REFERENCES

Ajzoul, T. *et al.*, (1995). Finite element analysis of a transient nonlinear heat transfer problem. Computers and Chemical Engineers, 19(4), 423-436.

Betts, A.J. and Haroutunian V. (1983). A stream function finite element solution for two-dimensional natural convection. T., Kawai, (Ed.). Finite Element Flow Analysis. Japan: University of Tokyo

Caire, J. P. *et al*, (2001). Coupling a finite element code and a finite volume code for modeling of a fluorine electrolyzer. Proceedings of 6th World Congress of Chemical Engineering.

Dechaumphai, P., (1998). Numerical Methods in Engineering. Bangkok: Chulalongorn University.

Dechaumphai, P., (1999). Finite Element Method in Engineering. Bangkok: Chulalongorn University.

Fletcher, C.A.J., (1983). A comparison of the finite element and finite difference methods for computational fluid dynamics. T., Kawai (Ed.). Finite Element Flow Analysis. Japan: University of Tokyo

Hollman, J.P., (1992). Heat Transfer 7th Edition. Singapore: McGraw-Hill.

Incropera, F.P., and DeWitt, D.P. Introduction to Heat Transfer 3rd Edition. New York: John Wiley and Son.

Janphaisaeng, P., (1998). A finite element method for high speed compressible flow. Master Thesis of Mechanical Engineering, Chulalongkorn University.

Liu, Y. and Reitz, R.D., (1998). Modeling of heat conduction within chamber walls for multidimensional internal combustion engine simulations. International Journal of Heat and Mass Transfer, 41(6-7), 859-869.

Machura, M. and Sweet, R.A., (1980) A survey of software for partial differential equations. ACM Transactions on Mathematical Software, 6(4), 461-488.

Nicklin, D.J. *et al.* (1962). Two-phase flow in vertical tubes. Transactions of the Institution of Chemical Engineers, 40(1), 61-68.

Nicolai, B.M. and Baerdemaeker, J.D., (1999). A variance propagation algorithm for the computational of heat conduction under stochastic conditions. International Journal of Heat and Mass Transfer, 41, 1513-1520.

Sun, S. and Marrero, T.R., (1998). An object-oriented programming approach for heat and mass transfer related finite element analyses. Computers and Chemical Engineers, 22(10), 1381-1385.

Wilkes, J.O. and Bike, S.G., (1999). Fluid Mechanics for Chemical Engineers. New Jersey: Prentice-Hall.

Wilkes, J.O. and La Valle, P.P., (1990). Computer-aided understanding of physical principles in chemical engineering laboratory. Proceedings of ASEE Annual Conference.

Zienkiewicz, O. C. and Cheung, Y. K., (1965). Finite elements in the solution of field problems, The Engineer. 507-510

APPENDICES

Appendix A Green's Theorem

Green's theorem is presented here. It is very important for finite element method. The net result is the replacement of a volume integral, whose integrand typically involves second order derivatives, by another volume integral, whose integrand now involves only first order derivatives, plus an integral over the surface S that bounds the volume V:

$$\int_V \gamma\nabla\cdot k\nabla dV = -\int_V k\nabla\gamma\cdot\nabla u dV + \int_S \gamma k\frac{\partial u}{\partial n} dS . \qquad (A1)$$

Here, n denotes the outward normal to the surface S, u is dependent variable, and γ is a known basis function.

Proof of (A1) can be achieved by starting with the identity

$$\nabla\cdot\gamma q = \nabla\gamma\cdot q + \gamma\nabla\cdot q , \qquad (A2)$$

where γ is a scalar and q is a vector. Now, integrate over region V:

$$\int_V \nabla\cdot\gamma q dV = \int_V \nabla\gamma\cdot q dV + \int_V \gamma\nabla\cdot q dV . \qquad (A3)$$

from Gauss or divergence theorem ($\int_V \nabla\cdot w dV = \int_S w\cdot dS$), the left hand side of (A2), being the volume integral of the divergence of γq, can be replaced by the surface integral of the component of γq normal to the surface:

$$\int_S \gamma q\cdot dS = \int_V \nabla\gamma\cdot q dV + \int_V \gamma\nabla\cdot q dV . \qquad (A4)$$

Here the vector dS has a magnitude equal to the area dS of a surface element and a direction the same as the outward normal n. Let $q = -k\nabla u$, note that $\nabla u \cdot dS = (\partial u / \partial n)dS$, and rearrange to give the desired result:

$$\int_V \gamma\nabla \cdot k\nabla u dV = -\int_V k\nabla\gamma \cdot \nabla u dV + \int_S \gamma k \frac{\partial u}{\partial n} dS. \qquad (A5)$$

The following expansions of the scalar product $\nabla\gamma \cdot \nabla u$ will be needed when working in the indicated coordinates: rectangular,

$$\nabla\gamma \cdot \nabla u = \frac{\partial\gamma}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial \gamma}{\partial y}, \qquad (A6)$$

and axisymmetric cylindrical coordinate,

$$\nabla\gamma \cdot \nabla u = \frac{\partial\gamma}{\partial r}\frac{\partial u}{\partial r} + \frac{\partial u}{\partial z}\frac{\partial \gamma}{\partial z}. \qquad (A7)$$

Other useful formulas involving integrals of first derivatives in rectangular coordinate are,

$$\int_V \gamma \frac{\partial u}{\partial x} dV = -\int_V u \frac{\partial \gamma}{\partial x} dV + \int_S \gamma u n_x dS,$$

$$\int_V \gamma \frac{\partial u}{\partial y} dV = -\int_V u \frac{\partial \gamma}{\partial y} dV + \int_S \gamma u n_y dS. \qquad (A8)$$

And in axisymmetric cylindrical coordinate,

$$\int_V \gamma \frac{1}{r}\frac{\partial (ur)}{\partial r} dV = -\int_V u \frac{\partial \gamma}{\partial r} dV + \int_S \gamma u n_r dS,$$

$$\int_V \gamma \frac{\partial u}{\partial z} dV = -\int_V u \frac{\partial \gamma}{\partial z} dV + \int_S \gamma u n_z dS. \qquad (A8)$$

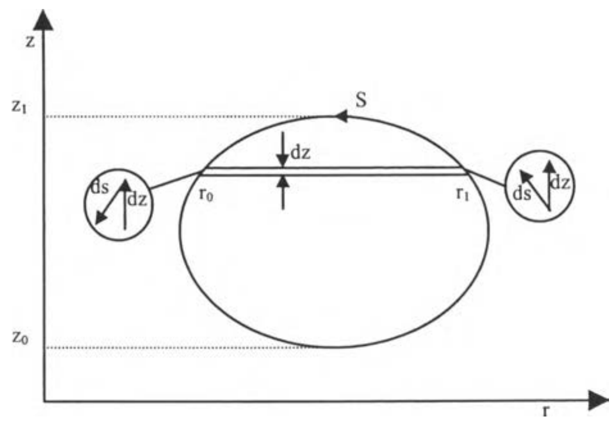

Figure A1 Geometry for proof of equation (A8)

Here, n_x, n_y, n_r, and n_z denote the direction cosines between the outward normal to S and the indicated coordinate directions x, y, r, and z.

Proof will be illustrated for the first of equations (A8). Noting that the volume element is dV = 2πrdrdz, start with

$$I = \int_V \gamma \frac{1}{r}\frac{\partial(ur)}{\partial r} 2\pi r dr dz, \qquad \text{(A9)}$$

and integrate by parts at constant z:

$$I = \int_{z_0}^{z_1} 2\pi\gamma ur\Big|_{r_0}^{r_1} dz - \int_z\int 2\gamma ur \frac{\partial\gamma}{\partial r} dr dz. \qquad \text{(A10)}$$

The first term on the right hand side may be rewritten as,

$$\int_{z_0}^{z_1} 2\pi(\gamma ur)_{r_1} dz + \int_{z_1}^{z_0} 2\pi(\gamma ur)_{r_0} dz = \int_S \gamma u n_r dS. \qquad \text{(A11)}$$

The last form being obtained by observing that $n_r dS$ equals $2\pi r_1 dz$ and $2\pi r_0 dz$ at the upper and lower limits, respectively. So,

$$I = \int_V \gamma \frac{1}{r}\frac{\partial(ur)}{\partial r} dV = \int_S \gamma u n_r dS - \int_V u \frac{\partial \gamma}{\partial r} dV . \qquad \text{(A12)}$$

and (A8) is proved.

After setting $\gamma = 1$ in (A7) and (A8), the following identities are immediately derived,

$$\int_V \frac{\partial u}{\partial x} dV = \int_S u n_x dS ; \qquad \int_V \frac{\partial u}{\partial y} dV = \int_S u n_y dS ; \qquad \text{(A13)}$$

$$\int_V \frac{1}{r}\frac{\partial(ur)}{\partial r} dV = \int_S u n_r dS ; \quad \int_V \frac{\partial u}{\partial z} dV = \int_S u n_z dS . \qquad \text{(A14)}$$

Appendix B All Program Interfaces

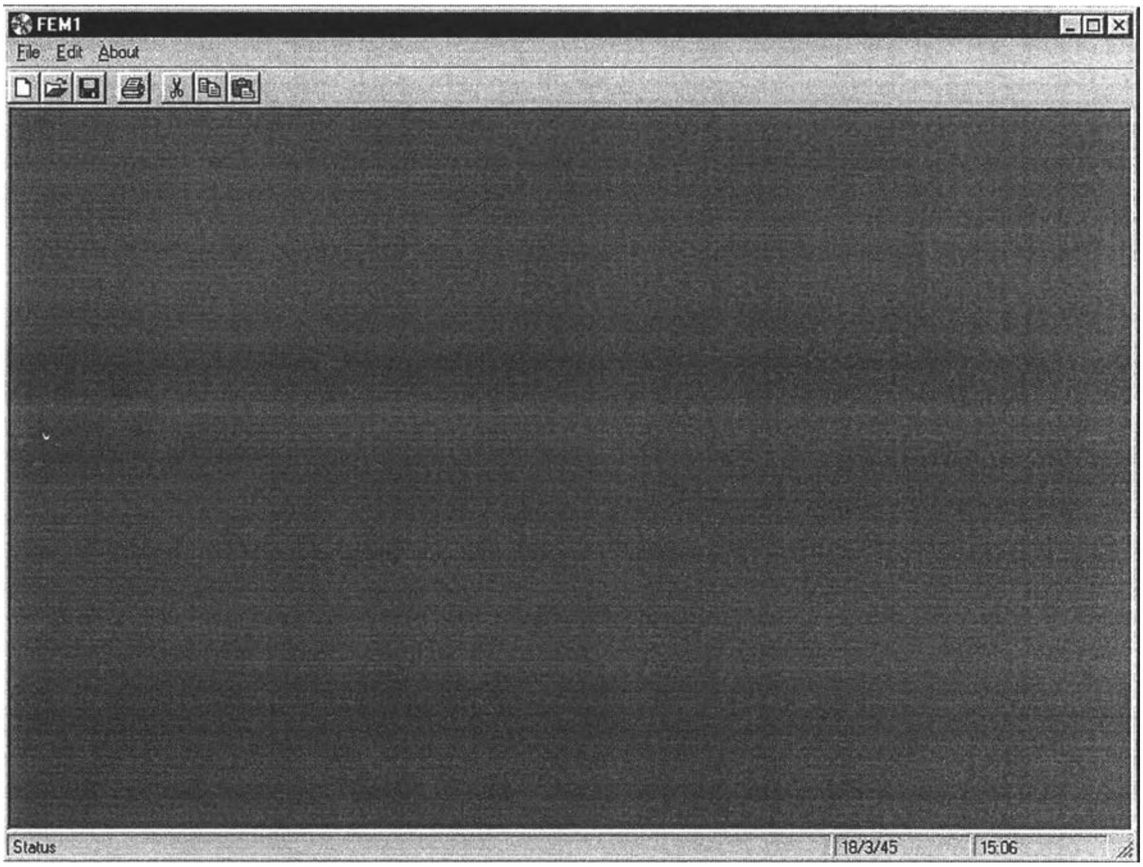

Figure B1 Form "frmMain"

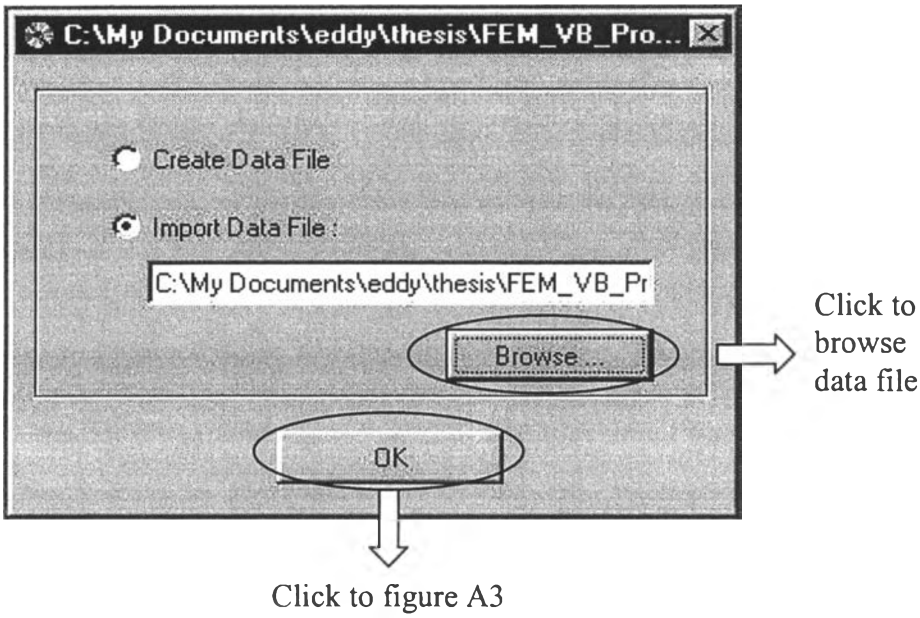

Figure B2 Form “frmInput”

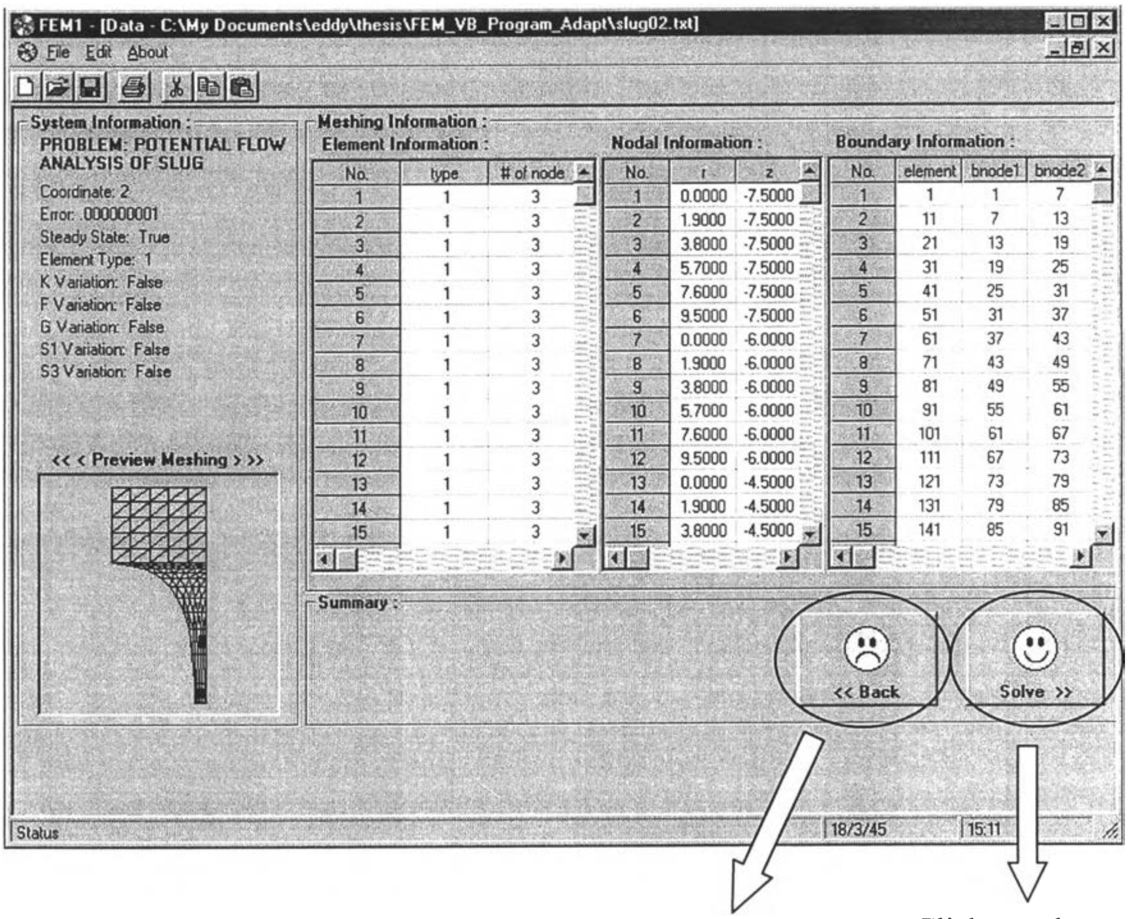

Figure B3 Form “frmFemWork”

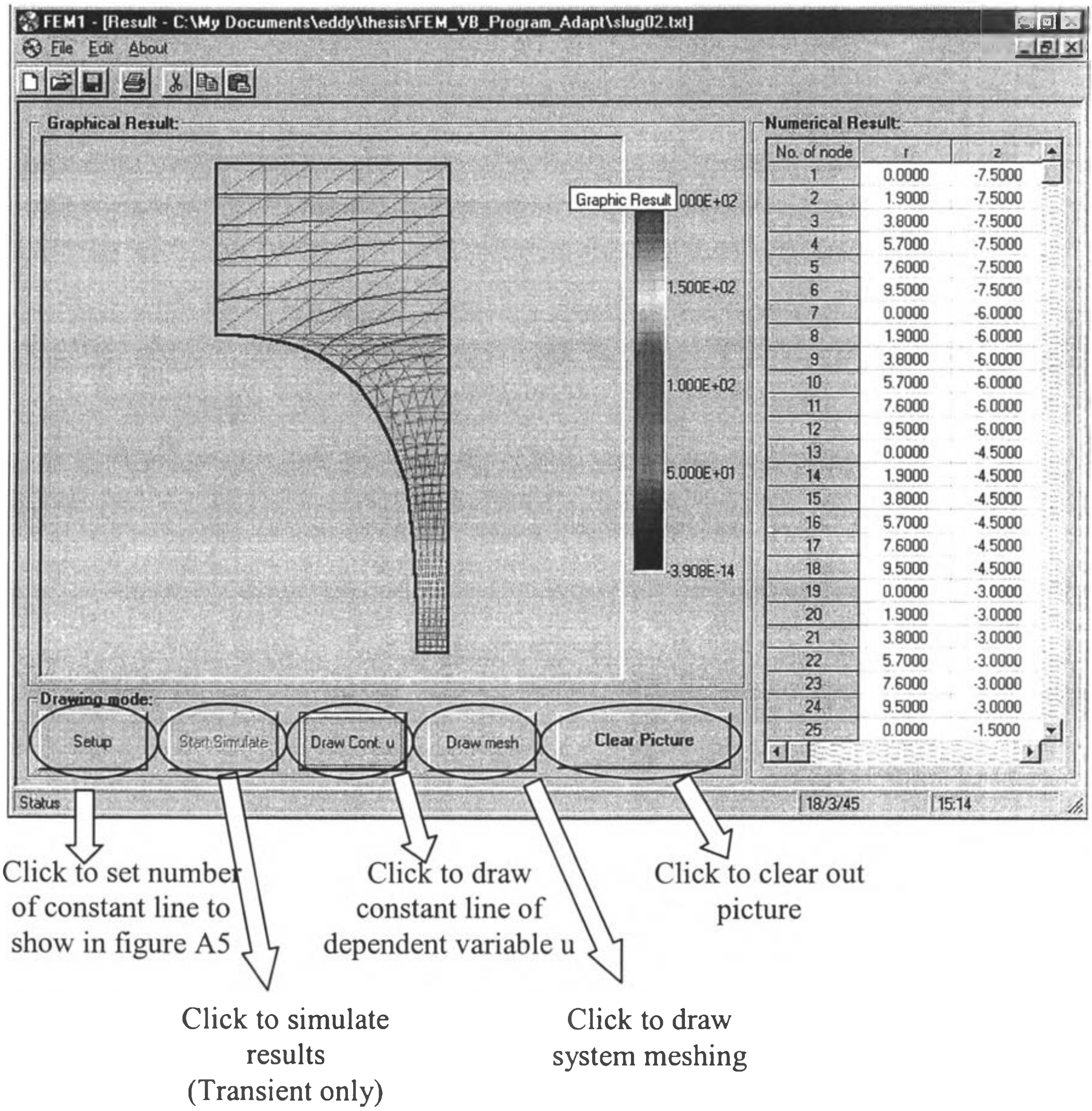

Figure B4 Form "frmResult"

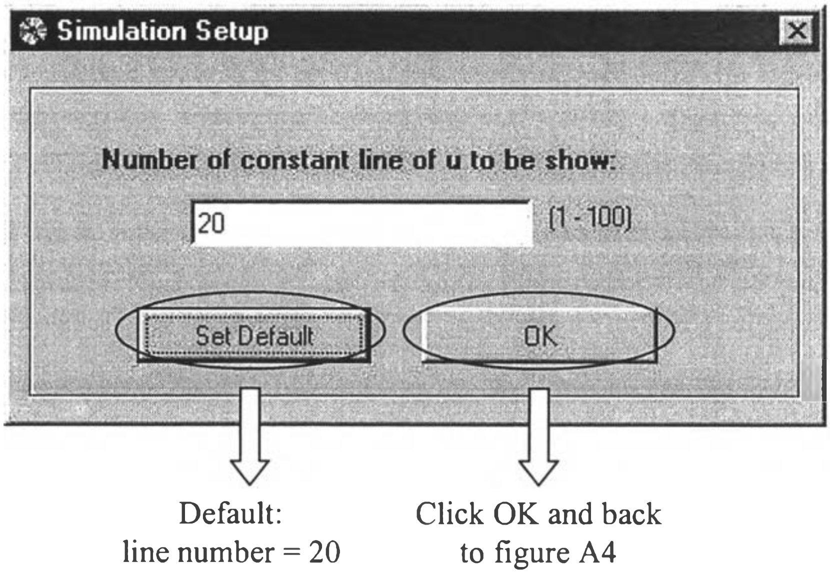

Figure B5 Form "frmSetpara"

Appendix C All Program Source Code for Each Form and Module

C1 Source Code of Form "frmMain"

```
Private Declare Function SendMessage Lib "User32" Alias "SendMessageA"
(ByVal hWnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam
As Any) As Long
Const EM_UNDO = &HC7


Private Sub MDIForm_Load()
   Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
   Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
   Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
   Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)
   LoadNewWork
End Sub


Private Sub LoadNewWork()
   Static lCount As Long
   Dim frmD As frmInput
   lCount = lCount + 1
   Set frmD = New frmInput
   frmD.Caption = "work" & lCount
   frmD.Show
End Sub


Private Sub MDIForm_Unload(Cancel As Integer)
   If Me.WindowState <> vbMinimized Then
      SaveSetting App.Title, "Settings", "MainLeft", Me.Left
      SaveSetting App.Title, "Settings", "MainTop", Me.Top
      SaveSetting App.Title, "Settings", "MainWidth", Me.Width
      SaveSetting App.Title, "Settings", "MainHeight", Me.Height
   End If
End Sub


Private Sub tbToolBar_ButtonClick(ByVal Button As MSComCtlLib.Button)
   On Error Resume Next
   Select Case Button.Key
      Case "New"
         LoadNewWork
      Case "Cut"
         mnuEditCut_Click
      Case "Copy"
         mnuEditCopy_Click
      Case "Paste"
         mnuEditPaste_Click
   End Select
End Sub


Private Sub mnuAboutAbout_Click()
   frmAbout.Show vbModal, Me
```

```
End Sub


Private Sub mnuEditPaste_Click()
   On Error Resume Next
   ActiveForm.rtfText.SelRTF = Clipboard.GetText
End Sub


Private Sub mnuEditCopy_Click()
   On Error Resume Next
   Clipboard.SetText ActiveForm.rtfText.SelRTF
End Sub


Private Sub mnuEditCut_Click()
   On Error Resume Next
   Clipboard.SetText ActiveForm.rtfText.SelRTF
   ActiveForm.rtfText.SelText = vbNullString
End Sub


Private Sub mnuFileExit_Click()
'  ... unload program ...
   Unload Me
End Sub


Private Sub mnuFileNew_Click()
   On Error Resume Next
   Unload frmFemwork
   Unload frmInput
   Unload frmResult
   Unload frmSetPara
   Unload frmSlug
   LoadNewWork
End Sub
```

C2 Source Code of Form "frmInput"

```
Private Sub cmdOK_Click()
   If OptImport = True Then
      If Txtbox.Text = "" Then
         MsgBox "!!!  Please enter file name (or browse) or select create
data file.", vbExclamation
      Else
         Me.Visible = False
         frmFemwork.Show
      End If
   Else
      MsgBox "!!!This part is not available now", vbExclamation
   End If
End Sub
```

```
Private Sub CmdBrowse_Click()

   With DlgCommonDialog1
      .DialogTitle = "Open"
      .CancelError = False
      .Filter = "Text Files |*.txt"
      .ShowOpen
      If Len(.FileName) = 0 Then
         Exit Sub
      End If
      sFile = .FileName
   End With
   Txtbox.Text = sFile
   Me.Caption = sFile

End Sub


Private Sub OptImport_click()
   Txtbox.Enabled = True
End Sub


Private Sub OptCreate_click()
   Txtbox.Enabled = False
End Sub
```

C3 Source Code of Form "frmFemWork"

```
Option Explicit

Private ALPHA() As Double
Private BCTYPE() As Integer
Private BETA() As Double
Private BNODE() As Integer
Private COORDS As Integer
Private DT As Single
Private E As Integer
Private ELEMS As Integer
Private EMATL() As Integer
Private EPS As Double
Private ES() As Integer
Private ETYPE() As Integer
Private F() As Double
Private FVAR As Boolean
Private G() As Double
Private GVAR As Boolean
Private I, J As Integer
Private K() As Double
Private KVAR As Boolean
Private NBSEGS As Integer
Private NELEMS As Integer
Private ND() As Boolean
```

```
Private NNODES As Integer
Private NODE() As Integer
Private NTOTEL() As Integer
Private PICSCALEX1 As Single
Private PICSCALEX2 As Single
Private PICSCALEY1 As Single
Private PICSCALEY2 As Single
Private PRNTFR As Integer
Private SEGTYP() As Integer
Private SMATL() As Integer
Private SSTATE As Boolean
Private S1VAR As Boolean
Private S3VAR As Boolean
Private TMAX As Single
Private U() As Double
Private U1() As Double
Private X() As Double
Private XMAX As Double
Private XMIN As Double
Private Y() As Double
Private YMAX As Double
Private YMIN As Double


Private Sub Draw_mesh()
'  ... This subroutine is to draw mesh in picture box ...
'  ... set scale for picture box ...
   If (XMAX - XMIN) > (YMAX - YMIN) Then
      PICSCALEX1 = XMIN - Abs(0.05 * (XMAX - XMIN))
      PICSCALEX2 = XMAX + Abs(0.05 * (XMAX - XMIN))
      PICSCALEY1 = (YMIN + YMAX) / 2# - Abs(PICSCALEX1 - PICSCALEX2) / 2#
      PICSCALEY2 = (YMIN + YMAX) / 2# + Abs(PICSCALEX1 - PICSCALEX2) / 2#
      PicMesh.Scale (PICSCALEX1, PICSCALEY1)-(PICSCALEX2, PICSCALEY2)
   Else
      PICSCALEY1 = YMIN - Abs(0.05 * (YMAX - YMIN))
      PICSCALEY2 = YMAX + Abs(0.05 * (YMAX - YMIN))
      PICSCALEX1 = (XMIN + XMAX) / 2# - Abs(PICSCALEY1 - PICSCALEY2) / 2#
      PICSCALEX2 = (XMIN + XMAX) / 2# + Abs(PICSCALEY1 - PICSCALEY2) / 2#
      PicMesh.Scale (PICSCALEX1, PICSCALEY1)-(PICSCALEX2, PICSCALEY2)
   End If
   If COORDS = 1 Then PicMesh.Scale (PICSCALEX1, PICSCALEY2)- _
                                    (PICSCALEX2, PICSCALEY1)

'  ... start drawing loop ...
   For E = 1 To NELEMS
      PicMesh.Line (X(NODE(1, E)), Y(NODE(1, E)))- _
                   (X(NODE(2, E)), Y(NODE(2, E)))
      PicMesh.Line (X(NODE(2, E)), Y(NODE(2, E)))- _
                   (X(NODE(3, E)), Y(NODE(3, E)))
      PicMesh.Line (X(NODE(3, E)), Y(NODE(3, E)))- _
                   (X(NODE(1, E)), Y(NODE(1, E)))
   Next E
End Sub
```

```
Private Sub CmdBack_Click()
   Unload Me
   frmInput.Show
End Sub


Private Sub CmdSolve_Click()
'  ... start calculation module & show frame result ...
   Call Calculation.MainProg(ALPHA, BCTYPE, BETA, BNODE, COORDS, DT, _
        ELEMS, EMATL, EPS, ES, ETYPE, F, FVAR, G, GVAR, K, KVAR, _
        NBSEGS, NELEMS, ND, NNODES, NODE, NTOTEL, PICSCALEX1, _
        PICSCALEX2, PICSCALEY1, PICSCALEY2, PRNTFR, SEGTYP, SMATL, _
        SSTATE, S1VAR, S3VAR, TMAX, U, U1, X, Y)
   Unload Me
End Sub


Private Sub Form_Load()

   Me.Caption = "Data - " & sFile

'  ... Open specific data file for inputdata and then close file ...
   Open sFile For Input As #1
   Call DataInput
   Close #1

'  ... Call subroutine show all system information in table ...
   Call DataShow

'  ... Call subroutine to preview system meshing in picture box ...
   Call Draw_mesh

End Sub


Private Sub DataShow()
Dim DAT(10) As String

'  ... Format all labels in frame ...
   lblProblem.Caption = "PROBLEM: " & NAME_TITLE
   Label1.Caption = "Coordinate: " & COORDS
   Label2.Caption = "Error: " & EPS
   Label3.Caption = "Steady State: " & Str(SSTATE)
   Label4.Caption = "Element Type: " & Str(ELEMS)
   Label5.Caption = "K Variation: " & Str(KVAR)
   Label6.Caption = "F Variation: " & Str(FVAR)
   Label7.Caption = "G Variation: " & Str(GVAR)
   Label8.Caption = "S1 Variation: " & Str(S1VAR)
   Label9.Caption = "S3 Variation: " & Str(S3VAR)
   If Not SSTATE Then
      Label14.Caption = "DT: " & Str(DT)
      Label15.Caption = "PRTFR: " & Str(PRNTFR)
      Label16.Caption = "TMAX: " & Str(TMAX)
   End If

'  ... show element information in table named "GrdElem" ...
```

```
' ... formatting and heading the table ...
For I = 0 To GrdElem.Cols - 1
   GrdElem.ColAlignment(I) = 4
   GrdElem.ColWidth(I) = 950
Next I
DAT(0) = "No."
DAT(1) = "type"
DAT(2) = "# of node"
DAT(3) = "Material"
DAT(4) = "K"
DAT(5) = "F"
DAT(6) = "G"
DAT(7) = "node1"
DAT(8) = "node2"
DAT(9) = "node3"
GrdElem.Row = 0
For I = 0 To 9
   GrdElem.Col = I
   GrdElem.Text = DAT(I)
Next I
' ... show element information ...
GrdElem.Rows = NELEMS + 1
For E = 1 To NELEMS
   GrdElem.Row = E
   DAT(0) = Str(E)
   DAT(1) = Str(ETYPE(E))
   DAT(2) = Str(NTOTEL(E))
   DAT(3) = Str(EMATL(E))
   DAT(4) = Format(K(E), "0.00E+00")
   DAT(5) = Format(F(E), "0.00E+00")
   DAT(6) = Format(G(E), "0.00E+00")
   DAT(7) = Str(NODE(1, E))
   DAT(8) = Str(NODE(2, E))
   DAT(9) = Str(NODE(3, E))
   For I = 0 To 9
      GrdElem.Col = I
      GrdElem.Text = DAT(I)
   Next I
Next E

' ... show nodal information in table named "GrdNode" ...
' ... table formatting and heading ...
With GrdNode
   If Not SSTATE Then .Cols = 4
   For I = 0 To .Cols - 1
      .ColAlignment(I) = 4
      .ColWidth(I) = 700
   Next I
   .Row = 0
   .Col = 0
   .Text = "No."
   .Col = 1
   If (COORDS = 1) Then .Text = "x" Else .Text = "r"
   .Col = 2
   If (COORDS = 1) Then .Text = "y" Else .Text = "z"
   If Not SSTATE Then
```

```
         .Col = 3
         .Text = "initial u"
      End If
      .Rows = NNODES + 1
'     ... show nodal information ...
      For I = 1 To NNODES
         .Row = I
         .Col = 0
         .Text = I
         .Col = 1
         .Text = Format(X(I), "#0.0000")
         .Col = 2
         .Text = Format(Y(I), "#0.0000")
         If (Not SSTATE) Then
            .Col = 3
            .Text = Format(U(I), "0.000E+00")
         End If
      Next I
   End With

'  ... show boundary information in table named "GrdBond" ...
'  ... table formatting and heading ...
   For I = 0 To 6
      GrdBond.ColAlignment(I) = 4
      GrdBond.ColWidth(I) = 700
   Next I
   GrdBond.ColWidth(5) = 900
   GrdBond.ColWidth(6) = 900
   DAT(0) = "No."
   DAT(1) = "element"
   DAT(2) = "bnode1"
   DAT(3) = "bnode2"
   DAT(4) = "type"
   DAT(5) = "A"
   DAT(6) = "B"
   GrdBond.Row = 0
   For I = 0 To 6
      GrdBond.Col = I
      GrdBond.Text = DAT(I)
   Next I
'  ... show boundary information ...
   GrdBond.Rows = NBSEGS + 1
   For I = 1 To NBSEGS
      DAT(0) = I
      DAT(1) = ES(I)
      DAT(2) = BNODE(1, I)
      DAT(3) = BNODE(2, I)
      DAT(4) = BCTYPE(I)
      If (BCTYPE(I) = 1) Then
         DAT(5) = Format(U1(BNODE(1, I)), "0.00E+00")
         DAT(6) = "-"
      ElseIf (BCTYPE(I) = 3) Then
         DAT(5) = Format(ALPHA(I), "0.00E+00")
         DAT(6) = Format(BETA(I), "0.00E+00")
      Else
         DAT(5) = "-"
```

```
         DAT(6) = "-"
      End If
      GrdBond.Row = I
      For J = 0 To 6
         GrdBond.Col = J
         GrdBond.Text = DAT(J)
      Next J
   Next I

End Sub


Private Sub DataInput()
Dim AA As Double
Dim BB As Double
Dim NONE As String
Dim NTOT As Integer

   On Error GoTo ERROR1

'  ... input system variable ...
   Line Input #1, NAME_TITLE
   GoSub Skipline
   NONE = Input(9, #1)
   COORDS = Val(Input(11, #1))
   NONE = Input(11, #1)
   EPS = Val(Input(11, #1))
   NONE = Input(11, #1)
   KVAR = CBool(Val(Input(11, #1)))
   NONE = Input(11, #1)
   FVAR = CBool(Val(Input(11, #1)))
   NONE = Input(11, #1)
   GVAR = CBool(Val(Input(11, #1)))
   NONE = Input(11, #1)
   SSTATE = CBool(Val(Input(11, #1)))
   NONE = Input(11, #1)
   If Not SSTATE Then
      DT = Val(Input(11, #1))
      NONE = Input(11, #1)
      PRNTFR = Val(Input(11, #1))
      NONE = Input(11, #1)
      TMAX = Val(Input(11, #1))
      NONE = Input(11, #1)
   End If
   S1VAR = CBool(Val(Input(11, #1)))
   NONE = Input(11, #1)
   S3VAR = CBool(Val(Input(11, #1)))
   NONE = Input(11, #1)
   ELEMS = Val(Input(11, #1))
   NONE = Input(11, #1)
   NELEMS = Val(Input(11, #1))
   NONE = Input(11, #1)
   NNODES = Val(Input(11, #1))
   NONE = Input(11, #1)
   NBSEGS = Val(Input(11, #1))
   NONE = Input(2, #1)
```

```
' ... Redimension for dinamic variables ...
ReDim ALPHA(NBSEGS) As Double
ReDim BCTYPE(NBSEGS) As Integer
ReDim BETA(NBSEGS) As Double
ReDim BNODE(4, NBSEGS) As Integer
ReDim EMATL(NELEMS) As Integer
ReDim ES(NBSEGS) As Integer
ReDim ETYPE(NELEMS) As Integer
ReDim F(NELEMS) As Double
ReDim G(NELEMS) As Double
ReDim K(NELEMS) As Double
ReDim ND(NNODES) As Boolean
ReDim NODE(8, NELEMS) As Integer
ReDim NTOTEL(NELEMS) As Integer
ReDim SEGTYP(NBSEGS) As Integer
ReDim SMATL(NBSEGS) As Integer
ReDim U(NNODES) As Double
ReDim U1(NNODES) As Double
ReDim X(NNODES) As Double
ReDim Y(NNODES) As Double

' ... input element information from data file ...
GoSub Skipline
For E = 1 To NELEMS
   NONE = Input(4, #1)
   ETYPE(E) = Val(Input(2, #1))
   NTOTEL(E) = Val(Input(2, #1))
   EMATL(E) = Val(Input(2, #1))
   K(E) = Val(Input(10, #1))
   F(E) = Val(Input(10, #1))
   G(E) = Val(Input(10, #1))
   For I = 1 To NTOTEL(E)
      NODE(I, E) = Val(Input(4, #1))
   Next I
   NONE = Input(2, #1)
Next E

' ... input nodal information ...
XMAX = 0#
XMIN = 0#
YMAX = 0#
YMIN = 0#
GoSub Skipline
For I = 1 To NNODES
   NONE = Input(5, #1)
   X(I) = Val(Input(10, #1))
   Y(I) = Val(Input(10, #1))
   If Not SSTATE Then U(I) = Val(Input(10, #1))
   NONE = Input(2, #1)
'  ... Determine maximum and minimum X and Y ...
   If X(I) > XMAX Then XMAX = X(I)
   If X(I) < XMIN Then XMIN = X(I)
   If Y(I) > YMAX Then YMAX = Y(I)
   If Y(I) < YMIN Then YMIN = Y(I)
Next I
```

```
'  ... input boundary information ...
   For I = 1 To NNODES
      ND(I) = False
   Next I
   GoSub Skipline
   For I = 1 To NBSEGS
      NONE = Input(5, #1)
      SEGTYP(I) = Input(5, #1)
      NTOT = Input(5, #1)
      SMATL(I) = Input(5, #1)
      BCTYPE(I) = Input(5, #1)
      AA = Input(10, #1)
      BB = Input(10, #1)
      ES(I) = Input(5, #1)
      For J = 1 To NTOT
         BNODE(J, I) = Input(5, #1)
      Next J
      NONE = Input(2, #1)
      If (BCTYPE(I) = 1) Then
         For J = 1 To NTOT
            U1(BNODE(J, I)) = AA
            ND(BNODE(J, I)) = True
         Next J
      ElseIf (BCTYPE(I) = 3) Then
         ALPHA(I) = AA
         BETA(I) = BB
      End If
   Next I
   Exit Sub

Skipline:
   For I = 1 To 4
      Line Input #1, NONE
   Next I
Return

ERROR1:
   MsgBox "An error occures in reading data file, please check file", _
          vbExclamation
   Unload Me
End Sub
```

C4 Source Code of Form "frmResult"

```
Option Explicit

'.. Declare global variable for use in this form ...
Private BNODE() As Integer
Private COORDS As Integer
Private I, J As Integer
Private LINENUM As Integer
Private N As Integer
Private NBSEGS As Integer
```

```
Private NELEMS As Integer
Private NNODES As Integer
Private NODE() As Integer
Private SSTATE As Boolean
Private STEP As Single
Private U() As Double
Private UIN As Single
Private UMIN As Double
Private UMAX As Double
Private X() As Double
Private Y() As Double


Sub PrepareData(P1, P2, P3, P4, P5, P6, P7, P8, P9, PICSCALEX1, _
                PICSCALEX2, PICSCALEY1, PICSCALEY2)
'  ... This subroutine is to get all result from calculation module
'      and prepare them to exhibit in table or even simulate ...

'  ... Get these individual data ...
   COORDS = P1
   NBSEGS = P2
   NELEMS = P3
   NNODES = P4
   SSTATE = P5

'  ... Set dimension for individual data ...
   ReDim NODE(8, NELEMS) As Integer
   ReDim BNODE(4, NBSEGS) As Integer
   ReDim U(NNODES) As Double
   ReDim X(NNODES) As Double
   ReDim Y(NNODES) As Double

'  ... Get value node, bnode, x, and y ...
   For I = 1 To NELEMS
      For J = 1 To 3
        NODE(J, I) = P6(J, I)
      Next J
   Next I

   For I = 1 To NBSEGS
      For J = 1 To 3
        BNODE(J, I) = P7(J, I)
      Next J
   Next I

   For I = 1 To NNODES
      X(I) = P8(I)
      Y(I) = P9(I)
   Next I

'  ... Set scale for picture box ...

   If COORDS = 1 Then
      PicGraphic.Scale (PICSCALEX1, PICSCALEY2)-(PICSCALEX2, PICSCALEY1)
   Else
      PicGraphic.Scale (PICSCALEX1, PICSCALEY1)-(PICSCALEX2, PICSCALEY2)
```

```
   End If

'  ... Set general properties ...
   If SSTATE Then
      CmdSimul.Enabled = False
      CmdContU.Enabled = True
   End If
   Me.Caption = "Result - " & sFile

'  ... Set line number of constant u to be shown ...
   Call SetPara(20)

'  ... Report numerical data in table ...
   Call ReportData

'  ... Show system meshing in picture box ...
   Call DrawMesh(0, 0, 0)
   Call DrawBoundary

End Sub


Private Sub DrawBoundary()
'  ... draw boundary segments ...
   PicGraphic.DrawWidth = 2
   For I = 1 To NBSEGS
      PicGraphic.Line (X(BNODE(1, I)), Y(BNODE(1, I)))- _
                      (X(BNODE(2, I)), Y(BNODE(2, I)))
   Next I
   PicGraphic.DrawWidth = 1
End Sub


Private Sub DrawContU()
'  ... This subroutine is used to draw constant line of u = UN
'      at specific colour of R(red), G(green), and B(blue) ...
Dim BLUE As Single
Dim E As Integer
Dim GREEN As Single
Dim N1 As Integer
Dim N2 As Integer
Dim N3 As Integer
Dim RED As Single
Dim UN As Double
Dim XN(2) As Double
Dim YN(2) As Double

   For I = 1 To LINENUM
      UN = I * STEP
'     ... set color parameter for each line ...
      If UN <= UIN / 4# Then
         BLUE = 255#
         GREEN = 255# * 4# * UN / UIN
         RED = 0#
      ElseIf UIN / 4# < UN And UN <= UIN / 2# Then
         BLUE = 255# * (1# - (UN * 4# / UIN - 1#))
```

```
         GREEN = 255#
         RED = 0#
      ElseIf UIN / 2# < UN And UN <= UIN * 3# / 4# Then
         BLUE = 0#
         GREEN = 255#
         RED = 255# * (UN * 4# / UIN - 2#)
      Else
         BLUE = 0#
         GREEN = 255# * (1# - (UN * 4# / UIN - 3#))
         RED = 255#
      End If
      UN = UN + UMIN
'     ... start to draw line ...
'     ... find which element has u equal to UN inside and then draw
'         constant UN line ...
      For E = 1 To NELEMS
         J = 1
         N1 = NODE(1, E)
         N2 = NODE(2, E)
         N3 = NODE(3, E)

'        ... if any two edges in element (triangular element) has value of
'            UN, we can interpolate to find constant line of that UN ...
         If (U(N1) < UN And UN < U(N2)) Or (U(N1) > UN And UN > U(N2)) Then
            XN(J) = X(N1) + (X(N2) - X(N1)) * ((UN - U(N1)) / _
                                               (U(N2) - U(N1)))
            YN(J) = Y(N1) + (Y(N2) - Y(N1)) * ((UN - U(N1)) / _
                                               (U(N2) - U(N1)))
            J = J + 1
         End If
         If (U(N2) < UN And UN < U(N3)) Or (U(N2) > UN And UN > U(N3)) Then
            XN(J) = X(N2) + (X(N3) - X(N2)) * ((UN - U(N2)) / _
                                               (U(N3) - U(N2)))
            YN(J) = Y(N2) + (Y(N3) - Y(N2)) * ((UN - U(N2)) / _
                                               (U(N3) - U(N2)))
            J = J + 1
         End If
         If J = 2 Then
            XN(J) = X(N3) + (X(N1) - X(N3)) * ((UN - U(N3)) / _
                                               (U(N1) - U(N3)))
            YN(J) = Y(N3) + (Y(N1) - Y(N3)) * ((UN - U(N3)) / _
                                               (U(N1) - U(N3)))
         End If

'        ... Draw line at specific colour ...
         PicGraphic.Line (XN(1), YN(1))-(XN(2), YN(2)), _
                         RGB(RED, GREEN, BLUE)
      Next E
   Next I
End Sub


Sub DrawMesh(BLUE, GREEN, RED)
Dim E As Integer
'  ... This subroutine is to draw mesh in picture box ...
   PicGraphic.Cls
```

```
   For E = 1 To NELEMS
      PicGraphic.Line (X(NODE(1, E)), Y(NODE(1, E)))-(X(NODE(2, E)), _
                       Y(NODE(2, E))), RGB(RED, GREEN, BLUE)
      PicGraphic.Line (X(NODE(2, E)), Y(NODE(2, E)))-(X(NODE(3, E)), _
                       Y(NODE(3, E))), RGB(RED, GREEN, BLUE)
      PicGraphic.Line (X(NODE(3, E)), Y(NODE(3, E)))-(X(NODE(1, E)), _
                       Y(NODE(1, E))), RGB(RED, GREEN, BLUE)
   Next E
End Sub


Private Sub CmdClear_Click()
'  ... Clear picture in picture box ...
   PicGraphic.Cls
End Sub


Private Sub CmdContU_Click()
'  ... After click botton "draw cont u", this subroutine prepare data
'      to draw constant line of dependent variable, which including
'      value of u foe each line (UN) and its colour ...

'  ... Clear picture and draw mesh first ...
   PicGraphic.Cls
   Call DrawMesh(150, 150, 150)

'  ... find value of u (UN) for each line ...
   UIN = (UMAX - UMIN)
   STEP = UIN / (LINENUM + 1)

'  ... draw constant u lines by call subroutine "DrawContU" ...
   Call DrawContU

'  ... draw system bounbaries ...
   Call DrawBoundary

End Sub


Private Sub CmdMesh_Click()
'  ... Show system meshing in picture box ...
   Call DrawMesh(0, 0, 0)
   Call DrawBoundary
End Sub


Private Sub ReportData()
'  ... This subroutine is to report node information in table named
'      "GrdResult" ...
'  ... Table heading ...
   With GrdResult
      .Row = 0
      .Col = 0
      .Text = "No. of node"
      .Col = 1
      If (COORDS = 1) Then
```

```
            .Text = "x"
            .Col = 2
            .Text = "y"
        Else
            .Text = "r"
            .Col = 2
            .Text = "z"
        End If

        .Rows = NNODES + 1
        For I = 0 To .Cols - 1
            .ColAlignment(I) = 4
        Next I

'       ... Show number of node and value of x and y for each node ...
        For I = 1 To NNODES
            .Row = I
            .Col = 0
            .Text = I
            .Col = 1
            .Text = Format(X(I), "#0.0000")
            .Col = 2
            .Text = Format(Y(I), "#0.0000")
        Next I
    End With
End Sub


Sub ShowSol(SWITCH, T, TCOUNT, P1)
Dim N As Integer

'   ... Add new column to table "GrdResult" and give format ...
    With GrdResult
        N = .Cols + 1
        .Cols = N
        .ColAlignment(N - 1) = 4
        .Col = N - 1
        .Row = 0
        Select Case SWITCH
            Case 1
                .Text = "u at t=" & T
            Case 2
                .Text = "solution u"
        End Select

'       ... print solutions into table ...
        For I = 1 To NNODES
            U(I) = P1(I)
            .Row = I
            .Text = Format(U(I), "0.0000E+00")
        Next I

'       ... find umin and umax ...
        UMAX = U(1)
        UMIN = U(1)
        For I = 1 To NNODES
```

```
         If U(I) > UMAX Then UMAX = U(I)
         If U(I) < UMIN Then UMIN = U(I)
      Next I
   End With

'  ... for transient problem, all solution will be kept in table
'      "grdInvisTable" for simulation ...
   If Not SSTATE Then
      With grdInvisTable
         .Rows = NNODES + 1
         .Cols = .Cols + 1
         .Col = .Cols - 1
         For I = 1 To NNODES
            .Row = I
            .Text = U(I)
         Next I
      End With
   End If

'  ... specify label to present value of U for different color ...
   Label1.Caption = Format(UMAX, "0.000E+00")
   Label2.Caption = Format((UMAX - UMIN) * 3# / 4# + UMIN, "0.000E+00")
   Label3.Caption = Format((UMAX - UMIN) / 2# + UMIN, "0.000E+00")
   Label4.Caption = Format((UMAX - UMIN) / 4# + UMIN, "0.000E+00")
   Label5.Caption = Format(UMIN, "0.000E+00")

End Sub


Private Sub CmdSetPara_Click()
   frmSetPara.Show
   frmSetPara.txtNumber.Text = LINENUM
End Sub


Sub SetPara(P1)
'  ... Get setting parameter ...
   LINENUM = P1
End Sub


Private Sub CmdSimul_Click()
'  ... when start, set other bottons to be deactivate ...
   If CmdSimul.Caption = "Start Simulate" Then
      CmdSimul.Font.Bold = True
      CmdSimul.Caption = "Stop"
      CmdSetPara.Enabled = False
      CmdClear.Enabled = False
      CmdMesh.Enabled = False

'     ... Read initial U(i) from grdInvisTable before show ...
      With grdInvisTable
         .Col = 1
         For I = 1 To NNODES
            .Row = I
            U(I) = Val(.Text)
```

```
         Next I
      End With
      UIN = (UMAX - UMIN)
      STEP = UIN / (LINENUM + 1)
      N = 0

'     ... start simulate by start clock ...
      tmeClock.Enabled = True

   Else
'  ... when want to stop, activate other bottons ...
      CmdSimul.Font.Bold = False
      CmdSimul.Caption = "Start Simulate"
      CmdSetPara.Enabled = True
      CmdClear.Enabled = True
      CmdMesh.Enabled = True

'     ... stop simulate by stop the clock ...
      tmeClock.Enabled = False
   End If

End Sub


Private Sub tmeClock_Timer()

'  ... Clear picture and draw mesh first ...
   PicGraphic.Cls
   Call DrawMesh(150, 150, 150)

'  ... draw constant u lines by call subroutine "DrawContU" ...
   Call DrawContU

'  ... draw system bounbaries ...
   Call DrawBoundary

'  ... read new data for show next time step ...
   With grdInvisTable
   If N < .Cols - 1 Then N = N + 1 Else N = 1
      .Col = N
      For I = 1 To NNODES
         .Row = I
         U(I) = Val(.Text)
      Next I
   End With

End Sub
```

C5 Source Code of Form "frmSetPara"

```
Private Sub cmdDefault_Click()
   txtNumber.Text = "20"
End Sub
```

```
Private Sub cmdOK_Click()
   Call frmResult.SetPara(Int(Val(txtNumber.Text)))
   Unload Me
End Sub
```

C6 Source Code of Module "Calculation"

```
'*************************************************************************
'**                                                                     **
'**     This module contains all calculation steps.                     **
'**                                                                     **
'*************************************************************************
Option Explicit

Private I As Integer
Private J As Integer
Public NAME_TITLE As String
Public sFile As String


Sub MainProg(ALPHA, BCTYPE, BETA, BNODE, COORDS, DT, ELEMS, EMATL, _
             EPS, ES, ETYPE, F, FVAR, G, GVAR, K, KVAR, NBSEGS, _
             NELEMS, ND, NNODES, NODE, NTOTEL, PICSCALEX1, PICSCALEX2, _
             PICSCALEY1, PICSCALEY2, PRNTFR, SEGTYP, SMATL, SSTATE, _
             S1VAR, S3VAR, TMAX, U, U1, X, Y)
'   ... Set local variable ...
Dim A(5000) As Double
Dim B(5000) As Double
Dim IER As Integer
Dim NBAND As Integer
Dim T As Single
Dim TCOUNT As Integer
'   ... Set dimension for local dinamics variable ...
ReDim AG(3, 3, NELEMS) As Double
ReDim AK(3, 3, NELEMS) As Double
ReDim DELTA(NELEMS) As Double
ReDim DIAG(NNODES) As Integer
ReDim L(NBSEGS) As Double
ReDim NTOTBS(NBSEGS) As Integer
ReDim PHI(NNODES) As Double
ReDim RBAR(NELEMS) As Double
ReDim UNEW(NNODES) As Double

'   ... Prepare form "frmResult" for show data and result ...
   With frmResult
      .Show
      Call .PrepareData(COORDS, NBSEGS, NELEMS, NNODES, SSTATE, NODE, _
                        BNODE, X, Y, PICSCALEX1, PICSCALEX2, _
                        PICSCALEY1, PICSCALEY2)
   End With

'   ... Determine semibandwidth NBAND & vector DIAG ...
   Call Matrix(DIAG, NBAND, NELEMS, NNODES, NODE, NTOTEL)
```

```
'   ... Examine element geometry, find areas or volumes
'       DELTA(E), boundary-segment lengths or areas L(S),
'       and constant parts of conductivity and genera-
'       tion/capacity submatrices ...
    Call Geom(AG, AK, BNODE, COORDS, DELTA, ELEMS, ETYPE, L, NBSEGS, _
              NELEMS, NODE, RBAR, X, Y)

'   ... Is problem steady state or transient? ...
    If (SSTATE) Then
'   ... Assemble matrices A & B and the vector PHI ...
       Call Asembl(A, AG, AK, ALPHA, B, BCTYPE, BETA, BNODE, COORDS, _
                   DELTA, DIAG, DT, ELEMS, EMATL, ETYPE, F, G, K, L, _
                   NBSEGS, NELEMS, NNODES, NODE, NTOTBS, NTOTEL, PHI, _
                   RBAR, SEGTYP, SSTATE, X, Y)
'   ... Compute solution of simultaneous equations ...
       Call NewVal(A, B, DIAG, EPS, IER, NBAND, ND, NNODES, PHI, _
                   SSTATE, U, UNEW, U1)
       If (IER = -1) Then Exit Sub
       Call PrtSol(2, NNODES, T, 1, UNEW)
'       Call Fluxes(ALPHA, BCTYPE, BETA, BNODE, DELTA, ES, F, G, K, _
'                   NBSEGS, NELEMS, NODE, NTOTBS, SEGTYP, SMATL, _
'                   UNEW, X, Y)
       Exit Sub
    Else
'      ... Initialize time, print starting values ...
       T = 0
       TCOUNT = 0
       Call PrtSol(1, NNODES, T, 1, U)
'      ... Increment time ...
10     T = T + DT
       TCOUNT = TCOUNT + 1
'      ... Compute and print new values ...
       Call Asembl(A, AG, AK, ALPHA, B, BCTYPE, BETA, BNODE, COORDS, _
                   DELTA, DIAG, DT, ELEMS, EMATL, ETYPE, F, G, K, L, _
                   NBSEGS, NELEMS, NNODES, NODE, NTOTBS, NTOTEL, PHI, _
                   RBAR, SEGTYP, SSTATE, X, Y)
       Call NewVal(A, B, DIAG, EPS, IER, NBAND, ND, NNODES, PHI, _
                   SSTATE, U, UNEW, U1)
       If (IER = -1) Then Exit Sub
       If (Int(Int(TCOUNT / PRNTFR) * PRNTFR) = TCOUNT) Then _
          Call PrtSol(1, NNODES, T, TCOUNT, UNEW)

'         ... New solutions become old ones ...
          For I = 1 To NNODES
             U(I) = UNEW(I)
          Next I

'      ... Time to quit ? ...
        If (T < TMAX - DT / 2#) Then GoTo 10
    End If

End Sub


Private Sub Matrix(DIAG, NBAND, NELEMS, NNODES, NODE, NTOTEL)
Dim E As Integer
```

```
Dim IMAX As Integer
Dim NTOT As Integer
Dim LAST As Integer
'  This subroutine determines the following:
'     1. Semibandwidth NBAND of the matrices A & B
'     2. The vector DIAG

'  ... Determine semibandwidth NBAND (excluding diagonal) ...
   NBAND = 0
'  ... Examine all elements for maximum difference in nodal numbers ...
   For E = 1 To NELEMS
      NTOT = NTOTEL(E)
      IMAX = NTOT - 1
      For I = 1 To IMAX
         For J = I + 1 To NTOT
            If Abs(NODE(I, E) - NODE(J, E)) > NBAND Then NBAND = _
                                       Abs(NODE(I, E) - NODE(J, E))
         Next J
      Next I
   Next E

'  ... Determine the vector DIAG ...
   DIAG(1) = 1
   LAST = 1 + NBAND
   For I = 2 To NNODES
      DIAG(I) = LAST + I - Max(1, I - NBAND) + 1
      LAST = DIAG(I) + Min(NNODES, I + NBAND) - I
   Next I
End Sub


Private Sub Geom(AG, AK, BNODE, COORDS, DELTA, ELEMS, ETYPE, L, _
                 NBSEGS, NELEMS, NODE, RBAR, X, Y)

Dim A1, A2, A3 As Double
Dim AGLOCL(3, 3) As Double
Dim AK12, AK13, AK23 As Double
Dim B1, B2, B3 As Double
Dim C1, C2, C3 As Double
Dim E As Integer
Dim I As Integer
Dim J As Integer
Dim S As Integer
Dim N1, N2, N3 As Integer
AGLOCL(1, 1) = 0.166666666666667
AGLOCL(1, 2) = 0.083333333333333
AGLOCL(1, 3) = 0.083333333333333
AGLOCL(2, 1) = 0.083333333333333
AGLOCL(2, 2) = 0.166666666666667
AGLOCL(2, 3) = 0.083333333333333
AGLOCL(3, 1) = 0.083333333333333
AGLOCL(3, 2) = 0.083333333333333
AGLOCL(3, 3) = 0.166666666666667

   If (ELEMS = 1) Then
'     ... Examine each element in turn ...
```

```
      For E = 1 To NELEMS
         N1 = NODE(1, E)
         N2 = NODE(2, E)
         N3 = NODE(3, E)
         A1 = X(N2) * Y(N3) - X(N3) * Y(N2)
         A2 = X(N3) * Y(N1) - X(N1) * Y(N3)
         A3 = X(N1) * Y(N2) - X(N2) * Y(N1)
         B1 = Y(N2) - Y(N3)
         B2 = Y(N3) - Y(N1)
         B3 = Y(N1) - Y(N2)
         C1 = X(N3) - X(N2)
         C2 = X(N1) - X(N3)
         C3 = X(N2) - X(N1)
'        ... Compute element areas and mean radii ...
         DELTA(E) = 0.5 * (A1 + A2 + A3)
         If (COORDS = 2) Then RBAR(E) = (X(N1) + X(N2) + X(N3)) / 3#

'        ... Form constant parts of conductivity
'            and generation/capacity submatrices ...
         AK(1, 1, E) = B1 ^ 2 + C1 ^ 2
         AK(2, 2, E) = B2 ^ 2 + C2 ^ 2
         AK(3, 3, E) = B3 ^ 2 + C3 ^ 2
            AK12 = B1 * B2 + C1 * C2
         AK(1, 2, E) = AK12
         AK(2, 1, E) = AK12
            AK13 = B1 * B3 + C1 * C3
         AK(1, 3, E) = AK13
         AK(3, 1, E) = AK13
            AK23 = B2 * B3 + C2 * C3
         AK(2, 3, E) = AK23
         AK(3, 2, E) = AK23
         For J = 1 To 3
            For I = 1 To 3
               If (COORDS = 2) Then AK(I, J, E) = AK(I, J, E) * RBAR(E)
               AK(I, J, E) = AK(I, J, E) / (4# * DELTA(E))
               AG(I, J, E) = AGLOCL(I, J) * DELTA(E)
            Next I
         Next J
      Next E

'     ... Find lengths L(S) of boundary segments...
      For S = 1 To NBSEGS
         N1 = BNODE(1, S)
         N2 = BNODE(2, S)
         L(S) = Sqr(((X(N1) - X(N2)) ^ 2) + ((Y(N1) - Y(N2)) ^ 2))
      Next S
   Else
   End If
End Sub


Private Sub PrtSol(SWITCH, NNODES, T, TCOUNT, U)

'    This subroutine prints the prevailing solutions
'    (steady state or transient) at all nodes.   For
'    transient problems, the values of time is also printed
```

```
'    together with the prevailing solutions at all nodes.
'    Values of SWITCH have the following meanings:
'       1. Initial values and transient solution
'       2. Steady-state solution

'   ... All steps to show result are in subroutine "ShowSol" in
'       form "frmResult" ...
   Call frmResult.ShowSol(SWITCH, T, TCOUNT, U)

End Sub


Private Sub Asembl(A, AG, AK, ALPHA, B, BCTYPE, BETA, BNODE, COORDS, _
                   DELTA, DIAG, DT, ELEMS, EMATL, ETYPE, F, G, K, L, _
                   NBSEGS, NELEMS, NNODES, NODE, NTOTBS, NTOTEL, PHI, _
                   RBAR, SEGTYP, SSTATE, X, Y)
'  This subroutine constructs the coefficient matrices
'  A & B, and the right-hand side vector PHI
'  ... Declarations for local variables ...
Dim C1, C2 As Double
Dim DI As Integer
Dim E As Integer
Dim GEDT As Double
Dim IL, IJ, JL As Integer
Dim IMAX As Integer
Dim S As Integer
Dim KE As Double
Dim M1(2, 2) As Double
ReDim rbars(NBSEGS) As Double
M1(1, 1) = 2#
M1(1, 2) = 1#
M1(2, 1) = 1#
M1(2, 2) = 2#

'   ... For steady-state, divisor DT in GEDT is one ...
   If (SSTATE) Then DT = 1#

'   ... Zero out matrices A & B and vector PHI ...
   IMAX = DIAG(NNODES)
   For I = 1 To IMAX
      A(I) = 0#
      B(I) = 0#
   Next I
   For I = 1 To NNODES
      PHI(I) = 0#
   Next I

'   ... Modify A and PHI for S3 boundary segments ...
   If (ELEMS = 1) Then
      For S = 1 To NBSEGS
         If (BCTYPE(S) = 3) Then
            C1 = BETA(S) * L(S) / 2#
            C2 = ALPHA(S) * L(S) / 6#
            If (COORDS = 2) Then rbars(S) = (X(BNODE(1, S)) + _
                                             X(BNODE(2, S))) / 2#
            For IL = 1 To 2
```

```
            I = BNODE(IL, S)
            If (COORDS = 1) Then PHI(I) = PHI(I) + C1
            If (COORDS = 2) Then PHI(I) = PHI(I) + C1 _
                                  * (X(I) + 2# * rbars(S)) / 3#
            DI = DIAG(I)
            For JL = 1 To 2
               J = BNODE(JL, S)
               IJ = DI + J - I
               If (COORDS = 1) Then A(IJ) = A(IJ) + C2 * M1(IL, JL)
               If (COORDS = 2) Then A(IJ) = A(IJ) + C2 * M1(IL, JL) _
                                       * (X(I) + rbars(S)) / 2#
            Next JL
         Next IL
      End If
   Next S

'     ... Conductivity & generation/capacity terms for AK & AG ...
   For E = 1 To NELEMS
      KE = K(E)
      GEDT = G(E) / DT
'      NODTOT = NTOTEL(E)
      For IL = 1 To NTOTEL(E)
         I = NODE(IL, E)
         DI = DIAG(I)
         For JL = 1 To NTOTEL(E)
            J = NODE(JL, E)
            IJ = DI + J - I
            A(IJ) = A(IJ) + AK(IL, JL, E) * KE
            If (COORDS = 2) Then AG(IL, JL, E) = AG(IL, JL, E) _
                                  * (X(I) + X(J) + 3# * RBAR(E)) / 5#
            B(IJ) = B(IJ) + AG(IL, JL, E) * GEDT
         Next JL
      Next IL
   Next E

'     ... GENERATION TERMS FOR VECTOR PHI ...
   For E = 1 To NELEMS
      For IL = 1 To 3
         I = NODE(IL, E)
         If (COORDS = 1) Then PHI(I) = PHI(I) + F(E) * DELTA(E) / 3#
         If (COORDS = 2) Then PHI(I) = PHI(I) + F(E) * DELTA(E) / 3# _
                                  * (X(I) + 3# * RBAR(E)) / 4#
      Next IL
   Next E
Else
   MsgBox "specific element type is not yet implemented in ASEMBL", _
          vbExclamation
End If
End Sub


Private Sub NewVal(A, B, DIAG, EPS, IER, NBAND, ND, NNODES, PHI, _
                   SSTATE, U, UNEW, U1)
'  ... This subroutine computes the new solutions at the
'      end of a time step, using the fully implicit method ...
Dim DI As Integer
```

```
Dim IJ As Integer
Dim JLOW, JHIGH As Integer

'  ... Form LHS coefficient matrix and RHS vector, adjusted for known
'      Dirichlet values ...
   For I = 1 To NNODES
      JLOW = Max(1, I - NBAND)
      JHIGH = Min(NNODES, I + NBAND)
      DI = DIAG(I)

      If Not ND(I) Then
'     ... Case of a non-Dirichlet node ...
         For J = JLOW To JHIGH
            IJ = DI + J - I
            A(IJ) = A(IJ) + B(IJ)
'           ... Term B(IJ)*U(J) is for transient case only ...
            If Not SSTATE Then PHI(I) = PHI(I) + B(IJ) * U(J)
         Next J
      Else

'     ... Case of a Dirichlet node ...
         For J = JLOW To JHIGH
            IJ = DI + J - I
            A(IJ) = 0#
         Next J
         A(DI) = 1#
         PHI(I) = U1(I)
      End If
   Next I

'  ... Solve banded system of simultaneous equations using Gaussian
'      elimination  ...
   Call Gelb(PHI, A, NNODES, 1, NBAND, EPS, IER)


'  ... Simultaneous equations not solvable ...
   If (IER = -1) Then
      MsgBox "Simultaneous equations not solved.", vbCritical
      Exit Sub
'  ... Possible loss of accuracy in GELB ...
   ElseIf (IER <> 0) Then
      MsgBox "Warning: possible loss of significance at elimination step.
Problem is not solved.", vbCritical
   End If

   For I = 1 To NNODES
      UNEW(I) = PHI(I)
   Next I
End Sub


Private Sub Fluxes(ALPHA, BCTYPE, BETA, BNODE, DELTA, ES, F, G, K, _
                   NBSEGS, NELEMS, NODE, NTOTBS, SEGTYP, SMATL, U, _
                   X, Y)
'  ... Subprogram for determining the fluxes through the boundary
'      segments (positive into the medium) ...
```

```
'   ... Declarations for local variables ...
Dim B1, B2, B3 As Double
Dim C1, C2, C3 As Double
Dim DX, DY As Double
Dim E As Integer
Dim ERROR As Double
Dim GEN As Double
Dim Q As Double
Dim S As Integer
Dim L As Double
Dim N1, N2, N3 As Integer
Dim SMALLQ As Double
Dim SUMGEN, SUMNEG, SUMPOS As Double
Dim TYPE_(3) As String
Dim UBAR As Double
TYPE_(1) = "Dirichlet"
TYPE_(2) = "Insulated"
TYPE_(3) = "Mixed"

   SUMNEG = 0#
   SUMPOS = 0#

'   ... Examine each boundary segment in turn ...
   For S = 1 To NBSEGS
'      ... Segment belongs to linear triangle ? ...
      If (SEGTYP(S) = 1) Then
         I = BNODE(1, S)
         J = BNODE(2, S)
         DX = X(I) - X(J)
         DY = Y(J) - Y(I)
         L = Sqr(DX ^ 2 + DY ^ 2)
'         ... Check for Dirichlet-type boundary condition ...
         If (BCTYPE(S) = 1) Then
            E = ES(S)
            N1 = NODE(1, E)
            N2 = NODE(2, E)
            N3 = NODE(3, E)
            B1 = Y(N2) - Y(N3)
            B2 = Y(N3) - Y(N1)
            B3 = Y(N1) - Y(N2)
            C1 = X(N3) - X(N2)
            C2 = X(N1) - X(N3)
            C3 = X(N2) - X(N1)
            Q = (U(N1) * (B1 * DY + C1 * DX) + U(N2) * (B2 * DY + C2 _
                * DX) + U(N3) * (B3 * DY + C3 * DX)) * K(E) / (2# * _
                DELTA(E))
'         ... Check for insulated-type boundary condition ...
         ElseIf (BCTYPE(S) = 2) Then
            Q = 0#
'         ... Otherwise, boundary condition is mixed ...
         Else
            Q = (BETA(S) - ALPHA(S) * (U(I) + U(J)) / 2#) * L
         End If
         SMALLQ = Q / L
         If (Q < 0#) Then
```

```
            SUMNEG = SUMNEG + Q
         Else
            SUMPOS = SUMPOS + Q
         End If
'     ... Segment does not belong to a linear triangle ...
      Else
      End If
   Next S
'  ... Total of element generation rates ...
   SUMGEN = 0#
   For E = 1 To NELEMS
      UBAR = (U(NODE(1, E)) + U(NODE(2, E)) + U(NODE(3, E))) / 3#
      GEN = (F(E) - G(E) * UBAR) * DELTA(E)
      SUMGEN = SUMGEN + GEN
   Next E
   ERROR = SUMGEN + SUMPOS + SUMNEG
End Sub


Private Sub Gelb(R, A, M, N, MUD, EPS, IER)
Dim MC As Integer
Dim MU As Integer
Dim ML As Integer
Dim MR As Integer
Dim MZ As Integer
Dim MA As Integer
Dim NM As Integer
Dim IC As Integer
Dim ID As Integer
Dim I As Integer
Dim II As Integer
Dim IDST As Integer
Dim ILR As Integer
Dim JJ As Integer
Dim J As Integer
Dim K As Integer
Dim KST As Integer
Dim PIV As Double
Dim TB As Double
Dim TOL As Double

   On Error GoTo ERROR1
'  ... Test on wrong input parameters ...
   If (MUD < 0) Then GoTo ERROR1
   MC = 1 + MUD * 2
   If (MC + 1 - 2 * M > 0) Then GoTo ERROR1

'  ... Prepare integer parameters ...
   If (MC - M > 0) Then MC = M
   MU = MC - MUD - 1
   ML = MC - MUD - 1
   MR = M - ML
   MZ = (MU * (MU + 1)) / 2
   MA = M * MC - (ML * (ML + 1)) / 2
   NM = N * M
```

```
' ... Move elements backward and search for absolutely greatest
'     element (not necessary in case of a matrix without lower
'     codiagonals) ...
IER = 0
PIV = 0#
If (MUD > 0) Then
   JJ = MA
   J = MA - MZ
   KST = J
   For K = 1 To KST
      TB = A(J)
      A(JJ) = TB
      TB = Abs(TB)
      If (TB - PIV > 0) Then PIV = TB
      J = J - 1
      JJ = JJ - 1
   Next K

' ... Insert zeros in first MU rows (not necessary in case MZ=0) ...
   If (MZ > 0) Then
      JJ = 1
      J = 1 + MZ
      IC = 1 + MUD
      For I = 1 To MU
         For K = 1 To MC
            A(JJ) = 0#
            If (K - IC <= 0) Then
               A(JJ) = A(J)
               J = J + 1
            End If
            JJ = JJ + 1
         Next K
         IC = IC + 1
      Next I
   End If
End If

' ... Generate test value for singularity ...
TOL = EPS * PIV

' ... Start decomposition loop ...
KST = 1
IDST = MC
IC = MC - 1
For K = 1 To M
   If (K - MR - 1 > 0) Then IDST = IDST - 1
   ID = IDST
   ILR = K + MUD
   If (ILR - M > 0) Then ILR = M
   II = KST

'  ... Pivot search in first column (row indexes from I=K up to
'      I=ILR) ...
   PIV = 0#
   For I = K To ILR
      TB = Abs(A(II))
```

```
   If (TB - PIV > 0) Then
      PIV = TB
      J = I
      JJ = II
   End If
   If (I - MR > 0) Then ID = ID - 1
   II = II + ID
Next I

... Test on singularity ...
If (PIV <= 0) Then GoTo ERROR1
If (IER = 0) Then
   If (PIV - TOL <= 0) Then IER = K - 1
End If
PIV = 1# / A(JJ)

... Pivot row reduction and row interchange in right hand
    side r ...
ID = J - K
For I = K To NM Step M
   II = I + ID
   TB = PIV * R(II)
   R(II) = R(I)
   R(I) = TB
Next I

... Pivot row reduction and row interchange in cofeeicient
    matrix A ...
II = KST
J = JJ + IC
For I = JJ To J
  TB = PIV * A(I)
  A(I) = A(II)
  A(II) = TB
  II = II + 1
Next I

... element reduction ...
If (K - ILR < 0) Then
   ID = KST
   II = K + 1
   MU = KST + 1
   MZ = KST + IC

   For I = II To ILR
      ... in matrix A ...
      ID = ID + MC
      JJ = I - MR - 1
      If (JJ > 0) Then ID = ID - JJ
      PIV = -A(ID)
      J = ID + 1
      For JJ = MU To MZ
         A(J - 1) = A(J) + PIV * A(JJ)
         J = J + 1
      Next JJ
      A(J - 1) = 0#
```

```
'           ... in matrix R ...
            J = K
            For JJ = I To NM Step M
               R(JJ) = R(JJ) + PIV * R(J)
               J = J + M
            Next JJ
         Next I
      End If
      KST = KST + MC
      If (ILR - MR >= 0) Then IC = IC - 1
      ID = K - MR
      If (ID > 0) Then KST = KST - ID
   Next K
'  ... end decomposition loop ...

'  ... back substitution ...
   If (MC - 1 <= 0) Then Exit Sub
   IC = 2
   KST = MA + ML - MC + 2
   II = M
   For I = 2 To M
      KST = KST - MC
      II = II - 1
      J = II - MR
      If (J > 0) Then KST = KST + J
      For J = II To NM Step M
         TB = R(J)
         MZ = KST + IC - 2
         ID = J
         For JJ = KST To MZ
            ID = ID + 1
            TB = TB - A(JJ) * R(ID)
         Next JJ
         R(J) = TB
      Next J
      If (IC - MC < 0) Then IC = IC + 1
   Next I
   Exit Sub

'  ... error return ...
ERROR1:
   IER = -1
   End Sub


Function Max(A, B)
'  ... subfunction used to find maximum value of A and B ...
   If A > B Then
      Max = A
   Else
      Max = B
   End If
End Function
```

```
Function Min(A, B)
'  ... subfunction used to find minimum value of A and B ...
   If A < B Then
      Min = A
   Else
      Min = B
   End If
End Function
```

C7 Source Code of Module "Complementary"

```
'*******************************************************************
'**                                                               **
'**  This module is used for other complementations of program.   **
'**                                                               **
'*******************************************************************

Public fMainForm As frmMain


Sub Main()
    Set fMainForm = New frmMain
    fMainForm.Show
End Sub
```

CURRICULUM VITAE

Name: Mr. Puchong Thipkhunthod

Date of Birth: January 2, 1979

Nationality: Thai

University Education:

1996 – 2000 Bachelor's Degree of Engineering in Chemical Engineering, Faculty of Engineering, Thammasat University.

Presentation and Publication:

1. Thipkhunthod, P., Siemanond, K, Rangsunvigit, P. and Meeyoo, V., (2001). Experimental and predictions of water and hydrocarbon adsorption on activated alumina prepared via sol-gel technique. Proceedings of the 6th World Congress of Chemical Engineering, Melbourne, Australia.