รายการอ้างอิง

[1] Thornton, E. A. TAP 1: A Finite Element Program for Steady-State Thermal Analysis of Convectively Cooled Structures. NASA Contractor Report 145069 (1976).

[2] Schmidt, F. W. and Szego, J. Transient Response of Solid Sensible Heat Thermal Storage Units-Single Fluid. Journal of Heat Transfer 98 (1976): 471-477.

[3] Szego, J. and Schmidt, F. W. Transient Behavior of a Solid Sensible Heat Thermal Storage Exchanger. Journal of Heat Transfer 100 (1978): 148-154.

[4] Thornton, E. A. TAP 2: A Finite Element Program for Thermal Analysis of Convectively Cooled Structures. NASA Contractor Report 159038 (1980).

[5] Kawahara, M., Yokouchi, Y., Tamano, T. and Ohtsubo, H. Steady and Unsteady Finite Element Analysis of Incompressible Viscous Fluid. International Journal of Numerical Methods in Engineering 10 (1976): 437-456.

[6] Yamada, Y., Ito, K., Yokouchi, Y., Tamano, T. and Ohtsubo, T. Finite Element Analysis of Steady Fluid and Metal Flow. Finite Element Methods in Fluids 1 (1975).

[7] ปราโมทย์ เดชะอำไพ. ไฟไนต์เอลิเมนต์ในงานวิศวกรรม. พิมพ์ครั้งที่ 3. กรุงเทพฯ: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2547.

[8] Dechaumphai, P. Adaptive Finite Element Technique for Heat Transfer Problems. Journal of Energy, Heat and Mass Transfer 17 (1995): 87-94.

[9] Dechaumphai, P. Adaptive Finite Element Technique for Heat Thermal Stress Analysis of Built-Up Structures. JSME International Journal 39 (1996): 223-230.

[10] ยศกร ประทุมวัลย์. ระเบียบไฟไนต์วอลลุมเพื่อการวิเคราะห์การถ่ายเทความร้อนแบบคอนจูเกต. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย 2545.

[11] อธิพงษ์ มาลาทิพย์. ระเบียบวิธีไฟไนต์เอลิเมนต์เพื่อการวิเคราะห์การถ่ายเทความร้อนแบบคอนจูเกต. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย 2547.

[12] Phongthanapanich, S., Traivivatana, S., Boonmaruth, P. and Dechaumphai, P. Nodeless Variable Finite Element Method for Heat Transfer Analysis by Means of Flux-Based Formulation and Mesh Adaptation. Acta Mechanica Sinica 22 (2006): 138-147.

[13] Phongthanapanich, S. and Dechaumphai, P. Nodeless Variable Finite Element Method for Stress Analysis Using Flux-Based Formulation. Journal of Mechanical Science and Technology 22 (2008): 639-646.

[14] Anderson, J. D. Jr. Computational Fluid Dynamics, The Basics with Applications. International Edition. Singapore: McGraw-Hill, 1995.

[15] ปราโมทย์ เดชะอำไพ. ระเบียบวิธีไฟไนต์เอลิเมนต์เพื่อการคำนวณพลศาสตร์ของไหล. พิมพ์ครั้งที่ 1. กรุงเทพฯ: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545.

[16] Christe, I., Griffiths, D. F., Mitchell, A. R. and Zienkiewicz, O. C., Finite Element Methods for Second Order Differential Equations with Significant First Derivatives. International Journal for Numerical Methods in Engineering 10 (1976): 1389-1396.

[17] Kythe, P. K. and Wei, D. An introduction to linear and nonlinear finite element analysis, A Computational Approach. Boston: Birkauser Boston,2004.

[18] ปราโมทย์ เดชะอำไพ และ สุทธิศักดิ์ พงษ์ธนาพานิช, ไฟไนต์เอลิเมนต์อย่างง่ายพร้อมซอฟแวร์, พิมพ์ครั้งที่ 1. กรุงเทพฯ: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2548.

[19] Poulikakos, D. Conduction Heat Transfer., New Jersey : Prentice-Hall, 1994.

[20] Kays, W. M, and Crawford, M. E. Convective Heat and Mass Transfer. Third Edition. Singapore: McGraw-Hill, 1993.

[21] Huebner, K. H., Thornton, E. A. and Byrom, T. G. The Finite Element Method for Engineers. Third Edition. New York: John Wiley & Sons, 1995.

[22] Parnes, R. Solid Mechanics in Engineering. Third Edition. West Sussex: John Wiley & Sons, 2001.

[23] สุทธิศักดิ์ พงศ์ธนาพานิช. การพัฒนาโปรแกรมคอมพิวเตอร์สำหรับการไหลไม่คงตัวความเร็วสูงแบบอัดตัวได้และไร้ความหนืดในสองมิติด้วยระเบียบวิธีไฟไนต์เอลิเมนต์. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย 2544.

[24] Phongthanapanich, S. and Dechaumphai, P. Evaluation of Combined Delaunay Triangulation and Adaptive Finite Elements for Heat Transfer Problems. Transactions of Canadian Society for Mechanical Engineering. 4, 27 (2003).

[25] McGowan, D. M., Camarda, C. J. and Scotti, S. J. A Simplified Method of Thermal Analysis of a Cowl Leading Edge Subjected to Intense Localized Heating. NASA TP 16505 (1990).

ภาคผนวก

ภาคผนวก ก

รายละเอียดของสมการไฟไนต์เอลิเมนต์เมตริกซ์

รายละเอียดของไฟไนต์เอลิเมนต์เมทริกซ์สำหรับของไหลต่าง ๆ

- เมทริกซ์ความจุความร้อน (capacitance matrix)

จาก
$$[C]=\int_0^L \rho_f c_f A_f \{N\}[N]dz \tag{3.42}$$

$$=\frac{\rho_f c_f A_f L}{30}\begin{bmatrix}10 & 5 & 10\\ 5 & 10 & 10\\ 10 & 10 & 16\end{bmatrix}$$

- เมทริกซ์การพามวลของไหล (mass transport fluid convection matrix)

จาก
$$[K_v]=\int_0^L \dot{m}_f c_f \{N\}\left[\frac{\partial N}{\partial z}\right]dz \tag{3.43}$$

$$=\frac{\dot{m}_f c_f}{6}\begin{bmatrix}-3 & 3 & -4\\ -3 & 3 & -4\\ -4 & 4 & 0\end{bmatrix}$$

- เมทริกซ์การนำความร้อน (conduction matrix)

จาก
$$[K_c]=\int_0^L k_f A_f \left\{\frac{\partial N}{\partial z}\right\}\left[\frac{\partial N}{\partial z}\right]dz \tag{3.44}$$

$$=\frac{k_f A_f}{3L}\begin{bmatrix}3 & -3 & 0\\ -3 & 3 & 0\\ 0 & 0 & 16\end{bmatrix}$$

- เมทริกซ์การพาความร้อนระหว่างของไหลกับของแข็ง (fluid-solid convection matrix)

จาก
$$\left[K_{f-s}\right]=\int_0^L hp\{N\}\lfloor N\rfloor dz \tag{3.45}$$

$$=\frac{hpL}{30}\begin{bmatrix}10 & 5 & 10\\ 5 & 10 & 10\\ 10 & 10 & 16\end{bmatrix}$$

- โหลดเวกเตอร์ของปริมาณความร้อนที่ผลิตได้เอง (internal heat generation load vector)

จาก
$$\left\{Q_Q\right\}=\int_0^L Q_f A_f \{N\}\, dz \tag{3.46}$$

$$=\frac{Q_f A_f L}{6}\begin{Bmatrix}3\\ 3\\ 4\end{Bmatrix}$$

รายละเอียดของไฟไนต์เอลิเมนต์เมทริกซ์สำหรับของแข็งต่าง ๆ

- เมทริกซ์ความจุความร้อน (capacitance matrix)

จาก
$$[C]=\int_{A_s}\rho_s c_s t_s \{N\}\lfloor N\rfloor dA_s \tag{3.63}$$

$$=\frac{\rho_s c_s t_s}{180}\begin{bmatrix}30 & 15 & 15 & 12 & 24 & 24\\ 15 & 30 & 15 & 24 & 12 & 24\\ 15 & 15 & 30 & 24 & 24 & 12\\ 12 & 24 & 24 & 32 & 16 & 16\\ 24 & 12 & 24 & 16 & 32 & 16\\ 24 & 24 & 12 & 16 & 16 & 32\end{bmatrix}$$

- เมทริกซ์การนำความร้อน (conduction matrix)

จาก
$$[K_c]=\int_{A_s} k_s t_s\left(\left\{\frac{\partial N}{\partial x}\right\}\left\lfloor\frac{\partial N}{\partial x}\right\rfloor+\left\{\frac{\partial N}{\partial y}\right\}\left\lfloor\frac{\partial N}{\partial y}\right\rfloor\right)dA_s \tag{3.64}$$

$$=\frac{k_s t_s}{A_s}\begin{bmatrix}k_{11} & k_{12} & k_{13} & k_{14} & k_{15} & k_{16}\\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} & k_{26}\\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} & k_{36}\\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} & k_{46}\\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} & k_{56}\\ k_{61} & k_{62} & k_{63} & k_{64} & k_{65} & k_{66}\end{bmatrix}$$

โดย
$$k_{11}=\frac{1}{4}\left(b_1^2+c_1^2\right)$$

$$k_{12}=\frac{1}{4}\left(b_1b_2+c_1c_2\right)$$

$$k_{13}=\frac{1}{4}\left(b_1b_3+c_1c_3\right)$$

$$k_{14}=\frac{1}{3}\left[b_1\left(b_2+b_3\right)+c_1\left(c_2+c_3\right)\right]$$

$$k_{15} = \frac{1}{3}\left[b_1\left(b_1 + b_3\right) + c_1\left(c_1 + c_3\right)\right]$$

$$k_{16} = \frac{1}{3}\left[b_1\left(b_1 + b_2\right) + c_1\left(c_1 + c_2\right)\right]$$

$$k_{22} = \frac{1}{4}\left(b_2^2 + c_2^2\right)$$

$$k_{23} = \frac{1}{4}\left(b_2 b_3 + c_2 c_3\right)$$

$$k_{24} = \frac{1}{3}\left[b_2\left(b_2 + b_3\right) + c_2\left(c_2 + c_3\right)\right]$$

$$k_{25} = \frac{1}{3}\left[b_2\left(b_1 + b_3\right) + c_2\left(c_1 + c_3\right)\right]$$

$$k_{26} = \frac{1}{3}\left[b_2\left(b_1 + b_2\right) + c_2\left(c_1 + c_2\right)\right]$$

$$k_{33} = \frac{1}{4}\left(b_3^2 + c_3^2\right)$$

$$k_{34} = \frac{1}{3}\left[b_3\left(b_2 + b_3\right) + c_3\left(c_2 + c_3\right)\right]$$

$$k_{35} = \frac{1}{3}\left[b_3\left(b_1 + b_3\right) + c_3\left(c_1 + c_3\right)\right]$$

$$k_{36} = \frac{1}{3}\left[b_3\left(b_1 + b_2\right) + c_3\left(c_1 + c_2\right)\right]$$

$$k_{44} = \frac{2}{3}\left[\left(b_2^2 + b_2 b_3 + b_3^2\right) + \left(c_2^2 + c_2 c_3 + c_3^2\right)\right]$$

$$k_{45} = \frac{1}{3}\left[\left(b_3^2 + b_1 b_3 + b_2 b_3 + 2b_1 b_2\right) + \left(c_3^2 + c_1 c_3 + c_2 c_3 + 2c_1 c_2\right)\right]$$

$$k_{46} = \frac{1}{3}\left[\left(b_2^2 + b_1 b_2 + b_2 b_3 + 2b_1 b_3\right) + \left(c_2^2 + c_1 c_2 + c_2 c_3 + 2c_1 c_3\right)\right]$$

$$k_{55} = \frac{2}{3}\left[\left(b_1^2 + b_1 b_3 + b_3^2\right) + \left(c_1^2 + c_1 c_3 + c_3^2\right)\right]$$

$$k_{56} = \frac{1}{3}\left[\left(b_1^2 + b_1b_2 + b_1b_3 + 2b_2b_3\right) + \left(c_1^2 + c_1c_2 + c_1c_3 + 2c_2c_3\right)\right]$$

$$k_{66} = \frac{2}{3}\left[\left(b_1^2 + b_1b_2 + b_2^2\right) + \left(c_1^2 + c_1c_2 + c_2^2\right)\right]$$

หมายเหตุ: b_1, b_2, b_3, c_1, c_2 และ c_3 หาได้จากสมการ (3.26) และ $k_{ij} = k_{ji}$

- เมทริกซ์การพาความร้อน (convection matrix)

จาก $$\left[K_h\right] = \int_{\Gamma} h_s t_s \{N\}\lfloor N \rfloor d\Gamma \tag{3.65}$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ผิวด้านบนของเอลิเมนต์

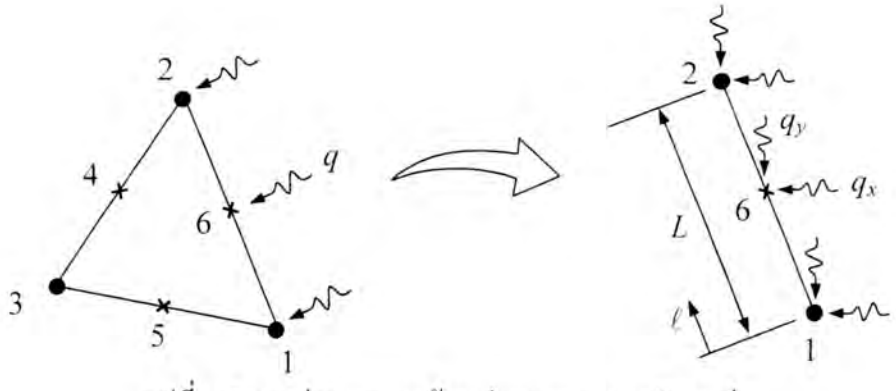
$$\left[K_h\right] = \int_{A_s} h_s t_s \{N\}\lfloor N \rfloor dA_s$$

$$= \frac{h_s A_s}{180}\begin{bmatrix} 30 & 15 & 15 & 12 & 24 & 24 \\ 15 & 30 & 15 & 24 & 12 & 24 \\ 15 & 15 & 30 & 24 & 24 & 12 \\ 12 & 24 & 24 & 32 & 16 & 16 \\ 24 & 12 & 24 & 16 & 32 & 16 \\ 24 & 24 & 12 & 16 & 16 & 32 \end{bmatrix}$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ขอบระหว่างจุดต่อที่ 1 และ 2 ดังแสดงในรูปที่ ก.1 ดังนั้น

$$\left[K_h\right] = \int_{\ell} h_s t_s \{N\}\lfloor N \rfloor d\ell$$

$$= \frac{h_s t_s L}{30} \begin{bmatrix} 10 & 5 & 0 & 0 & 0 & 10 \\ 5 & 10 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 & 0 & 16 \end{bmatrix}$$



รูปที่ ก.1 การถ่ายเทความร้อนผ่านขอบของเอลิเมนต์

- เมทริกซ์การพาความร้อนระหว่างของไหลกับของแข็ง (fluid-solid convection matrix)

จาก $$\left[K_{f-s}\right] = \int_{\Gamma} ht \{N\} \lfloor N \rfloor d\Gamma \qquad (3.66)$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ขอบระหว่างจุดต่อที่ 1 และ 2 ดังแสดงในรูปที่ ก.1 ดังนั้น

$$\left[K_{f-s}\right] = \int_{\ell} ht \{N\} \lfloor N \rfloor d\ell$$

$$= \frac{htL}{30} \begin{bmatrix} 10 & 5 & 0 & 0 & 0 & 10 \\ 5 & 10 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 & 0 & 16 \end{bmatrix}$$

- โหลดเวกเตอร์ของปริมาณความร้อนที่ผลิตได้เอง (internal heat generation load vector)

จาก
$$\{Q_Q\} = \int_{A_s} Q_s t_s \{N\}\, dA_s \tag{3.67}$$

$$= \frac{Q_s t_s A_s}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}$$

- โหลดเวกเตอร์ของปริมาณความร้อนที่กำหนดให้ (specific heat load vector)

จาก
$$\{Q_q\} = \int_{\Gamma} q_s t_s \{N\}\, d\Gamma \tag{3.68}$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ผิวด้านบนของเอลิเมนต์

$$\{Q_q\} = \int_{A_s} q_s t_s \{N\}\, dA_s$$

$$= \frac{q_s A_s}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ขอบระหว่างจุดต่อที่ 1 และ 2 ดังแสดงในรูปที่ ก.1 ดังนั้น

$$\{Q_q\} = \int_{\ell} q_s t_s \{N\}\, d\ell$$

$$= \frac{q_s t_s L}{6} \begin{Bmatrix} 3 \\ 3 \\ 0 \\ 0 \\ 0 \\ 4 \end{Bmatrix}$$

- โหลดเวกเตอร์ของปริมาณความร้อนจากการพาความร้อน (convection heat load vector)

จาก $$\{Q_h\} = \int_{\Gamma} h_s t_s T_\infty \{N\} d\Gamma \qquad (3.69)$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ผิวด้านบนของเอลิเมนต์

$$\{Q_h\} = \int_{A_s} h_s t_s T_\infty \{N\} dA_s$$

$$= \frac{h_s T_\infty A_s}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}$$

หากการถ่ายเทความร้อนเกิดขึ้นที่ขอบระหว่างจุดต่อที่ 1 และ 2 ดังแสดงในรูปที่ ก.1 ดังนั้น

$$\{Q_h\} = \int_{\ell} h_s t_s T_\infty \{N\} d\ell$$

$$= \frac{h_s T_\infty t_s L}{6} \begin{Bmatrix} 3 \\ 3 \\ 0 \\ 0 \\ 0 \\ 4 \end{Bmatrix}$$

ภาคผนวก ข

รายละเอียดของโปรแกรมคอมพิวเตอร์ Nodeless FE

โปรแกรมคอมพิวเตอร์ Nodeless FE ที่ได้ประดิษฐ์ขึ้นเพื่อการวิเคราะห์อุณหภูมิของโครงสร้างหล่อเย็นด้วยการพาความร้อนที่ได้กล่าวไว้ในบทที่ 4 มีรายละเอียดดังนี้

```
MODULE HF
IMPLICIT NONE
CHARACTER(len=20) name1
interger(4)     ::iline,nlines,i,j,k,ii,jj,kk,ie,num,step,step2
interger (4)    ::ana,FE,sol,nsave,neleF,neleS,neleFS,npoin,onpoin
real(8)         ::fac,zeta,dt,time,rtime
real (8)        ::condF,specF,densF,areaF,flowF,QgenF,QspecF,convF,periF,TsurrF
real (8)        ::condS,specS,densS,thick,QgenS,QspecS,convS,TsurrS
interger (4), allocatable, dimension(:)      ::ibc,ibc2
interger (4), allocatable, dimension (:,:)   ::intmatF,QtypeF,intmatS
interger (4), allocatable, dimension (:,:)   :: QtypeS,intmatFS,edge
interger (4), allocatable, dimension (:,:,:) ::acheck
real (8), allocatable, dimension (:)   ::convFS,temp,temp2,sysQ,sysQ2,sysR
real (8), allocatable, dimension (:,:) ::coord,sysK,sysC,sysK1
real (8), dimension (20)               ::text
CONTAINS
!
!-------------------------------------------------------------------------
!
SUBROUTINE MAIN()
IMPLICIT NONE
call read_input()
allocate(sysK(npoin,npoin),sysQ(npoin))
sysK=0.
sysQ=0.
if (ana==1)then
        allocate (sysC(npoin,npoin))
```

```
            sysC=0.
 endif
 if (neleF>0) call setmatF()
 if (neleS>0) call setmatS()
 if (neleFS>0) call setmatFS()
 step=0
 step2=0
 rtime=0.0
 10      continue
            step=step+1
            step2=step2+1
            rtime=rtime+dt
 call manmat()
 call solver()
 if(ana==1)then
            write(6,15) step,rtime
            15 format(i10,F30.8,F30.13)
 endif
 if (ana==0) goto 10000
 if (rtime>=time) goto 10000
 call printtxt()
 goto 10
 10000 continue
 call printtxt()
 END SUBROUTINE main
 !
 !-------------------------------------------------------------------------
 !
 SUBROUTINE READ_INPUT()
 IMPLICIT NONE
 write(6,5)
 5 format(/,'PLEASE ENTER THE INPUT FILE NAME:')
 read(5,*) name1
 open(unit=7,file=name1,status='old')
 ! read title of computation
 read(7,*) nlines
 do iline=1,nlines
```

```
        read(7,10)text
enddo
!
10 FORMAT(20A4)
read(7,10)text
READ(7,*) Ana,FE,Sol
READ(7,10) text
READ(7,*) Zeta,dt,time,nsave
READ(7,10) text
READ(7,10) text
READ(7,*) condF,specF,densF
READ(7,10) text
READ(7,*) areaF,flowF,QgenF,QspecF,convF,periF,TsurrF
READ(7,10) text
READ(7,*) neleF
READ(7,10) text
num=2
if(FE==1)num=3
allocate(intmatF(neleF,num),QtypeF(neleF,3))
do ie=1,neleF
        READ(7,*) i,(intmatF(i,j),j=1,2),(QtypeF(i,j),j=1,3)
enddo
READ(7,10) text
READ(7,10) text
READ(7,*) condS,specS,densS
READ(7,10) text
READ(7,*) Thick,QgenS,QspecS,convS,TsurrS
READ(7,10) text
READ(7,*) neleS
READ(7,10) text
num=3
if(FE==1)num=6
allocate(intmatS(neleS,num),QtypeS(neleS,3))
do ie=1,neleS
        READ(7,*) i,(intmatS(i,j),j=1,3),(QtypeS(i,j),j=1,3)
enddo
READ(7,10) text
```

```
READ(7,10) text
READ(7,*) neleFS
READ(7,10) text
num=4
if(FE==1)num=6
allocate(intmatFS(neleFS,num),convFS(neleFS))
do ie=1,neleFS
        READ(7,*) i,(intmatFS(i,j),j=1,4),convFS(i)
enddo
READ(7,10) text
READ(7,10) text
READ(7,*) npoin
READ(7,10) text
allocate(coord(npoin,2),ibc2(npoin),temp2(npoin))
do i=1,npoin
        READ(7,*) ii,(coord(ii,j),j=1,2),ibc2(ii),temp2(ii)
enddo
onpoin=npoin
if (FE==1) then
        if(neleF>0)call CRE_NOD_F()
        if(neleS>0)call CRE_NOD_S()
endif
ALLOCATE (temp(npoin),ibc(npoin))
temp=0.
ibc=0
do i=1,onpoin
        temp(i)=temp2(i)
        ibc(i)=ibc2(i)
enddo
deallocate(ibc2,temp2)
if (FE==1)then
        call setBC()
        if(neleS>0)deallocate(edge)
endif
RETURN
END SUBROUTINE READ_INPUT
!
```

```
!-------------------------------------------------------------------------
!
SUBROUTINE CRE_NOD_F()
IMPLICIT NONE
INTEGER(4),ALLOCATABLE,DIMENSION(:,:)  ::rod
allocate(rod(onpoin,onpoin))
rod=0
DO  ie=1,neleF
        i=intmatF(ie,1)
        j=intmatF(ie,2)
        if(rod(i,j)==0)then
                NPOIN=NPOIN+1
                intmatF(ie,3)=NPOIN
                rod(i,j)=npoin
        endif
END DO
!       set node interaction with fluid
DO  ie=1,neleFS
        ii=intmatFS(ie,1)
        jj=intmatFS(ie,2)
        IF (rod(ii,jj)>0) intmatFS(ie,5)=rod(ii,jj)
END DO
deallocate(rod)
RETURN
END SUBROUTINE CRE_NOD_F
!
!-------------------------------------------------------------------------
!
SUBROUTINE CRE_NOD_S()
IMPLICIT NONE
ALLOCATE (acheck(onpoin,onpoin,2),edge(onpoin,onpoin))
DO i=1,onpoin
DO j=1,onpoin
        edge(i,j)=0
        DO k=1,2
                acheck(i,j,k)=0
        ENDDO
```

```
ENDDO
ENDDO
element:DO ie=1,NELES
        ii=intmatS(ie,1)
        jj=intmatS(ie,2)
        kk=intmatS(ie,3)
!       CREATE NODELESS AT EDAGE JJ-KK
        acheck(jj,kk,2)=acheck(jj,kk,2)+1
        acheck(kk,jj,2)=acheck(kk,jj,2)+1
        IF (acheck(jj,kk,1)==1) GOTO 140
        acheck(jj,kk,1)=1
        acheck(kk,jj,1)=1
        NPOIN=NPOIN+1
        intmatS(IE,4)=NPOIN
        edge(jj,kk)=NPOIN
        edge(kk,jj)=NPOIN
        GOTO 150
        140 CONTINUE
        intmatS(ie,4)=edge(jj,kk)
        150 CONTINUE
!       CREATE NODELESS AT EDAGE II-KK
        acheck(ii,kk,2)=acheck(ii,kk,2)+1
        acheck(kk,ii,2)=acheck(kk,ii,2)+1
        IF (acheck(ii,kk,1)==1) GOTO 160
        acheck(ii,kk,1)=1
        acheck(kk,ii,1)=1
        NPOIN=NPOIN+1
        intmatS(ie,5)=NPOIN
        edge(ii,kk)=NPOIN
        edge(kk,ii)=NPOIN
        GOTO 170
        160     CONTINUE
        intmatS(ie,5)=edge(ii,kk)
        170     CONTINUE
!       CREATE NODELESS AT EDAGE II-JJ
        acheck(ii,jj,2)=acheck(ii,jj,2)+1
        acheck(jj,ii,2)=acheck(jj,ii,2)+1
```

```
            IF (acheck(ii,jj,1)==1) GOTO 180
            acheck(ii,jj,1)=1
            acheck(jj,ii,1)=1
            NPOIN=NPOIN+1
            intmatS(IE,6)=NPOIN
            edge(ii,jj)=NPOIN
            edge(jj,ii)=NPOIN
            GOTO 190
            180    CONTINUE
            intmatS(ie,6)=edge(ii,jj)
            190    CONTINUE
END DO element
!       set node interaction with fluid
DO  ie=1,NELEF
            ii=intmatFS(ie,3)
            jj=intmatFS(ie,4)
            IF (edge(ii,jj)>0) intmatFS(ie,6)=edge(ii,jj)
END DO
RETURN
END SUBROUTINE CRE_NOD_S
!
!-------------------------------------------------------------------------------
!
SUBROUTINE setBC()
IMPLICIT NONE
! set fluid nodeless BC
if (neleF>0) then
do  ie=1,neleF
            i=intmatF(ie,1)
            j=intmatF(ie,2)
            k=intmatF(ie,3)
            if (ibc(i)==1.and.ibc(j)==1) then
                        ibc(k)=1
                        temp(k)=0.
            else
                        ibc(k)=0
                        if(ibc(i)==1)then
```

```
                    fac=temp(j)
                else
                    fac=temp(i)
                endif
                temp(k)=fac-(temp(i)+temp(j))/2
            endif
        enddo
        endif
        ! set solid nodeless BC
        if (neleS>0) then
        do i=1,onpoin
        do j=1,onpoin
            if (i<=j.and.edge(i,j)>0) then
                k=edge(i,j)
                if (ibc(i)==1.and.ibc(j)==1) then
                !it mean that node is outdomain
                    ibc(k)=1
                    temp(k)=0.
                else
                !it mean that node is indomain
                    ibc(k)=0
                    if(ibc(i)==1)then
                        fac=temp(j)
                    else
                        fac=temp(i)
                    endif
                    temp(k)=fac-(temp(i)+temp(j))/2.
                endif
            endif
        enddo
        enddo
        !       set solid nodeless BC in domain equal zero
        loop1:  do i=1,onpoin
        loop2:  do j=1,onpoin
            if (acheck(i,j,2)<2) cycle
            k=edge(i,j)
            ibc(k)=0
```

```
enddo loop2
enddo loop1
endif
RETURN
END SUBROUTINE setBC
!
!-------------------------------------------------------------------------------
!
SUBROUTINE setmatF()
IMPLICIT NONE
real(8)                         ::      xx,yy,dxF
real(8),dimension(3)            ::      Qg,Qs,Qh
real(8),dimension(3,3)          ::      Cf,Kf,Kv,Kh,Khf,Khs
if (FE==0) then
        num=2
else
        num=3
endif
ele:do ie=1,neleF
!       set matrix zero
        Cf=0.
        Kf=0.
        Kv=0.
        Kh=0.
        Khf=0.
        Khs=0.
        Qg=0.
        Qs=0.
        Qh=0.
        ii=intmatF(ie,1)
        jj=intmatF(ie,2)
        xx=coord(ii,1)-coord(jj,1)
        yy=coord(ii,2)-coord(jj,2)
        dxF=((xx*xx)+(yy*yy))**0.5
! spechcific heat matrix
        if (ana==1) then
                fac=densF*specF*areaF*dxF/30.
```

```
                Cf(1,1)=10.
                Cf(1,2)=5.
                Cf(2,1)=5.
                Cf(2,2)=10.
                if (FE==1) then
                        Cf(1,3)=10.
                        Cf(2,3)=10.
                        Cf(3,1)=10.
                        Cf(3,2)=10.
                        Cf(3,3)=16.
                endif
                Cf=Cf*fac
        endif
! conduction heatmatrix
        fac=condF*areaF/(dxF*3.)
        Kf(1,1)= 3.
        Kf(1,2)=-3.
        Kf(2,1)=-3.
        Kf(2,2)= 3.
        if (FE==1) then
                Kf(3,3)= 16.
        endif
        Kf=Kf*fac
!       convection matrix
        fac=flowF*specF/6.
        Kv(1,1)=-3.
        Kv(1,2)= 3.
        Kv(2,1)=-3.
        Kv(2,2)= 3.
        if (FE==1) then
                Kv(1,3)= 4.
                Kv(2,3)=-4.
                Kv(3,1)=-4.
                Kv(3,2)= 4.
                Kv(3,3)= 0.
        endif
        Kv=Kv*fac
```

```
!       heat generate matrix
        if(QtypeF(ie,1)==1)then
                fac=QgenF*areaF*dxF/6.
                Qg(1)=3.
                Qg(2)=3.
                if (FE==1) then
                        Qg(3)=4.
                endif
                Qg=Qg*fac
        endif
!       specified surface heating matrix
        if(QtypeF(ie,2)==1)then
                fac=QspecF*periF*dxF/6.
                Qs(1)=3.
                Qs(2)=3.
                if (FE==1) then
                        Qs(3)=4.
                endif
                Qs=Qs*fac
        endif
!       heat convection surrouding matrix
        if(QtypeF(ie,3)==1)then
                fac=convF*periF*dxF/60.
                Kh(1,1)=10.
        Kh(1,2)=5.
                Kh(2,1)=5.
                Kh(2,2)=10.
                if (FE==1) then
                        Kh(1,3)=10.
                        Kh(2,3)=10.
                        Kh(3,1)=10.
                        Kh(3,2)=10.
                        Kh(3,3)=16.
                endif
                Kh=Kh*fac
                fac=convF*TsurrF*periF*dxF/6.
                Qh(1)=3.
```

```
                Qh(2)=3.
                if (FE==1) Qh(3)=4.
                Qh=Qh*fac
        endif
!       assemble fluid matrix
        do i=1,num
        do j=1,num
                ii=intmatF(ie,i)
                JJ=intmatF(ie,j)
                sysK(ii,jj)=sysK(ii,jj)+Kv(i,j)+Kf(i,j)+Kh(i,j)
                if(ana==1)sysC(ii,jj)=sysC(ii,jj)+Cf(i,j)
                if (i==j) sysQ(ii)=sysQ(ii)+Qg(i)+Qs(i)+Qh(i)
        enddo
        enddo
enddo ele
return
end subroutine setmatF
!
!-------------------------------------------------------------------------
!
subroutine setmatS()
implicit none
integer(4) ::loca
real(8) :: xg1,xg2,xg3,yg1,yg2,yg3,area,xx,yy,dxS
real(8) ::      a1,a2,a3,b1,b2,b3,c1,c2,c3
real(8), dimension(6) ::Qe,Qh,Qq,Qss
real(8), dimension(6,6)         ::ake,akc,akh,ac
if (FE==0) then
        num=3
else
        num=6
endif
ele:do ie=1,neles
        ii=intmatS(ie,1)
        jj=intmatS(ie,2)
        kk=intmatS(ie,3)
        xg1=coord(ii,1)
```

```
xg2=coord(jj,1)
xg3=coord(kk,1)
yg1=coord(ii,2)
yg2=coord(jj,2)
yg3=coord(kk,2)
a1=xg2*yg3-xg3*yg2
a2=xg3*yg1-xg1*yg3
a3=xg1*yg2-xg2*yg1
area=0.5*(xg2*(yg3-yg1)+xg1*(yg2-yg3)+xg3*(yg1-yg2))
b1=yg2-yg3
b2=yg3-yg1
b3=yg1-yg2
c1=xg3-xg2
c2=xg1-xg3
c3=xg2-xg1
Qe=0.
Qh=0.
Qq=0.
Qss=0.
ake=0.
akc=0.
akh=0.
ac=0.
!       TRANSIENT
if(ana==1) then
        fac=densS*specS*area*thick/180.
        do i=1,3
        do j=1,3
                ac(i,j)=15.
                if (FE==1) then
                        ac(i,j+3)  =24.
                        ac(i+3,j)  =24.
                        ac(i+3,j+3)=16.
                endif
        enddo
        enddo
        do i=1,3
```

```
                ac(i,i)=ac(i,i)*2.
                if (FE==1) then
                        ac(i,i+3)  =ac(i,i+3)/2.
                        ac(i+3,i)  =ac(i+3,i)/2.
                        ac(i+3,i+3)=ac(i+3,i+3)*2.
                endif
        enddo
        ac=ac*fac
endif

!       ELEMENT CONDUCTION MATRIX
akc(1,1)=(b1*b1+c1*c1)/4.
akc(1,2)=(b1*b2+c1*c2)/4.
akc(1,3)=(b1*b3+c1*c3)/4.
akc(2,2)=(b2*b2+c2*c2)/4.
akc(2,3)=(b2*b3+c2*c3)/4.
akc(3,3)=(b3*b3+c3*c3)/4.
if(FE==1)then
        akc(1,4)=(b1*(b2+b3)+c1*(c2+c3))/3.
        akc(1,5)=(b1*(b1+b3)+c1*(c1+c3))/3.
        akc(1,6)=(b1*(b1+b2)+c1*(c1+c2))/3.
        akc(2,4)=(b2*(b2+b3)+c2*(c2+c3))/3.
        akc(2,5)=(b2*(b1+b3)+c2*(c1+c3))/3.
        akc(2,6)=(b2*(b1+b2)+c2*(c1+c2))/3.
        akc(3,4)=(b3*(b2+b3)+c3*(c2+c3))/3.
        akc(3,5)=(b3*(b1+b3)+c3*(c1+c3))/3.
        akc(3,6)=(b3*(b1+b2)+c3*(c1+c2))/3.
        akc(4,4)=(b2*b2+b2*b3+b3*b3+ &
                        c2*c2+c2*c3+c3*c3)*2./3.
        akc(4,5)=(b3*b3+b1*b3+b2*b3+2*b1*b2+ &
                        c3*c3+c1*c3+c2*c3+2*c1*c2)/3.
        akc(4,6)=(b2*b2+b1*b2+b2*b3+2*b1*b3+ &
                        c2*c2+c1*c2+c2*c3+2*c1*c3)/3.
        akc(5,5)=(b1*b1+b1*b3+b3*b3+ &
                        c1*c1+c1*c3+c3*c3)*2./3.
        akc(5,6)=(b1*b1+b1*b2+b1*b3+2*b2*b3+ &
                        c1*c1+c1*c2+c1*c3+2*c2*c3)/3.
```

```
            akc(6,6)=(b1*b1+b1*b2+b2*b2+ &
                                c1*c1+c1*c2+c2*c2)*2./3.
        endif
        do i=1,6
        do j=1,6
            if(i/=j.and.i>j) akc(i,j)=akc(j,i)
        enddo
        enddo
        fac=condS*thick/area
        akc=akc*fac
        do i=1,num
        do j=1,num
            ake(i,j)=ake(i,j)+akc(i,j)
        enddo
        enddo
!       ELEMENT HEAT LOAD DUE TO INTERNAL HEAT GENERATION
        if(QtypeS(ie,1)==1)then
            fac=QgenS*area*thick/3.
            do i=1,num
                Qq(i)=1
            enddo
            Qq=Qq*fac
            do i=1,num
                Qe(i)=Qe(i)+Qq(i)
            enddo
        endif
!       ELEMENT HEAT LOAD DUE TO SPECIFIED SURFACE HEATING
        loca=QtypeS(ie,2)
        if(loca>0)then
!           SPECIFIED HEATING FOR TOP AND BOTTOM
            if(loca==4)then
                fac=QspecS*area/3.
                do i=1,num
                    Qss(i)=1.
                enddo
                Qss=Qss*fac
                do i=1,num
```

```
                Qe(i)=Qe(i)+Qss(i)
            enddo
        ENDIF
!       SPECIFIED HEATING AT SIDE
        if(loca>0.and.loca<4)then
            do i=1,6
                Qss(i)=0.
            enddo
            ii=intmatS(ie,loca)
            if (loca/=3) then
                jj=intmatS(ie,loca+1)
            else
                jj=intmatS(ie,1)
            endif
            xx=coord(ii,1)-coord(jj,1)
            yy=coord(ii,2)-coord(jj,2)
            dxS=((xx*xx)+(yy*yy))**0.5
            fac=QspecS*thick*dxS/6.
            if (loca<3) then
                ii=loca
                jj=loca+1
                if (loca==1) then
                    kk=6
                else
                    kk=4
                endif
            else
                ii=1
                jj=3
                kk=5
            endif
            Qss(ii)=3.
            Qss(jj)=3.
            if (FE==1) Qss(kk)=4.
            Qss=Qss*fac
            do i=1,num
                Qe(i)=Qe(i)+Qss(i)
```

```
                enddo
            endif
        endif
!       ELEMANT CONVECTION MATRICES
        loca=QtypeS(ie,3)
        if(loca>0)then
!           CONVECTION FOR TOP AND BOTTOM
            if (loca==4) then
                fac=convS*area/180.
                do i=1,3
                do J=1,3
                    akh(i,j)=15.
                    if (FE==1) then
                        akh(i,j+3)  =24.
                        akh(i+3,j)  =24.
                        akh(i+3,j+3)=16.
                    endif
                enddo
                enddo
                do i=1,3
                    akh(i,i)=akh(i,i)*2.
                    if (FE==1) then
                        akh(i,i+3)  =akh(i,i+3)/2.
                        akh(i+3,i)  =akh(i+3,i)/2.
                        akh(i+3,i+3)=akh(i+3,i+3)*2.
                    endif
                enddo
                akh=akh*fac
                fac=convS*area*TsurrS/3.
                do i=1,num
                    Qh(I)=1
                enddo
                Qh=Qh*fac
                do i=1,num
                    Qe(i)=Qe(i)+Qh(i)
                do j=1,num
                    ake(i,j)=ake(i,j)+akh(i,j)
```

```
            enddo
            enddo
        endif
!       CONVECTION FOR SIDE
        if (loca>0.and.loca<4) then
            Qh=0.
            akh=0.
            ii=intmatS(ie,loca)
            if (loca/=3) then
                jj=intmatS(ie,loca+1)
            else
                jj=intmatS(ie,1)
            endif
            xx=coord(ii,1)-coord(jj,1)
            yy=coord(ii,2)-coord(jj,2)
            dxS=((xx*xx)+(yy*yy))**0.5
            fac=convS*thick*dxS/30.
            if (loca<3) then
                ii=loca
                jj=loca+1
                if (loca==1) then
                    kk=6
                else
                    kk=4
                endif
            else
                ii=1
                jj=3
                kk=5
            endif
            akh(ii,ii)=10.
            akh(ii,jj)=5.
            akh(jj,ii)=5.
            akh(jj,jj)=10.
            if (FE==1) then
                akh(ii,kk)=10.
                akh(jj,kk)=10.
```

```
                        akh(kk,ii)=10.
                        akh(kk,jj)=10.
                        akh(kk,kk)=16.
                    endif
                    akh=akh*fac
                    fac=convS*thick*TsurrS*dxS/6.
                    Qh(ii)=3.
                    Qh(jj)=3.
                    if (FE==1) Qh(kk)=4.
                    Qh=Qh*fac
                    do i=1,num
                        Qe(i)=Qe(i)+Qh(i)
                    do j=1,num
                        ake(i,j)=ake(i,j)+akh(i,j)
                    enddo
                    enddo
                endif
            endif
!           ASSEMBLE
            do i=1,num
                ii=intmatS(ie,i)
                sysQ(II)=sysQ(ii)+Qe(i)
            do j=1,num
                jj=intmatS(ie,j)
                sysK(ii,jj)=sysK(ii,jj)+ake(i,j)
                if(ana==1)sysC(ii,jj)=sysC(ii,jj)+ac(i,j)
            enddo
            enddo
enddo ele
return
end subroutine setmatS
!
!-------------------------------------------------------------------------
!
SUBROUTINE setmatFS()
IMPLICIT NONE
INTEGER(4), DIMENSION(6)        ::intmatFS2
```

```
real(8)                         ::      xx,yy,dx
real(8),dimension(3)            ::      Qg,Qs,Qh
real(8),dimension(3,3)          ::      Cf,Kf,Kv,Kh,Khf,Khs
ele:do ie=1,neleFS
        Kh=0.
        ii=intmatFS(ie,1)
        jj=intmatFS(ie,2)
        xx=coord(ii,1)-coord(jj,1)
        yy=coord(ii,2)-coord(jj,2)
        dx=((xx*xx)+(yy*yy))**0.5
        fac=convFS(ie)*thick*dx/30.
        Kh(1,1)=10.
  Kh(1,2)=5.
        Kh(2,1)=5.
        Kh(2,2)=10.
        if (FE==1) then
                Kh(1,3)=10.
                Kh(2,3)=10.
                Kh(3,1)=10.
                Kh(3,2)=10.
                Kh(3,3)=16.
        endif
        Kh=Kh*fac
        if (FE==0) then
                do i=1,2
                do j=1,2
                        ii=intmatFS(ie,i)
                        jj=intmatFS(ie,j)
                        sysK(ii,jj)=sysK(ii,jj)+Kh(i,j)

                        jj=intmatFS(ie,j+2)
                        sysK(ii,jj)=sysK(ii,jj)-Kh(i,j)

                        ii=intmatFS(ie,i+2)
                        jj=intmatFS(ie,j)
                        sysK(ii,jj)=sysK(ii,jj)-Kh(i,j)
```

```
                    jj=intmatFS(ie,j+2)
                    sysK(ii,jj)=sysK(ii,jj)+Kh(i,j)
                enddo
                enddo
        else
                intmatFS2(1)=intmatFS(ie,1)
                intmatFS2(2)=intmatFS(ie,2)
                intmatFS2(3)=intmatFS(ie,5)
                intmatFS2(4)=intmatFS(ie,3)
                intmatFS2(5)=intmatFS(ie,4)
                intmatFS2(6)=intmatFS(ie,6)
                do i=1,3
                do j=1,3
                        ii=intmatFS2(i)
                        jj=intmatFS2(j)
                        sysK(ii,jj)=sysK(ii,jj)+Kh(i,j)

                        jj=intmatFS2(j+3)
                        sysK(ii,jj)=sysK(ii,jj)-Kh(i,j)

                        ii=intmatFS2(i+3)
                        jj=intmatFS2(j)
                        sysK(ii,jj)=sysK(ii,jj)-Kh(i,j)

                        jj=intmatFS2(j+3)
                        sysK(ii,jj)=sysK(ii,jj)+Kh(i,j)
                        enddo
                        enddo
                endif
enddo ele
return
end subroutine setmatFS
!
!-------------------------------------------------------------------------
!
subroutine manmat()
implicit none
```

```
 if (ana==1)then
        if (step==1)then
              allocate (sysQ2(npoin),sysR(npoin),sysK1(npoin,npoin))
              sysQ2=0.
              sysR=0.
              sysK1=sysK
        else
              sysQ2=0.
              sysR=0.
        endif
        do i=1,npoin
        do j=1,npoin
              sysK(i,j)=(sysC(i,j)/dt)+(sysK1(I,J)*zeta)
              sysQ2(i)=sysQ2(i)+(  ((sysC(i,j)/dt)-(sysK1(i,j)*(1-zeta)))  *temp(j))
        enddo
        enddo
        sysR=sysQ+sysQ2
else
        allocate (sysR(npoin))
        sysR=sysQ
        deallocate(sysQ)
endif
return
end subroutine manmat
!
!-------------------------------------------------------------------------------
!
subroutine solver()
use solve
implicit none
call applyBC()
if (sol==0) then
        call gauss()
endif
if (sol==1) then
        call unsympcg(sysK,sysR,temp,npoin)
endif
```

```
return
end subroutine solver
!
!-------------------------------------------------------------------------
!
subroutine applyBC()
implicit none
integer(4) ::ieq,ir,ic
do 10 ieq=1,npoin
        if (ibc(ieq)==0) goto 10
        do 20 ir=1,npoin
                if (ir==ieq) goto 20
                sysR(ir)=sysR(ir)-sysK(ir,ieq)*temp(ieq)
                sysK(ir,ieq)=0.
        20 continue
        do ic=1,npoin
                sysK(ieq,ic)=0
        enddo
        sysK(ieq,ieq)=1.
        sysR(ieq)=temp(ieq)
10 continue
return
end subroutine applyBC
!
!-------------------------------------------------------------------------
!
subroutine gauss
implicit none
integer(4)              ::      ip,n,ie,ic
real(8)                 ::      ratio,sum
n=npoin
call scale (n)
do ip=1,n-1
        call pivot(n,ip)
        do ie=ip+1,n
                ratio=sysK(ie,ip)/sysK(ip,ip)
                do ic=ip+1,n
```

```
                    sysK(ie,ic)=sysK(ie,ic)-ratio*sysK(ip,ic)
                enddo
                sysR(ie)=sysR(ie)-ratio*sysR(ip)
        enddo
        do ie=ip+1,n
                sysK(ie,ip)=0.
        enddo
enddo
temp(n)=sysR(n)/sysK(n,n)
do ie=n-1,1,-1
        sum=0.
        do ic=ie+1,n
                sum=sum+sysK(ie,ic)*temp(ic)
        enddo
        temp(ie)=(sysR(ie)-sum)/sysK(ie,ie)
enddo
return
end subroutine gauss
!
!-------------------------------------------------------------------------------
!
subroutine pivot(n,ip)
implicit none
integer(4)              ::      n,ip,jp
real(8)                 ::      big,amax,dumy
jp=ip
big=abs(sysK(ip,ip))
do i=ip+1,n
        amax=abs(sysK(i,ip))
        if (amax>big) then
                big=amax
                jp=i
        endif
enddo
if (jp/=ip) then
        do j=ip,n
                dumy=sysK(jp,j)
```

```
                sysK(jp,j)=sysK(ip,j)
                sysK(ip,j)=dumy
        enddo
        dumy=sysR(jp)
        sysR(jp)=sysR(ip)
        sysR(ip)=dumy
endif
return
end subroutine pivot
!
!-------------------------------------------------------------------------
!
subroutine scale(n)
implicit none
integer(4)              ::      n,ic,ie
real(8)                 ::      big,amax
do ie=1,n
        big=abs(sysK(ie,1))
        do ic=2,n
                amax=abs(sysK(ie,ic))
                IF (amax>big) big=amax
        enddo
        do ic=1,n
                sysK(ie,ic)=sysK(ie,ic)/big
        enddo
        sysR(ie)=sysR(ie)/big
enddo
return
end subroutine scale
!
!-------------------------------------------------------------------------
!
subroutine printtxt()
IMPLICIT NONE
INTEGER(4), ALLOCATABLE, DIMENSION(:)  ::Fup
if(ana==0)then
        open(unit=8,file='Zss_.txt',status='old')
```

```
        WRITE(8,45) FE
        45 FORMAT('FE     ',I16 )
        WRITE(8,50) zeta, dt
        50 FORMAT('zeta     ', F14.2,'    dt              ',E16.8 )
        WRITE(8,55) nelef,neles
        55 FORMAT('F element  ',I13,'    S element        ',I16)
        num= npoin-onpoin
        WRITE(8,60) onpoin,num
        60 FORMAT('actual node  ', I11,'    nodeless variable  ', I16)
        WRITE(8,40) step
        40 FORMAT('step    ', I16)
        WRITE(8,80) rtime
        80 FORMAT('time    ',E16.8)
        WRITE(8,70)
        70 FORMAT('node           T' )
! calaulate true temperature of nodeless
        DO i = 1,onpoin
                WRITE(8,140) i,temp(i)
                140 FORMAT(I6,E18.8)
        END DO
else
        open(unit=9,file='Zts_.txt',status='old')
        if(step==1)then
                WRITE(9,450) FE
                450 FORMAT('FE     ',I16 )
                WRITE(9,500) zeta, dt
                500 FORMAT('zeta     ', F14.2,'    dt              ',E16.8 )
                WRITE(9,550) nelef,neles
                550 FORMAT('F element  ',I13,'    S element        ',I16)
                WRITE(9,600) onpoin, npoin
                600 FORMAT('actual node  ', I11,'    nodeless variable  ', I16)
        endif
        if(step==1.or.step2==nsave.or.rtime>=time)then
                if(step==1)then
                        WRITE(9,510) step
                        510 FORMAT('step    ', I16)
                else
```

```
                WRITE(9,400) step
                400 FORMAT(/,'step    ', I16)
            endif

            WRITE(9,480) rtime
            480 FORMAT('time    ',E16.8)
            WRITE(9,470)
            470 FORMAT('node          T' )
            DO i = 1,onpoin
                WRITE(9,1400) i,temp(i)
                1400 FORMAT(I6,E18.8)
            END DO
            step2=0
        endif
endif
end subroutine printtxt
!
!-------------------------------------------------------------------------
!
END MODULE HF
!
!-------------------------------------------------------------------------
!
PROGRAM NODELESS
use HF
IMPLICIT NONE
CALL MAIN()
WRITE(6,100)
100 FORMAT('FINISH')
END PROGRAM NODELESS
```

ภาคผนวก ค

รายละเอียดของโปรแกรมคอมพิวเตอร์ Adaptive

โปรแกรมคอมพิวเตอร์ Adaptive ที่ได้ประดิษฐ์ขึ้นเพื่อใช้คำนวณหาขนาดของเอลิเมนต์ที่เหมาะสมสำหรับกการวิเคราะห์อุณหภูมิของโครงสร้างหล่อเย็นด้วยการพาความร้อนที่ได้กล่าวไว้ในบทที่ 6 มีรายละเอียดดังนี้

```
MODULE HF
IMPLICIT NONE
CHARACTER(len=20) text,name1
integer(4)      ::i,j,ii,jj,kk,ie,neleF,neleS,npoin,num
real(8)         ::D2Fmax,D2Smax,D2max
real (8)        ::D2Fmin,D2Smin,D2min,D2mean,hmin,hmax
integer(4),allocatable, dimension(:)          ::check
integer(4), allocatable, dimension (:,:)      ::intmatF,intmatS
real (8), allocatable, dimension (:)          ::temp,D2F,hnew,D2
real (8), allocatable, dimension (:,:)        ::coord,D2S
CONTAINS
!
!-------------------------------------------------------------------------
!
SUBROUTINE MAIN()
IMPLICIT NONE
call read_input()
allocate(hnew(npoin),D2(npoin),check(npoin))
call DS()
call DF()
call DS_F()
call cal_hmin()
call printout()
END SUBROUTINE main
!
!-------------------------------------------------------------------------
!
```

```
subroutine read_input()
IMPLICIT NONE
write(6,5)
5 format(/,'PLEASE ENTER THE INPUT FILE NAME:')
read(5,*)name1
OPEN(UNIT=7,FILE=name1,STATUS='OLD')
!       read solid data
READ(7,10) text
READ(7,*) hmin,hmax
READ(7,10) text
10 FORMAT(20A4)
READ(7,10) text
READ(7,*) neleS
READ(7,10) text
allocate(intmatS(neleS,3))
do ie=1,neleS
        read(7,*) ii,(intmatS(ie,j),j=1,3)
enddo
!       read solid-fluid data
READ(7,10) text
READ(7,10) text
READ(7,*) neleF
READ(7,10) text
allocate(intmatF(nelef,4))
do ie=1,neleF
        read(7,*) ii,(intmatF(ie,j),j=1,4)
enddo
!       read node data
READ(7,10) text
READ(7,10) text
READ(7,*) npoin
READ(7,10) text
allocate(temp(npoin),coord(npoin,npoin))
do i=1,npoin
        read(7,*) ii,(coord(ii,j),j=1,2),temp(i)
enddo
```

```
end subroutine read_input
!
!-------------------------------------------------------------------------
!
subroutine DF()
IMPLICIT NONE
INTEGER(4)          ::in
REAL(8)             ::xx,yy,L,Dx,Dxx
REAL(8), ALLOCATABLE, DIMENSION(:)          ::D1n,Nsum
allocate(D1n(npoin),Nsum(npoin),D2F(npoin))
!       first derivatives for each element
D1n=0.
Nsum=0
do ie=1,neleF
        ii=intmatF(ie,1)
        jj=intmatF(ie,2)
        xx=coord(ii,1)-coord(jj,1)
        yy=coord(ii,2)-coord(jj,2)
        L=((xx*xx)+(yy*yy))**0.5
        Dx=(-temp(ii)+temp(jj))/L
        do j=1,2
                in=intmatF(ie,j)
                D1n(in)=D1n(in)+Dx
                Nsum(in)=Nsum(in)+1
        enddo
enddo
!       first derivatives for each node
do i=1,npoin
        if(Nsum(i)>0)then
                D1n(i)=D1n(i)/Nsum(i)
        endif
enddo
!       second derivatives for each element
D2F=0.
Nsum=0
do ie=1,neleF
        ii=intmatF(ie,1)
```

```
        jj=intmatF(ie,2)
        xx=coord(ii,1)-coord(jj,1)
        yy=coord(ii,2)-coord(jj,2)
        L=((xx*xx)+(yy*yy))**0.5
        Dxx=(-D1n(ii)+D1n(jj))/L

        do j=1,2
                in=intmatF(ie,j)
                D2F(in)=D2F(in)+Dxx
                Nsum(in)=Nsum(in)+1
        enddo
enddo
!       second derivatives for each node
D2Fmax=0.
D2Fmin=1.e16
do i=1,npoin
        if(Nsum(i)>0)then
                D2F(i)=D2F(i)/Nsum(i)
        endif
        if(abs(D2F(i))>D2Fmax)D2Fmax=abs(D2F(i))
        if(abs(D2F(i))<D2Fmin)D2Fmin=abs(D2F(i))
enddo

end subroutine DF
!
!-------------------------------------------------------------------------
!
subroutine DS()
IMPLICIT NONE
REAL(8)             ::xg1,xg2,xg3,yg1,yg2,yg3,area
REAL(8)             ::a1,a2,a3,b1,b2,b3,c1,c2,c3,p1,p2,p3
integer(4)          ::in
integer(4), ALLOCATABLE, DIMENSION(:)          ::Nsum
REAL(8)                                        ::Dx,Dy,Dxx,Dyy,Dxy
REAL(8), ALLOCATABLE, DIMENSION(:,:)           ::D1n,D2n
allocate(D1n(npoin,2),Nsum(npoin),D2n(npoin,3),D2S(npoin,2))
!       first derivatives for each element
```

```
D1n=0.
Nsum=0
do ie=1,neleS
        ii=intmatS(ie,1)
        jj=intmatS(ie,2)
        kk=intmatS(ie,3)

        xg1=coord(ii,1)
        xg2=coord(jj,1)
        xg3=coord(kk,1)
        yg1=coord(ii,2)
        yg2=coord(jj,2)
        yg3=coord(kk,2)

        b1=yg2-yg3
        b2=yg3-yg1
        b3=yg1-yg2
        c1=xg3-xg2
        c2=xg1-xg3
        c3=xg2-xg1

        a1=xg2*yg3-xg3*yg2
        a2=xg3*yg1-xg1*yg3
        a3=xg1*yg2-xg2*yg1

        area=0.5*(xg2*(yg3-yg1)+xg1*(yg2-yg3)+xg3*(yg1-yg2))

        Dx=(b1*temp(ii)+b2*temp(jj)+b3*temp(kk))/(2.*area)
        Dy=(c1*temp(ii)+c2*temp(jj)+c3*temp(kk))/(2.*area)

        do j=1,3
                in=intmatS(ie,j)
                D1n(in,1)=D1n(in,1)+Dx
                D1n(in,2)=D1n(in,2)+Dy
                Nsum(in)=Nsum(in)+1
        enddo
enddo
```

```
!       first derivatives for each node
do i=1,npoin
        if(Nsum(i)>0)then
                D1n(i,1)=D1n(i,1)/Nsum(i)
                D1n(i,2)=D1n(i,2)/Nsum(i)
        endif
enddo
!       second derivatives for each element
D2n=0.
Nsum=0
do ie=1,neleS
        ii=intmatS(ie,1)
        ii=intmatS(ie,1)
        jj=intmatS(ie,2)
        kk=intmatS(ie,3)

        xg1=coord(ii,1)
        xg2=coord(jj,1)
        xg3=coord(kk,1)
        yg1=coord(ii,2)
        yg2=coord(jj,2)
        yg3=coord(kk,2)

        b1=yg2-yg3
        b2=yg3-yg1
        b3=yg1-yg2
        c1=xg3-xg2
        c2=xg1-xg3
        c3=xg2-xg1

        a1=xg2*yg3-xg3*yg2
        a2=xg3*yg1-xg1*yg3
        a3=xg1*yg2-xg2*yg1

        area=0.5*(xg2*(yg3-yg1)+xg1*(yg2-yg3)+xg3*(yg1-yg2))

        P1=D1n(ii,1)
```

```
            P2=D1n(jj,1)
            P3=D1n(kk,1)
            Dxx =(b1*P1+b2*P2+b3*P3)/(2.*area)

            P1=D1n(ii,2)
            P2=D1n(jj,2)
            P3=D1n(kk,2)
            Dyy =(c1*P1+c2*P2+c3*P3)/(2.*area)
            Dxy =(b1*P1+b2*P2+b3*P3)/(2.*area)

            do j=1,3
                  in=intmatS(ie,j)
                  D2n(in,1)=D2n(in,1)+Dxx
                  D2n(in,2)=D2n(in,2)+Dyy
                  D2n(in,3)=D2n(in,3)+Dxy
                  Nsum(in)=Nsum(in)+1
            enddo
      enddo
      !second derivatives for each node
      do i=1,npoin
            if(Nsum(i)>0)then
                  D2n(i,1)=D2n(i,1)/Nsum(i)
                  D2n(i,2)=D2n(i,2)/Nsum(i)
                  D2n(i,3)=D2n(i,3)/Nsum(i)
            endif
      enddo
      ! Principle second derivatives for each node
      D2S=0.
      D2Smax=0.
      D2Smin=1.e16
      do i=1,npoin
            if(Nsum(i)>0)then
                  Dxx=D2n(i,1)
                  Dyy=D2n(i,2)
                  Dxy=D2n(i,3)
                  D2S(i,1)=((Dxx+Dyy)/2.)+(((Dxx-Dyy)/2.)**2+Dxy**2)**0.5
                  D2S(i,2)=((Dxx+Dyy)/2.)-(((Dxx-Dyy)/2.)**2+Dxy**2)**0.5
```

```
            endif
            if(abs(D2S(i,1))>D2Smax)D2Smax=abs(D2S(i,1))
            if(abs(D2S(i,2))>D2Smax)D2Smax=abs(D2S(i,2))
            if(abs(D2S(i,1))<D2Smin)D2Smin=abs(D2S(i,1))
            if(abs(D2S(i,2))<D2Smin)D2Smin=abs(D2S(i,2))
    enddo
    end subroutine DS
    !
    !-------------------------------------------------------------------------
    !
    subroutine DS_F()
    IMPLICIT NONE
    REAL(8)                 ::fac
    if(neleF==0)then
            D2Fmax=0
            D2Fmin=1.e16
    endif
    if(neleS==0)then
            D2Smax=0
            D2Smin=1.e16
    endif
    D2max=D2Fmax
    D2min=D2Fmin
    if(D2max<D2Smax)D2max=D2Smax
    if(D2min>D2Smin)D2min=D2Smin

    D2=0.
    D2F=abs(D2F)
    D2S=abs(D2S)

    do i=1,npoin
            fac=0.
            if(neleF>0.and.D2F(i)>0)then
                    D2(i)=D2F(i)
            endif
            if(neleS>0)then
                    if(D2S(i,1)>0.or.D2S(i,2)>0)then
```

```
                    D2(i)=D2S(i,1)
                    if(D2(i)<D2S(i,2))D2(i)=D2S(i,2)
                endif
        endif
enddo
check=0
if(neleS>0.and.neleF>0)then
        do ie=1,neleF
                ii=intmatF(ie,1)
                jj=intmatF(ie,3)
                if(D2(ii)>D2(jj))then
                        D2(jj)=D2(ii)
                        check(jj)=1
                endif
                ii=intmatF(ie,2)
                jj=intmatF(ie,4)
                if(D2(ii)>D2(jj))then
                        D2(jj)=D2(ii)
                        check(jj)=1
                endif
        enddo
endif
do i=1,npoin
        D2mean=D2mean+D2(i)
enddo
D2mean=D2mean/npoin
end subroutine DS_F
!
!-------------------------------------------------------------------------------
!
subroutine cal_hmin()
IMPLICIT NONE
write(6,50)
50 format('              node      ',' h_min')
hnew=0.
do i=1,npoin
                hnew(i)=((hmin**2.)*D2max/D2(i))**0.5
```

```
                if(hnew(i)>hmax) hnew(i)=hmax
                if(check(i)==1)then
                        write(6,100) i,hnew(i)
                        100 format(i20,e20.6,'          Fluid')
                else
                        write(6,200) i,hnew(i)
                        200 format(i20,e20.6)
                endif
                if(hnew(i)==hmin) num=i
enddo

write(6,500)D2max
500 format('maximum second derivertive T'e20.6)
write(6,600)D2min
600 format('minimum second derivertive T'e20.6)
write(6,700)D2mean
700 format('mean second derivertive T   'e20.6)
write(6,1000)num
1000 format('Node h_min       ',i20)
end subroutine cal_hmin
!
!-------------------------------------------------------------------------------
!
subroutine printout()
IMPLICIT NONE
open(unit=9,file='h_new.txt',status='old')
write(9,500)D2max
500 format('maximum second derivertive T'e20.6)
write(9,600)D2min
600 format('minimum second derivertive T'e20.6)
write(9,700)D2mean
700 format('mean second derivertive T   'e20.6)
write(9,1000)num
1000 format('Node h_min       ',i20)
write(9,50)
50 format('              node      ',' h_min')
do i=1,npoin
```

```
        write(9,1200) i,hnew(i)
        1200 format(i20,e20.6)
enddo
end subroutine printout
!
!-------------------------------------------------------------------------
!
END MODULE HF
!
!-------------------------------------------------------------------------
!
PROGRAM adaptive
use HF
IMPLICIT NONE
CALL MAIN()
WRITE(6,100)
100 FORMAT('FINISH')
END PROGRAM adaptive
```

ภาคผนวก ง

The 22nd Conference of Mechanical Engineering Network of Thailand
15-17 October 2008, Thammasat University, Rangsit Campus, Pathum Thani, Thailand

Adaptive Nodeless Variable Finite Element Method for Convectively-Cooled Solids

Sutthikom Puntimakornkij, Atipong Malatip and Pramote Dechaumphai*

Department of Mechanical Engineering, Chulalongkorn University,
254 Phyathai Road,Patumwan, Bangkok 10330, Thailand,
Tel: 0-2218-6621, Fax: 0-2218-6621 *E-mail: fmepdc@eng.chula.ac.th

Abstract

The adaptive nodeless variable finite element method for convectively-cooled solids is presented. The nodeless variable finite element method is developed for analyzing heat transfer in solids that is coupled with the flow in channels. The nodeless variable element employs quadratic interpolation functions to provide higher solution accuracy without requiring actual nodes. The coupled fluid/solid solution is further improved by incorporating an adaptive meshing technique. Several examples are presented to demonstrate the efficiency of the combined method.

Key words: Adaptive mesh, nodeless variable Finite element, convectively-cooled solids.

1. Introduction

Design and analysis of convectively-cooled solids are encountered in many practical engineering problems. Currently, the finite element method is widely used to solve for the temperature distribution in solids, as well as the behavior of the fluid flow [1-3]. The problem is more complicated when the coupled analysis is required to predict the solid and fluid behavior simultaneously. Such analysis is known as the conjugate heat transfer [4-5] that needs immense effort to solve the Navier-Stokes equations of the fluid. Recently, the nodeless variable finite element [6-7] has been developed to provide higher solution accuracy without requiring actual nodes. In addition, the solution accuracy can be further improved by using the adaptive finite element technique [3,6-7]. The technique generates small elements clustered in the high temperature gradient regions to provide accurate solution. Larger elements are generated in the other regions where the temperature is uniform to reduce the number of unknowns and computational time.

In this paper, an adaptive nodeless variable finite element method is developed to predict the temperatures in the solid and the fluid flowing in a channel. Convection heat transfer between the solid and fluid is included along the solid/fluid interface. For a fluid flow through a channel, the fluid analysis may be treated as one dimensional flow. In this case, the couple solid/fluid analysis can be simplified so that the computational effort is reduced significantly. Heat transfer in the fluid thus can be characterized by the fluid bulk temperature and the convection coefficient. The solid/fluid heat transfer is then coupled and their solutions can be solved simultaneously. The nodeless variable finite element is employed to improve the predicted temperature distribution. The nodeless variable finite element uses the quadratic interpolation functions to describe the temperature distribution over the element. The use of the nodeless variable finite element can also be referred to as a hierarchical methodology, since the element reduces to the standard linear element when the nodeless variables are constrained to zero or eliminated.

To further improve the predicted solution of the solid and fluid temperatures, an adaptive finite element technique has been incorporated. Examples are presented in the paper to demonstrate the efficiency of the proposed method. These examples are a convectively-cooled solid subjected to uniform heating and a plate with intense heating.

2. Theoretical formulation

2.1 Governing equations

The equations which govern convectively-cooled solids are the one-dimensional conservation of energy equation for the fluid flow, and the two-dimensional conservation of energy equation for the solid. These governing differential equations are,

Energy equation in fluid,

$$\rho_f c_f u_f A_f \frac{\partial T_f}{\partial x} - k_f A_f \frac{\partial^2 T_f}{\partial x^2} - hp\left(T_f - T_s\right) - Q_f A_f = 0 \quad (1)$$

Energy equation in solid,

$$k_s\left(\frac{\partial^2 T_s}{\partial x^2} + \frac{\partial^2 T_s}{\partial y^2}\right) - Q_s = 0 \quad (2)$$

where the subscript f and s are for the fluid and the solid, respectively; u is the velocity in x direction, ρ is the density, c is the specific heat, k is the coefficient of thermal conductivity, h is the convective heat transfer coefficient, p is the perimeter, Q is the internal heat generation rate per volume and T is the temperature.

2.2 Finite element formulation

The nodeless variable finite element equations are derived using the method of weighted residuals. The mass transport element and triangular element are employed in this study. For both elements, the distributions of temperature over the elements are assumed respectively in the form

$$T(x) = \sum_{i=1}^{3} N_i(x) T_i \tag{3}$$

$$T(x,y) = \sum_{i=1}^{6} N_i(x,y) T_i \tag{4}$$

where N_i consists of the element interpolation functions and T_i is the vector of the unknown temperatures and the nodeless variables. For the fluid, the nodal temperatures are T_1 and T_2, while T_3 is the nodeless variable. For the solid, the nodal temperatures are T_1 through T_3, while T_4 through T_6 are the nodeless variables. The element interpolation functions, N_1 and N_2 in fluid, N_1 through N_3 in solid are the standard two-node element and three-node triangular element, respectively while N_3 in fluid and N_4 through N_6 in solid are the nodeless variable interpolation functions. The interpolation functions implemented in this paper are.

For fluid,

$$\begin{aligned} N_1 &= 1 - x/L \\ N_2 &= x/L \\ N_3 &= 4N_1N_2 \end{aligned} \tag{5}$$

For solid,

$$\begin{aligned} N_1 &= L_1 \\ N_2 &= L_2 \\ N_3 &= L_3 \\ N_4 &= 4L_2L_3 \\ N_5 &= 4L_1L_3 \\ N_6 &= 4L_1L_2 \end{aligned} \tag{6}$$

where L in eq. (6) is area coordinates [8-9].

$$L_i = (a_i + b_i x + c_i y)/2A \tag{7}$$

To derive the nodeless variable finite element matrices, the method of weighted residuals is first applied to Eqs. (1) and (2). Integration by parts is then performed using the Gauss theorem to yield the boundary terms for applying boundary conditions. The nodeless variable finite element equations are,

For fluid,

$$\begin{aligned} &\int_0^L kA\left\{\frac{\partial N}{\partial x}\right\}\left[\frac{\partial N}{\partial x}\right]dx\{T_f\} + \int_0^L \dot{m}c\{N\}\left[\frac{\partial N}{\partial x}\right]dx\{T_f\} \\ &+ \int_0^L hp\{N\}[N]dx\{T_f\} - \int_0^L hp\{N\}[N]dx\{T_s\} \\ &= \left(\{N\}kA\frac{\partial T_f}{\partial x}\right)\Big|_0^L + \int_0^L QA\{N\}dx \end{aligned} \tag{8}$$

For solid,

$$\begin{aligned} &\int_A kt\left(\left\{\frac{\partial N}{\partial x}\right\}\left[\frac{\partial N}{\partial x}\right] + \left\{\frac{\partial N}{\partial y}\right\}\left[\frac{\partial N}{\partial y}\right]\right)dA\{T_s\} \\ &+ \int_0^L hp\{N\}[N]dx\{T_s\} - \int_0^L hp\{N\}[N]dx\{T_f\} \\ &= \int_\Gamma kt\{N\}\left(\frac{\partial T_s}{\partial x}n_x + \frac{\partial T_s}{\partial y}n_y\right)d\Gamma + \int_A Qt\{N\}dA \end{aligned} \tag{9}$$

where A is the element cross-sectional area for fluid and the element area in solid, $\dot{m}$ is the mass flow rate, L is the element length, Γ is the element boundary, t is the element thickness, n_x and n_y are the direction cosines of the unit vector normal to the edge. The nodeless variable finite element matrices are.

For fluid,

$$\left([K_c] + [K_v] + [K_{f-s}]\right)\{T_f\} - [K_{f-s}]\{T_s\} = \{Q_{Q_f}\} \tag{10}$$

For solid,

$$\left([K_s] + [K_{f-s}]\right)\{T_s\} - [K_{f-s}]\{T_f\} = \{Q_{Q_s}\} \tag{11}$$

The above Eqs. (10) and (11) can be written together as,

$$\begin{bmatrix} [K_c] + [K_v] + [K_{f-s}] & -[K_{f-s}] \\ -[K_{f-s}] & [K_s] + [K_{f-s}] \end{bmatrix} \begin{Bmatrix} T_f \\ T_s \end{Bmatrix} = \begin{Bmatrix} Q_{Q_f} \\ Q_{Q_s} \end{Bmatrix} \tag{12}$$

where $[K_c]$ is the conduction matrix, $[K_v]$ is the mass transport convection matrix, $[K_{f-s}]$ is the convection matrix between fluid and solid, $\{Q_Q\}$ is the internal heat generate load vector, $\{T_f\}$ and $\{T_s\}$ are the vectors of the nodal temperatures in the fluid and solid, respectively.

2.3 Adaptive Meshing

The basic idea of the adaptive meshing technique is to construct a completely new mesh based on the solution obtained from the previous mesh. The technique consists of two main steps; the first step is the determination of proper element sizes and the second step is the new mesh generation. The temperature, T, is used herein as the indicator for computing proper element sizes at different locations in the domain. As small elements must be placed in the region where changes in the temperature gradients are high, the second derivatives of the temperature at a point with respect to the global coordinates X and Y are needed. Using principal stresses determination from a given state of stresses at a point, the maximum principal quantities are then used to compute the proper element size h_i by requiring that the error

should be uniform for all elements,

$$h_i^2 \lambda_i = h_{min}^2 \lambda_{max} = \text{constant} \quad (13)$$

where the subscript i denotes the direction of the maximum and minimum element length, and λ_i is the higher principal quantity of the element considered,

$$\lambda_i = \max\left(\left|\frac{\partial^2 T}{\partial X^2}\right|, \left|\frac{\partial^2 T}{\partial Y^2}\right|\right) \quad (14)$$

In Eq. (13), λ_{max} is the maximum principal quantity for all elements and h_{min} is the minimum element size specified by users. The node spacing, h_i, is scaled according to the maximum value of the second derivatives of the temperature. Such technique generates small elements in the regions with large change in the temperature gradients to increase the analysis solution accuracy. At the same time, larger elements are generated in the other regions where the temperature profile is nearly uniform to reduce the computational time and the computer memory.

3. Results

In this section two example problems are presented. The first example, a convectively-cooled solid subjected to uniform heating, is chosen to evaluate the nodeless variable finite element formulation and to validate the developed computer programs. The second example, a plate with intense heating, is used to evaluate the performance of the adaptive nodeless variable finite element method. The conjugate gradient method is used to solve the set of algebraic equations of these problems.

3.1 Convectively-cooled solid subjected to uniform heating

The first example for evaluating the efficiency of the nodeless variable finite element formulation is the problem of a convectively-cooled as shown in Fig. 1. The solid is subjected to uniform heating q along the upper wall. Heat is conducted in the solid and convection is occurred to the fluid that flows along the lower wall. All other walls of the solid are assumed to be adiabatic.

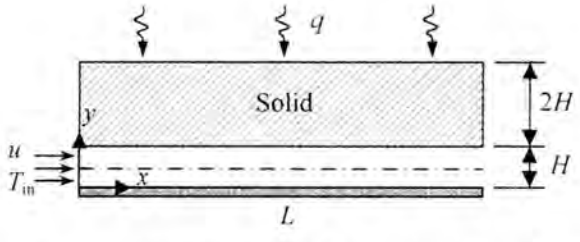


Figure 1. Problem statement of convectively-cooled solid subjected to uniform heating.

The parameters used in this example are as follows: the geometry sizes H = 0.1 m, L = 2 m, the uniform heating q = 8,000 W/m^2, the thermal conductivity k_s = 1,000 W/m-K, the flow channel parameters are Pe = 200 (Re = 286) and Pe = 400 (Re = 572) with Pr = 0.7. The convection coefficient, h is determined using the procedure presented in [10]. The finite element model consisting of 650 elements and 408 nodes, as shown in Fig. 2, is used in this study.

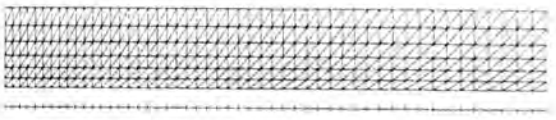
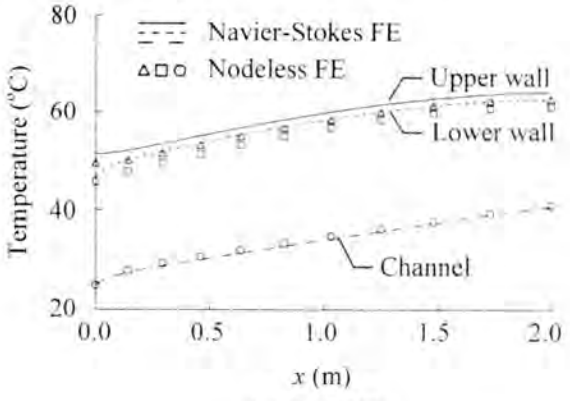
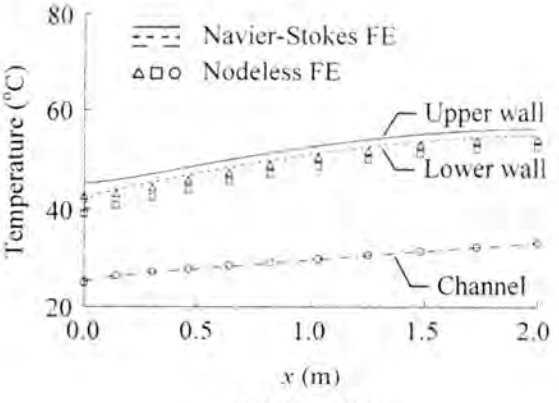


Figure 2. Nodeless variable finite element model consisting of 650 elements and 408 nodes.

Figure 3 shows the predicted temperature distributions along the upper wall, the lower wall and in the channel. The predicted temperature distributions in solid are shown in Fig. 4. Figure 5 shows the predicted temperature distributions at $x = L$, Pe = 100 for different conductivity ratios of K = k_s/k_f. The presented scheme is compared with the Navier-Stokes solution from Malatip et al. [5]. The figure shows good agreement of both the solutions.



(a) Pe = 200



(b) Pe = 400

Figure 3. Comparative temperature distributions from the Navier-Stokes and nodeless variable finite element methods along the two walls and in the channel at (a) Pe = 200 and (b) Pe = 400.

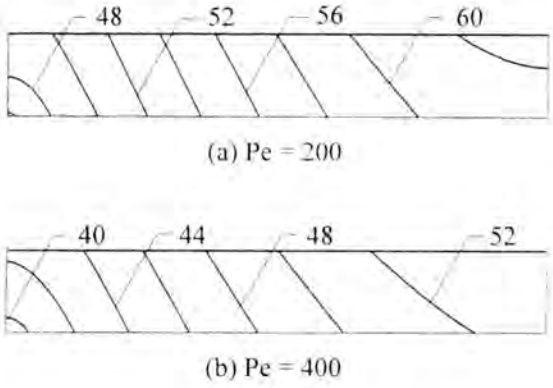


Figure 4. Predicted temperature contours from the nodeless variable finite element method at (a) Pe = 200 and (b) Pe = 400.

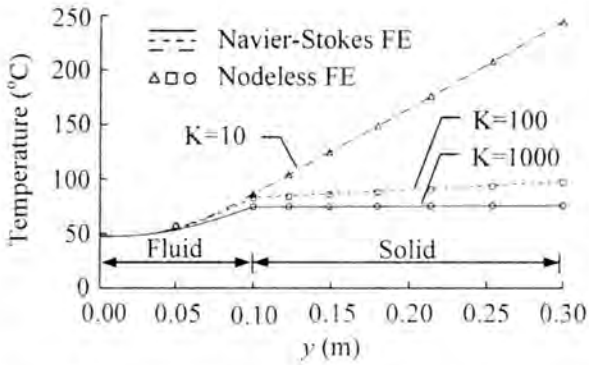


Figure 5. Comparative temperature distributions from the Navier-Stokes and nodeless variable finite element methods at $x = L$ and Pe = 100.

3.2 Plate with intense heating

To further evaluate the performance of the nodeless variable finite element method incorporated by the adaptive meshing technique, a plate subjected to intense heating is considered. The heating is simulated as a square width and the problem is considered into two cases. In the first case, the temperatures along the left, the right and the lower edges are constrained to zero. In the second case, convection heat transfer occurs from the lower edge of the plate to the fluid flow, while the left and the right edges are insulated.

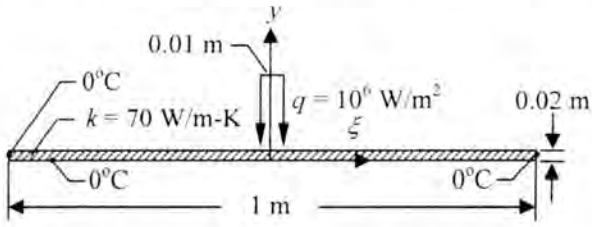


Figure 6. Problem statement of a plate subjected to intense heating.

In the first case as shown in Fig. 6, the exact plate temperature response can be calculated from [11].

$$T(\xi, y) = \frac{q}{Lk}\left\{\sum_{n=2,4}^{\infty}\frac{1}{\lambda_n^3}\left[\sin\left(\frac{n\pi\xi}{2L}\right)\frac{\sinh(\lambda_n y)}{\cosh(\lambda_n H)}\right] + \sum_{n=1,3}^{\infty}\frac{1}{\lambda_n^3}\left[\frac{n\pi}{L}\sin(\alpha)\right]\left[\cos\left(\frac{n\pi\xi}{2L}\right)\frac{\sinh(\lambda_n y)}{\cosh(\lambda_n H)}\right]\right\} \quad (15)$$

where the origin of the ξ - y coordinate system is shown in Fig. 6, q is the heat source, H is the plate width, k is the plate thermal conductivity. The parameter α and λ_n in Eq. (15) are defined by

$$\alpha = \frac{n\pi w}{2L} \quad (16)$$

$$\lambda_n = \sqrt{\frac{n^2\pi^2}{4L^2}} \quad (17)$$

where L is the plate length, and w is the width of heat source.

Figure 7 shows a structured finite element mesh model consisting of 5600 elements and 3208 nodes, and an adaptive mesh model that consists of 742 elements and 446 nodes. Table 1 compares the predicted peak temperatures obtained from the two finite element meshes using the convectional and nodeless variable finite element methods. The values in the brackets denote the percentage errors of the peak temperatures as compared to the exact solution. Table 1 shows that the adaptive mesh uses fewer elements than the structured mesh but provides higher solution accuracy.

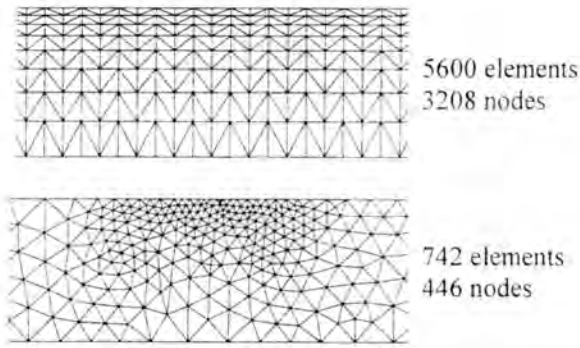


Figure 7. Structured and adaptive mesh models.

Figure 8 shows the adaptive mesh and the predicted temperature solution contours. Details of the adaptive mesh near the intense heating location and the temperature contours are shown in the lower figures. These figures show that small clustered elements are generated in the region of steep temperature gradients to capture the peak temperature and localized temperature distribution. At the same time, larger elements are generated in the other regions to reduce the computational time and the computer memory. The comparison of the exact and the predicted temperature distributions along the top edge is shown in Fig. 9. The

figure shows that the temperature distribution obtained from the adaptive nodeless variable finite element method is in good agreement with the exact solution.

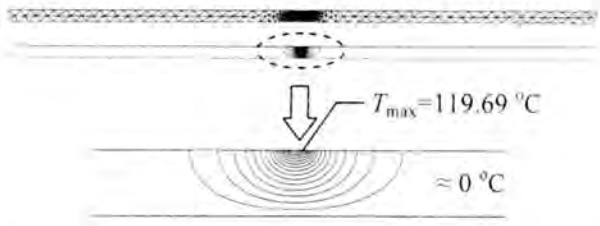


Figure 8. Adaptive mesh and the predicted temperature contours.

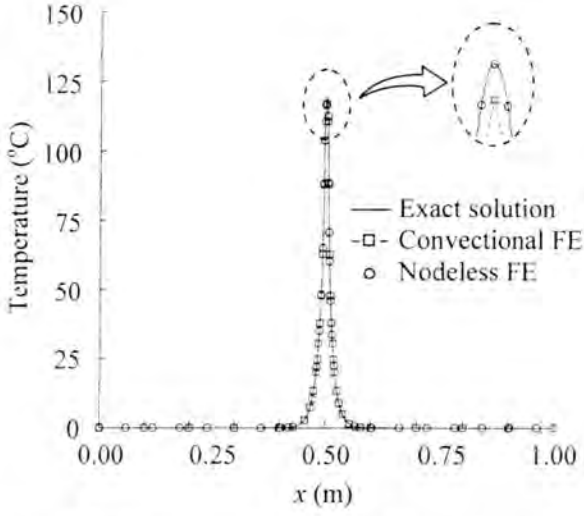


Figure 9. Comparison of the exact temperature and the predicted temperatures from the conventional method on structured mesh and the adaptive nodeless variable finite element methods.

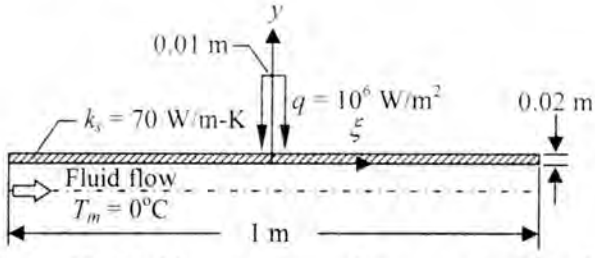


Figure 10. Problem statement of a convectively-cooled plate subjected to intense heating.

For the second case as shown by the problem statement in Fig. 10, a fully developed fluid flows beneath the lower edge of the plate while the other edges are insulated. The parameters of fluid used in the computation are as follows: the thermal conductivity k_f = 0.32 W/m-K, the specific heat c_f = 12,000 W/m-K, the density ρ = 54 kg/m^3, the mass flow rate $\dot{m}$ = 0.054 kg/s, the convection coefficient h = 929.52 W/m^2-k.

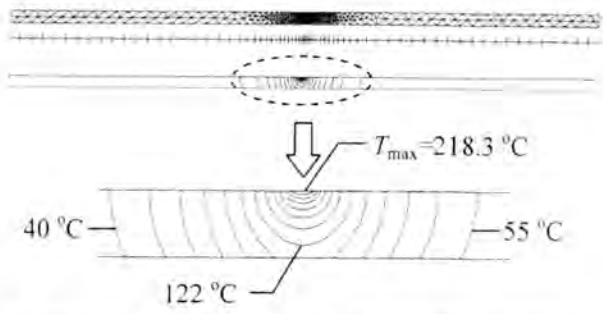


Figure 11. Adaptive mesh and the predicted temperature response for a convectively-cooled plate subjected to intense heating.

Figure 11 shows the adaptive mesh that consists of 1096 elements with 667 nodes and the predicted temperature contours. Details of the adaptive mesh near the intense heating location and the temperature contours are shown in the lower figures. At the heating location, the predicted peak temperatures are 218.30 °C and 217.91 °C from the nodeless variable and the conventional finite element methods. Figure 12 shows the temperature distributions along the top edge, the lower edge and in the channel obtained from the adaptive nodeless variable finite element method.

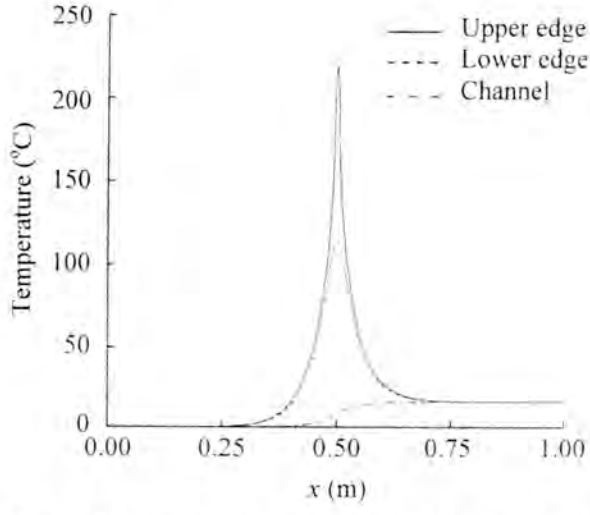


Figure 12. Predicted temperature distributions from the adaptive nodeless variable finite element method.

Table 1. Comparison of the predicted peak temperatures obtained from the conventional and the nodeless variable finite element methods on both the structured and adaptive meshes.

Mesh	Temperature (%Error)	
	Convectional FE	Nodeless FE
Structured	117.094 (2.169)	119.773 (0.070)
Adaptive	119.061 (0.526)	119.688 (0.002)

4. Conclusions

The adaptive nodeless variable finite element method for analysis of convectively-cooled solids was presented. The nodeless variable finite element is employed to improve the predicted solution without requiring actual nodes. The adaptive meshing technique was incorporated to reduce both the computer memory and computational time. Examples demonstrated that the adaptive nodeless variable finite element method can provide higher solution accuracy as compared to the conventional finite element technique.

5. Acknowledgement

The authors are pleased to acknowledge the Thailand Research Fund (TRF) for supporting this research work.

References

1. Thornton, E. A., and Dechaumphai, P., "Convective heat transport in merging flows" Third International Conference on Finite Elements in Water Resources, The University of Mississippi Oxford Campus, United States of America, 19-23 May 1980.
2. Wansophark, N., and Dechaumphai, P., 2002. Enhancement of Streamline Upwinding Finite Element Solution by Adaptive Meshing Technique. JSME International Journal, Vol. 45, No. 4, pp. 770-779.
3. Wansophark, N., and Dechaumphai, P., 2004. Combined Adaptive Meshing Technique and Segregated Finite Element Algorithm for Analysis of Free and Forced Convection Heat Transfer. Finite Elements in Analysis and Design, Vol. 40, pp. 645-663.
4. Thornton, E. A., and Dechaumphai, P., "Finite Element Analyses of Plane Thermal Entry-length Flows" Third International Conference on Finite Elements in Flow Problems, Banff, Alberta, Canada, 10-13 June 1980.
5. Malatip, A., Wansophark, N., and Dechaumphai, P., 2006. Combined Streamline Upwind Petrov Galerkin Method and Segregated Finite Element Algorithm for Conjugate Heat Transfer Problems. Journal of Mechanical Science and Technology, Vol. 20, No. 10, pp. 1741-1752.
6. Phongthanapanich, S., Traivivatana, S., Boonmaruth, P., and Dechaumphai, P., 2006. Nodeless Variable Finite Element Method for Heat Transfer Analysis by Means of Flux-Based Formulation and Mesh Adaption. Acta Mechanica Sinica, Vol. 22, No. 2, pp. 138-147.
7. Phongthanapanich, S., and Dechaumphai, P., 2008. Nodeless Variable Finite Element Method for Stress Analysis using Flux-Based Formulation. Journal of Mechanical Science and Technology, Vol. 22, No. 4, pp. 639-646.
8. Zienkiewicz, O. C., and Taylor, R. L., 2000. Finite Element Method, Volume 1, 5th ed. Butterworth-Heinemann, Woburn.
9. Huebner, K. A., Thornton, E. A., and Byrom, T. G., 1995. The Finite Element Method for Engineers, 3rd ed. John Wiley & Sons.
10. Kays, W. M, and Crawford, M. E., 1993. Convective Heat and Mass Transfer, 3rd ed. McGraw-Hill, Singapore.
11. McGowan, D. M., Carmarda, C. J., and Scotti, S. J., 1990. A Simplified Method of Thermal Analysis of a Cowl Leading Edge Subjected to Intense Localized Heating. NASA TP-16505.

ประวัติผู้เขียนวิทยานิพนธ์

นายสุทธิคมน์ พันธิมากรกิจ เกิดเมื่อวันที่ 13 เดือนสิงหาคม พุทธศักราช 2525 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิตจากภาควิชาเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี เมื่อปีการศึกษา 2545 เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2548