnodes+1,2)**2)))
c
       endsubroutine
```

```
c*********************************************************************
c
      subroutine FormdKdT(X,nnodes,Ta,hs,MM1,RoAir,CpAir,MM2,
     +                    dKdT)
c
c*********************************************************************
c*********************************************************************
c
c.....Calculates derivatives of mass transfer coeficients.
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 X(3*nnodes+1,2),Ta,hs,MM1,RoAir,CpAir,MM2,
     +       dKdT(2,1)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      real*8 Tmd
c
c---------------------------------------------------------------------
c.....Tmd:Average temperature between liquid and air  [K]
c---------------------------------------------------------------------
c
c---------------------------------------------------------------------
c.....Constants
c---------------------------------------------------------------------
c.....R: Universal gas constant                    [(g/cm.s2).cm3/mol.K]
c
      R=8.31451E07
c
c.....a: Thermal conductivity of air               [W/cm.K]
c
      a=0.00026
c---------------------------------------------------------------------
c
      Tmd=(X(3*nnodes+1,2)+Ta)/2
c
      DAir= 0.086
c
      dKdT(1,1)=-0.5*hs*MM1*((RoAir*CpAir*DAir/a)**(0.67))/
     +           (RoAir*CpAir*R*(Tmd**2))
c
      dKdT(2,1)=-0.5*hs*MM2*((RoAir*CpAir*DAir/a)**(0.67))/
     +           (RoAir*CpAir*R*(Tmd**2))
c
       endsubroutine
```

```
double precision function dnrm2 ( n, dx, incx)
      integer i, incx, ix, j, n, next
      double precision   dx(*), cutlo, cuthi, hitest, sum, xmax,zero,one
      data   zero, one /0.0d0, 1.0d0/
c
c     euclidean norm of the n-vector stored in dx() with storage
c     increment incx .
c     if    n .le. 0 return with result = 0.
c     if n .ge. 1 then incx must be .ge. 1
c
c           c.l.lawson, 1978 jan 08
c     modified to correct failure to update ix, 1/25/92.
c     modified 3/93 to return if incx .le. 0.
c
c     four phase method     using two built-in constants that are
c     hopefully applicable to all machines.
c         cutlo = maximum of  dsqrt(u/eps)  over all known machines.
c         cuthi = minimum of  dsqrt(v)      over all known machines.
c     where
c         eps = smallest no. such that eps + 1. .gt. 1.
c         u   = smallest positive no.   (underflow limit)
c         v   = largest  no.            (overflow  limit)
c
c     brief outline of algorithm..
c
c     phase 1    scans zero components.
c     move to phase 2 when a component is nonzero and .le. cutlo
c     move to phase 3 when a component is .gt. cutlo
c     move to phase 4 when a component is .ge. cuthi/m
c     where m = n for x() real and m = 2*n for complex.
c
c     values for cutlo and cuthi..
c     from the environmental parameters listed in the imsl converter
c     document the limiting values are as follows..
c     cutlo, s.p.   u/eps = 2**(-102) for  honeywell.  close seconds are
c                   univac and dec at 2**(-103)
c                   thus cutlo = 2**(-51) = 4.44089e-16
c     cuthi, s.p.   v = 2**127 for univac, honeywell, and dec.
c                   thus cuthi = 2**(63.5) = 1.30438e19
c     cutlo, d.p.   u/eps = 2**(-67) for honeywell and dec.
c                   thus cutlo = 2**(-33.5) = 8.23181d-11
c     cuthi, d.p.   same as s.p.  cuthi = 1.30438d19
c     data cutlo, cuthi / 8.232d-11,  1.304d19 /
c     data cutlo, cuthi / 4.441e-16,  1.304e19 /
      data cutlo, cuthi / 8.232d-11,  1.304d19 /
c
      if(n .gt. 0 .and. incx.gt.0) go to 10
         dnrm2  = zero
         go to 300
c
   10 assign 30 to next
      sum = zero
      i = 1
      ix = 1
c                                                 begin main loop
   20    go to next,(30, 50, 70, 110)
   30 if( dabs(dx(i)) .gt. cutlo) go to 85
      assign 50 to next
      xmax = zero
```

```
c
c                        phase 1.  sum is zero
c
   50 if( dx(i) .eq. zero) go to 200
      if( dabs(dx(i)) .gt. cutlo) go to 85
c
c                                prepare for phase 2.
      assign 70 to next
      go to 105
c
c                                prepare for phase 4.
c
  100 continue
      ix = j
      assign 110 to next
      sum = (sum / dx(i)) / dx(i)
  105 xmax = dabs(dx(i))
      go to 115
c
c                   phase 2.  sum is small.
c                             scale to avoid destructive underflow.
c
   70 if( dabs(dx(i)) .gt. cutlo ) go to 75
c
c                     common code for phases 2 and 4.
c                     in phase 4 sum is large.  scale to avoid overflow.
c
  110 if( dabs(dx(i)) .le. xmax ) go to 115
         sum = one + sum * (xmax / dx(i))**2
         xmax = dabs(dx(i))
         go to 200
c
  115 sum = sum + (dx(i)/xmax)**2
      go to 200
c
c
c                  prepare for phase 3.
c
   75 sum = (sum * xmax) * xmax
c
c
c     for real or d.p. set hitest = cuthi/n
c     for complex      set hitest = cuthi/(2*n)
c
   85 hitest = cuthi/float( n )
c
c                   phase 3.  sum is mid-range.  no scaling.
c
      do 95 j = ix,n
      if(dabs(dx(i)) .ge. hitest) go to 100
         sum = sum + dx(i)**2
         i = i + incx
   95 continue
      dnrm2 = dsqrt( sum )
      go to 300
c
  200 continue
      ix = ix + 1
      i = i + incx
      if( ix .le. n ) go to 20
c
c              end of main loop.
c
c              compute square root and adjust for scaling.
c
      dnrm2 = xmax * dsqrt(sum)
  300 continue
      return
      end




c***********************************************************************l
      subroutine Gauss (n, A, b, x, success)

c***********************************************************************

c     Version: Oct. 6, 1994
c     -------
c
c     Purpose:   Solve a system of linear equations using Gauss
elimination
c     -------    with partial pivoting
c                (ref: Golub, G.H., and Van Loan, C.F. (1983)
c                 Matrix Computations, Johns Hopkins University Press,
c                 Baltimore, Maryland, pp 92)
c
c***********************************************************************
c
c     Variables Definition:
c     --------------------
c     IN:
c            n      : dimension of the system of equations
c            A      : n x n matrix of system coefficients
c            b      : vector containing right-hand side of equations
c
c     OUT:
c            x      : solution vector
c            success: true (if solution is found) or false (if not)
c
c     LOCAL:
c            pivot  : pivot from partial pivoting
c            factor : auxiliar variable
c            sum    : auxiliar variable
c            aaux, baux: auxiliar variables used for swapping rows
c
c***********************************************************************
      implicit double precision (a-h,o-z)

c     Variables:
c     ---------
      integer n
      logical success
      real*8 A(n,n), b(n), x(n)
      real*8 pivot, factor, sum, aaux, baux

c***********************************************************************
```

```
      success = .true.

      do 5 i = 1,n
        x(i) = 0.
  5   continue

c-----loop over rows

      do 100 i = 1, n

c-----find pivot

        pivot = abs(A(i,i))
        irow = i
        do 10 k = i+1, n
          if (abs(A(k,i)) .gt. pivot) then
            pivot = abs(A(k,i))
            irow = k
          endif
  10    continue

c-----check pivot


        if (pivot .lt. 1.d-15 ) then
          success = .false.
          goto 1000
        endif

c-----swap rows

        do 20 j = i, n
          aaux = A(irow, j)
          A(irow, j) = A(i,j)
          A(i,j) = aaux
  20    continue

        baux = b(irow)
        b(irow) = b(i)
        b(i) = baux

c-----eliminate elements below row i

        do 80 k = i+1, n
          factor = A(k,i)/A(i,i)
          do 70 j = i, n
            A(k,j) = A(k,j) - A(i,j) * factor
  70      continue

          b(k) = b(k) - b(i) * factor
  80    continue

  100 continue

c-----backsubstitution

      do 200 i = n, 1, -1
        sum = 0.
        do 150 j = i+1, n
```

```
         sum = sum + A(i,j) * x(j)
 150   continue

       x(i) = (b(i) - sum) / A(i,i)

 200 continue

1000 return

     end




c*********************************************************************
c
      subroutine Solver(nnodes,J,R,
     +                  X)
c
c*********************************************************************
c*********************************************************************
c
c.....Sets the next guessing to current time step
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes
      real*8 J(3*nnodes+1,3*nnodes+1),R(3*nnodes+1),
     +       X(3*nnodes+1,2)
c
c---------------------------------------------------------------------
c.....Local variables
c---------------------------------------------------------------------
      logical success
      real*8 DeltaX(3*nnodes+1)
c---------------------------------------------------------------------
      DeltaX=0

c---------------------------------------------------------------------
c.....Calculation of DeltaX
c---------------------------------------------------------------------
      call gauss (3*nnodes+1,J,-R,
     +             DeltaX,success)
c
c---------------------------------------------------------------------
c----------------------------------------------------------------------
c.....Storage of Newton step solution
c----------------------------------------------------------------------
c
      do inode=1,3*nnodes+1
c
      X(inode,2)=X(inode,2)+DeltaX(inode)
c
      enddo
c
      endsubroutine
```

```
c*******************************************************************
c
      subroutine StoreSolution(nnodes,itime,ttnodes,X,
     +                         C,Tbb,SolRes)
c
c*******************************************************************
c*******************************************************************
c
c.....Store the solution of current time step
c
c*******************************************************************
c-------------------------------------------------------------------
c.....External & returning variables
c-------------------------------------------------------------------
      integer nnodes,itime,ttnodes
      real*8 X(3*nnodes+1,2),
     +       C(3*nnodes+1,ttnodes),Tbb(ttnodes),SolRes(ttnodes,2)
c
c-------------------------------------------------------------------
c
      do inode=1,3*nnodes+1
c
      C(inode,itime)=X(inode,2)
c
      enddo
c
c-------------------------------------------------------------------

c.....SolvResid calculates residual solvent each time step
c-------------------------------------------------------------------

      call SolvResid(itime,nnodes,ttnodes,C,
     +              SolRes)
c-------------------------------------------------------------------
c
      write (*,*) "Time step:",itime
      write (*,*) "Concentration of Specie 1 at base:",C(1,itime)
      write (*,*) "Concentration of Specie 2 at base:",C(nnodes+1,itime)
      write (*,*) "    "
      write (*,*) "Free surface position:",C(3*nnodes,itime)
      write (*,*) "    "
      write (*,*) "Temperature:",C(3*nnodes+1,itime)
      write (*,*) "Solution bubble point temp.:",Tbb(itime)
      write (*,*) "    "
      write (*,*) "Residual Solv.1:",SolRes(itime,1)
      write (*,*) "Residual Solv.2:",SolRes(itime,2)
      write (*,*) "    "

c
      endsubroutine
```

```
c*********************************************************************
c
      subroutine SolvResid(itime,nnodes,ttnodes,C,
     +                     SolRes)
c
c*********************************************************************
c
c....SolRes calculates de residual solvent for each time step / solvent
c                              [g/cm²]
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer itime,nnodes,ttnodes
      real*8 C(3*nnodes+1,ttnodes),SolRes(ttnodes,2)
c
c---------------------------------------------------------------------
c
      SolRes(itime,1)=0
      SolRes(itime,2)=0
c
      do inode=1,nnodes-1
c
      SolRes(itime,1)=SolRes(itime,1)+0.5*(C(inode+1,itime)+
     +              C(inode,itime))*(C(2*nnodes+inode+1,itime)-
     +              C(2*nnodes+inode,itime))
c
      SolRes(itime,2)= SolRes(itime,2)+0.5*(C(nnodes+inode+1,itime)+
     +              C(nnodes+inode,itime))*(C(2*nnodes+inode+1,itime)-
     +              C(2*nnodes+inode,itime))
c
      enddo
c
      endsubroutine




c*********************************************************************
c
      subroutine PostPro(C,nnodes,ttnodes,tm,Tbb,SolRes,Tbbmin,Ta1s,
     +                   Ta1i)
c
c*********************************************************************
c*********************************************************************
c
c.....Prepare data to generate reports and graphics
c
c*********************************************************************
c---------------------------------------------------------------------
c.....External & returning variables
c---------------------------------------------------------------------
      integer nnodes,ttnodes
      real*8 C(3*nnodes+1,ttnodes),Tbb(ttnodes),Tm(ttnodes),
     +       SolRes(ttnodes,2),Tbbmin(2),Ta1s,Ta1i
c
c---------------------------------------------------------------------
```

```
c.....Local variables
c-----------------------------------------------------------------------
      integer itime
      real*8 SolResTotal(ttnodes),Tk(ttnodes),FT(ttnodes),
     +       C1base(ttnodes),C2base(ttnodes),
     +       C1middle(ttnodes),C2middle(ttnodes),C1top(ttnodes),
     +       C2top(ttnodes),grad1ini(nnodes),grad2ini(nnodes),
     +       grad1fin(nnodes),grad2fin(nnodes),grad1mid(nnodes),
     +       grad2mid(nnodes),Ta1Max,Zini(nnodes),Zmid(nnodes),
     +       Zfin(nnodes)
c-----------------------------------------------------------------------
c
      open(UNIT=2,FILE='Data.xls',STATUS='REPLACE')
      open(UNIT=3,FILE='Data1.xls',STATUS='REPLACE')
      open(UNIT=4,FILE='Data2.xls',STATUS='REPLACE')
      open(UNIT=5,FILE='Data3.xls',STATUS='REPLACE')
c-----------------------------------------------------------------------
c......Determination of maximum temperature on zone 1
c-----------------------------------------------------------------------
      if (Ta1s.gt.Ta1i) then
c
      Ta1Max=Ta1s
c
      else
c
      Ta1Max=Ta1i
c
      endif
c
c-----------------------------------------------------------------------
c......Calculation of total residual solvent        [g/cm²]
c-----------------------------------------------------------------------
      do itime=1,ttnodes
c
      SolResTotal(itime)=SolRes(itime,1)+SolRes(itime,2)
c
      enddo
c-----------------------------------------------------------------------
c......Converting K to C
c-----------------------------------------------------------------------
      do itime=1,ttnodes
c
      Tbb(itime)=Tbb(itime)-273.0
      C(3*nnodes+1,itime)=C(3*nnodes+1,itime)-273.0
c
      enddo
c
      Tbbmin(1)=Tbbmin(1)-273.0
      Tbbmin(2)=Tbbmin(2)-273.0
      Ta1Max=Ta1Max-273.0
c-----------------------------------------------------------------------
c......Creating vectors temperature,thickness & concentration
c-----------------------------------------------------------------------
      do itime=1,ttnodes
c
      FT(itime)=C(3*nnodes+1,itime)
      Tk(itime)=C(3*nnodes,itime)
      C1base(itime)=C(1,itime)
      C2base(itime)=C(nnodes+1,itime)
      C1middle(itime)=C(nnodes*0.5,itime)
      C2middle(itime)=C(1.5*nnodes,itime)
      C1top(itime)=C(nnodes,itime)
      C2top(itime)=C(2*nnodes,itime)
c
      enddo
c----------------------------------------------------------------------
c......Creating vector concentration gradient
c----------------------------------------------------------------------
      do inode=1,nnodes
c
      grad1ini(inode)=C(inode,10)
      grad2ini(inode)=C(nnodes+inode,10)
      grad1mid(inode)=C(inode,ttnodes*0.5)
      grad2mid(inode)=C(nnodes+inode,ttnodes*0.5)
      grad1fin(inode)=C(inode,ttnodes-10)
      grad2fin(inode)=C(nnodes+inode,ttnodes-10)
c
      Zini(inode)=C(2*nnodes+inode,10)
      Zmid(inode)=C(2*nnodes+inode,ttnodes*0.5)
      Zfin(inode)=C(2*nnodes+inode,ttnodes-10)
c
      enddo
c----------------------------------------------------------------------
c......Send data to output file
c----------------------------------------------------------------------
c
      write (2,*) "Time[s]"," ","ResidSolv1[g/cm²]",
     +         " ","ResidSolv2[g/cm²]"," ","ResidSolvTT[g/cm²]"
c
      do itime=1,ttnodes
c
      write (2,10) Tm(itime),SolRes(itime,1),SolRes(itime,2),
     +      SolResTotal(itime)
   10 format(F6.1,F12.6,F12.6,F12.6)
c
      enddo
c----------------------------------------------------------------------
      write (3,*) "Time[s]"," ","SolutionBubbleTemp[C]"," ",
     +          "FilmThickness[cm]"," ",
     +          "FilmTemp[C]"," ","BoilTempSol1[C]"," ",
     +          "BoilTempSol2[C]"," ","MaxTempZone1[C]"
c
      do itime=1,ttnodes
c
      write(3,20)Tm(itime),Tbb(itime),Tk(itime),
     +           FT(itime),Tbbmin(1),Tbbmin(2),Ta1Max
   20 format(F6.1,F7.1,F12.6,F6.1,F7.1,F7.1,F7.1)
c
      enddo
c
c----------------------------------------------------------------------
c
      write (4,*) "Time[s]"," ","Solv1-base[g/cm3]"," ",
     +          "Solv1-middle[g/cm3]"," ","Solv1-top[g/cm3]"," ",
     +          "Solv2-base[g/cm3]"," ","Solv2-middle[g/cm3]"," ",
     +          "Solv2-top[g/cm3]"
c
      do itime=1,ttnodes
c
```

```
      write(4,30)Tm(itime),C1base(itime),C1middle(itime),C1top(itime),
     +           C2base(itime),C2middle(itime),C2top(itime)
   30 format(F6.1,F12.6,F12.6,F12.6,F12.6,F12.6,F12.6)
c
      enddo
c
c-----------------------------------------------------------------------
c
      write (5,*) "Z-initial[cm]"," ","Solv1-Initial[g/cm3]"," ",
     +          "Solv2-Initial[g/cm3]"," ","Z-mean[cm]"," ",
     +          "Solv1-Mean[g/cm3]"," ","Solv2-Mean[g/cm3]"," ",
     +          "Z-final[cm]"," ","Solv1-Final[g/cm3]"," ",
     +          "Solv2-Final[g/cm3]"
c
      do inode=1,nnodes
c
      write(5,40) Zini(inode),grad1ini(inode),grad2ini(inode),
     +            Zmid(inode),grad1mid(inode),grad2mid(inode),
     +            Zfin(inode),grad1fin(inode),grad2fin(inode)
   40 format(9(F12.6))

c
      enddo
c
c-----------------------------------------------------------------------
c
      pause
      pause
c
      endsubroutine
```