REFERENCES

Chaudhry, M. H. (1993). Open Channel Flow. New Jersey: Prentice-Hall Inc.

Daugherty, R. L., and Franzini, J. B. (1965). Fluid Mechanics, 6th ed. New York: McGraw-Hill, Inc.

Fox, R. W., and McDonald, A. T. (1994). Introduction to Fluid Mechanic. New York: John Wiley & Son, Inc.

Potter, M. C., and Wiggert, D. C., (1991). Mechanics of Fluids. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Wilkes, J. O. (1999). Fluid Mechanics for Chemical Engineers. New Jersey: Prentice-Hall, Inc.

APPENDIX A
THE PCN INTERFACE

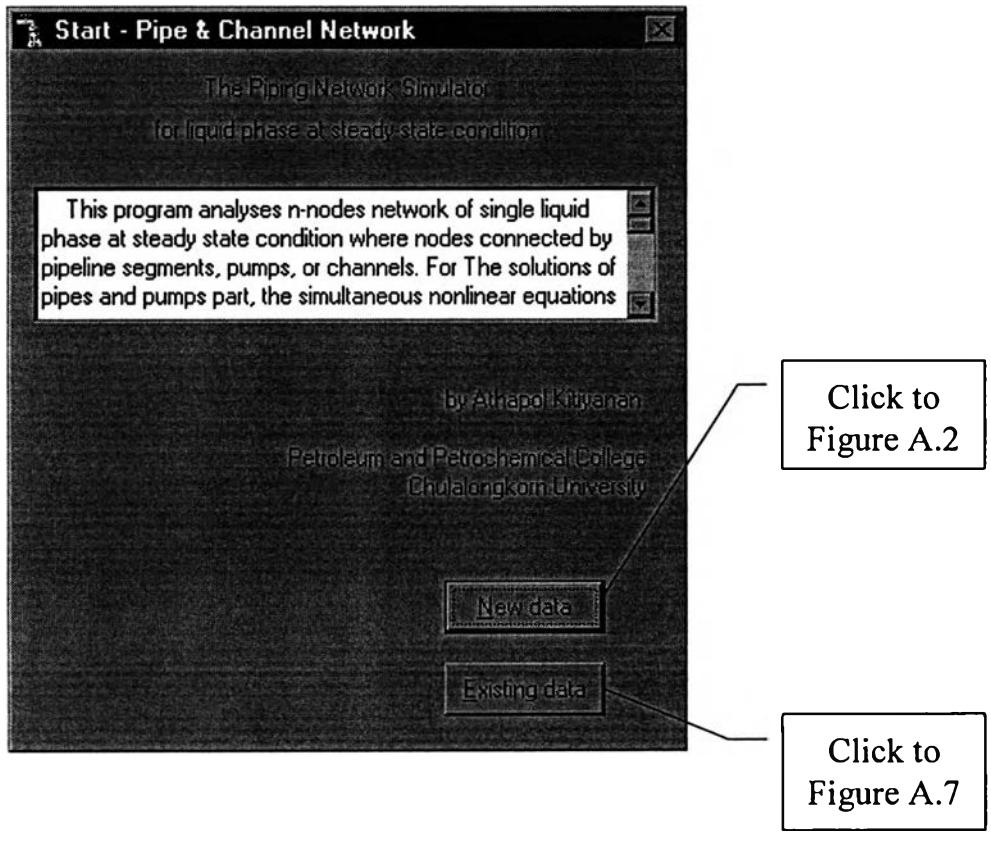

Figure A.1 -Start-

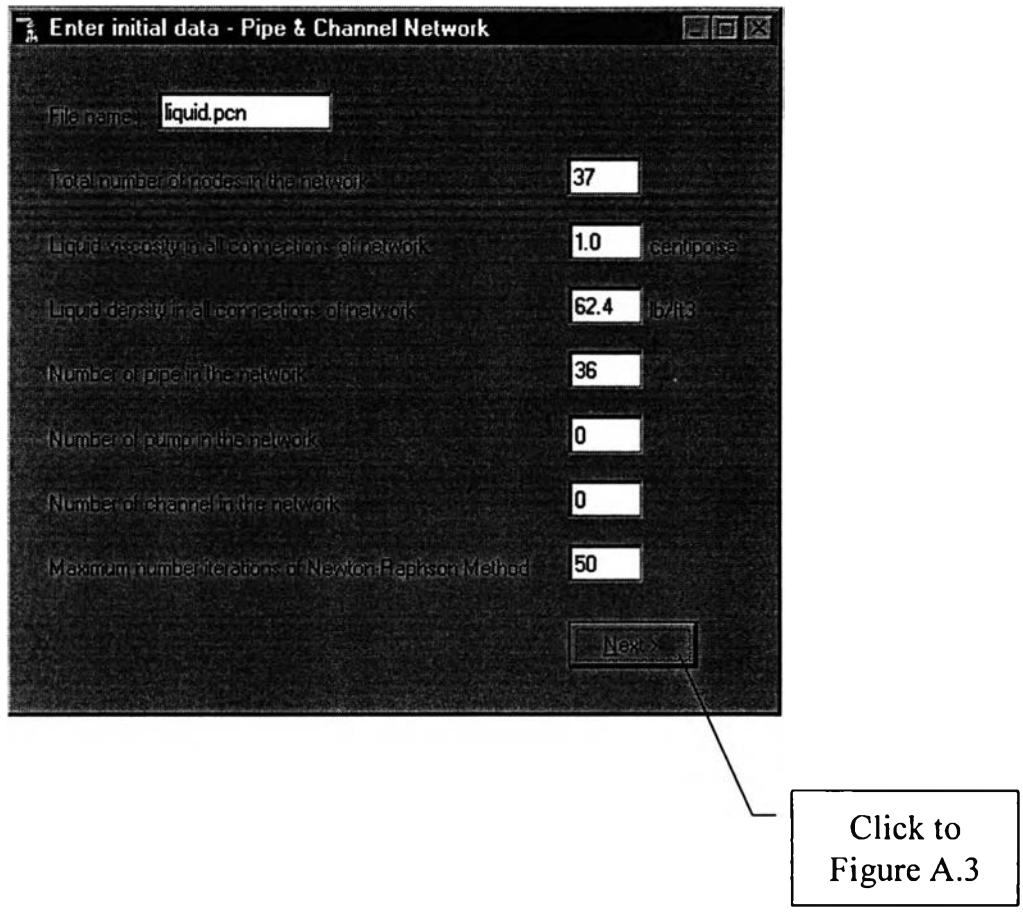

Figure A.2 -Enter initial data-

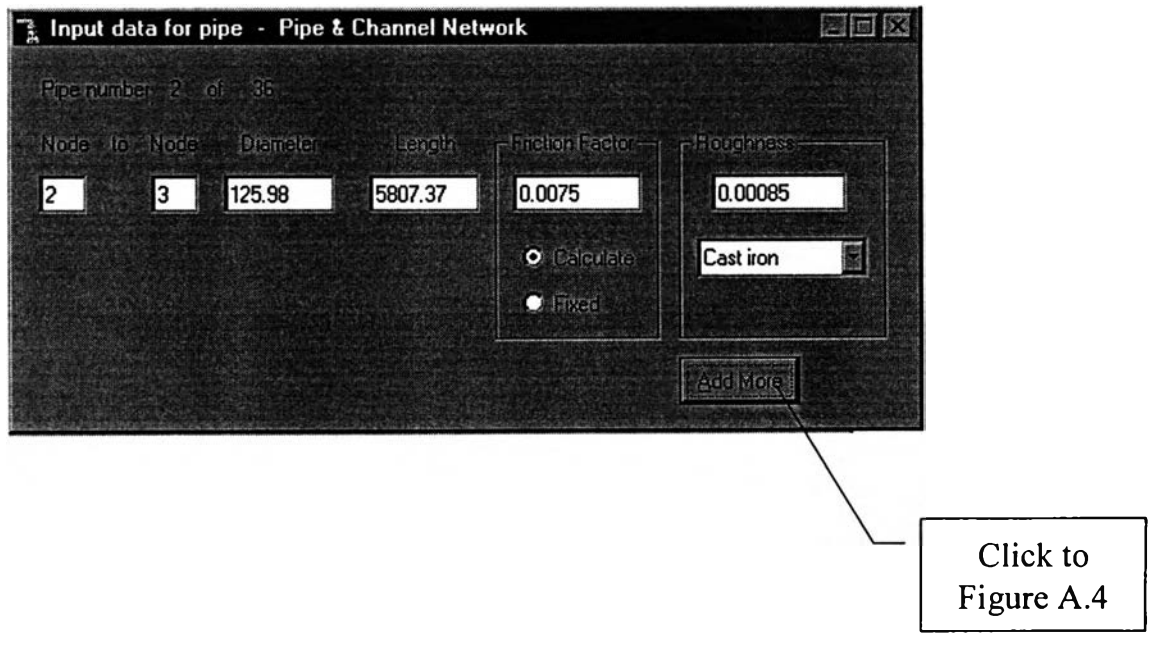

Figure A.3 -Input pipe data-

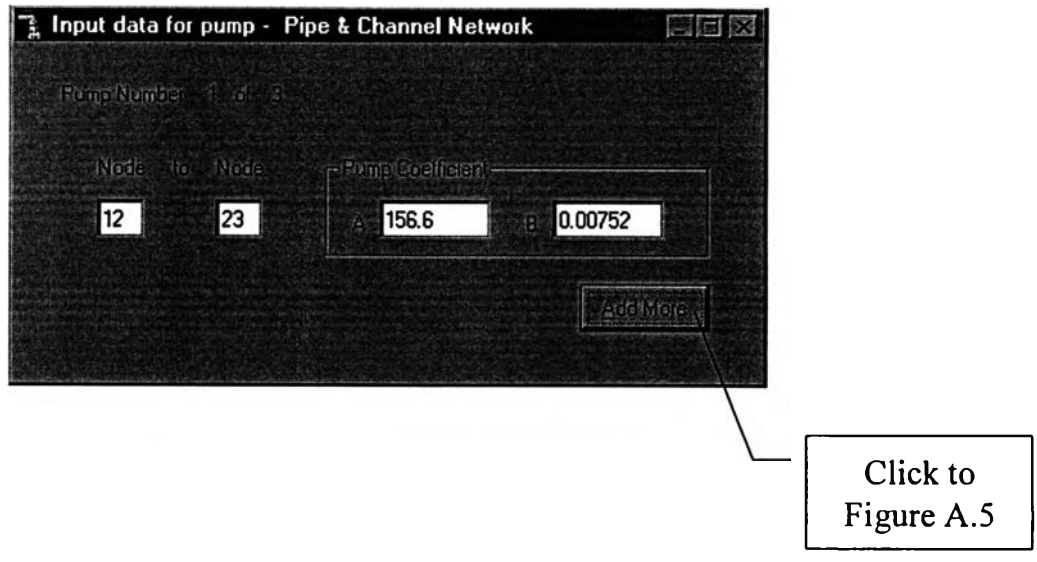

Figure A.4 -Input pump data-

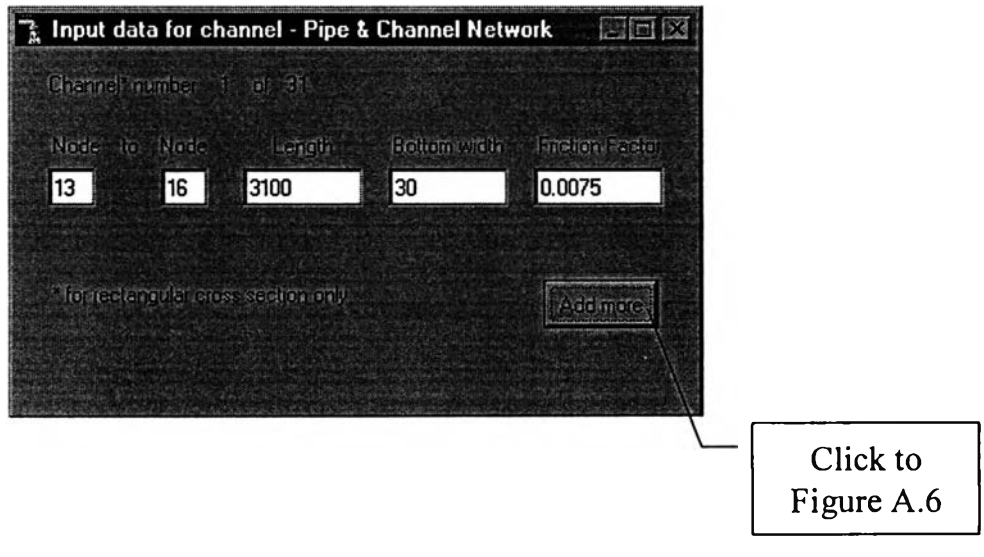

Figure A.5 -Input channel data-

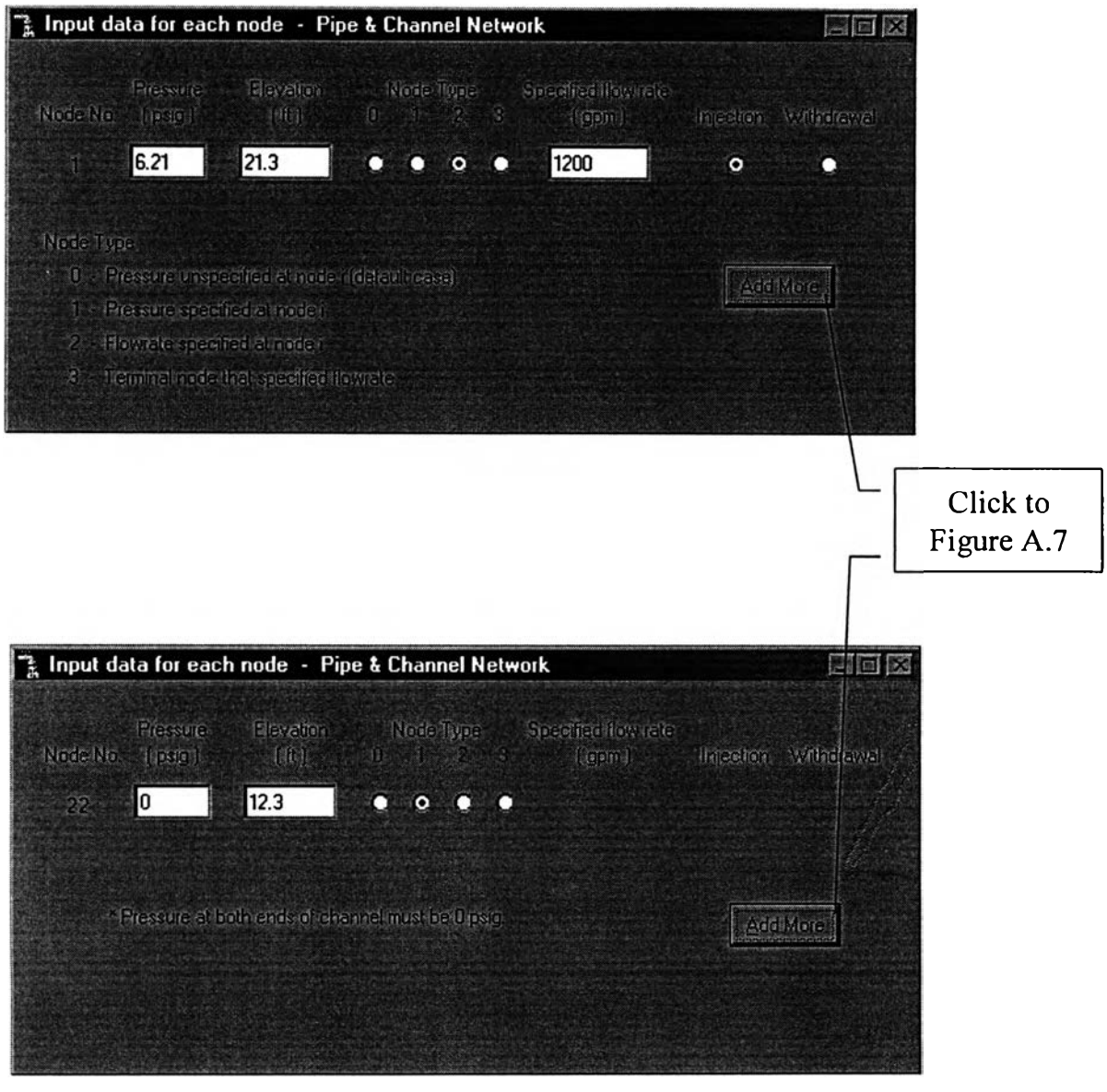

Figure A.6 -Input node data-

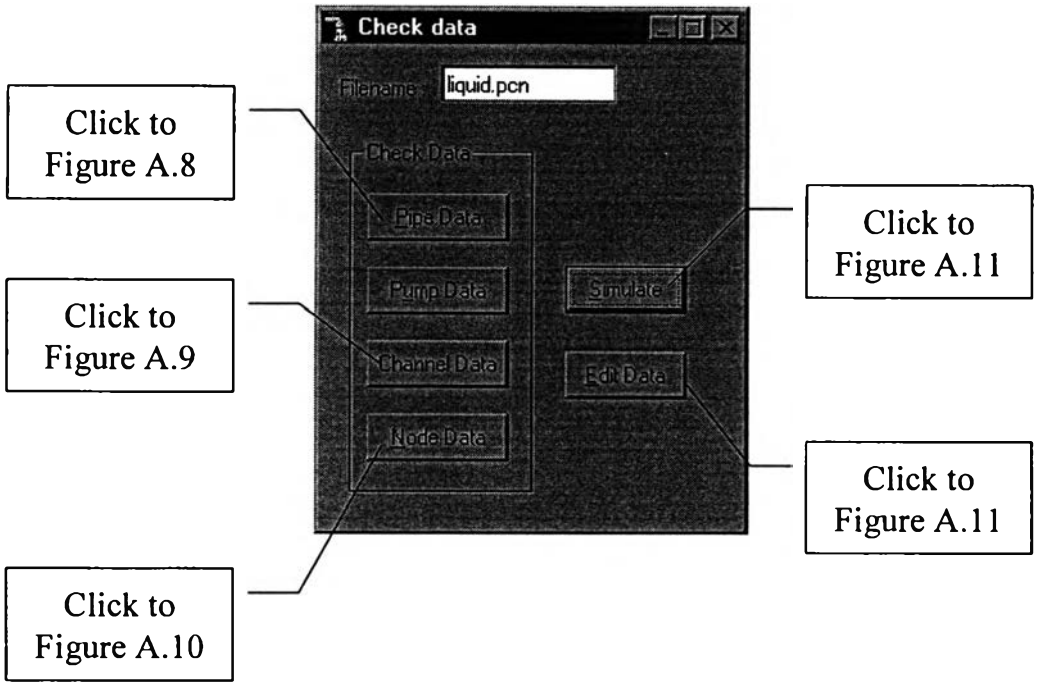

Figure A.7 -Check data-

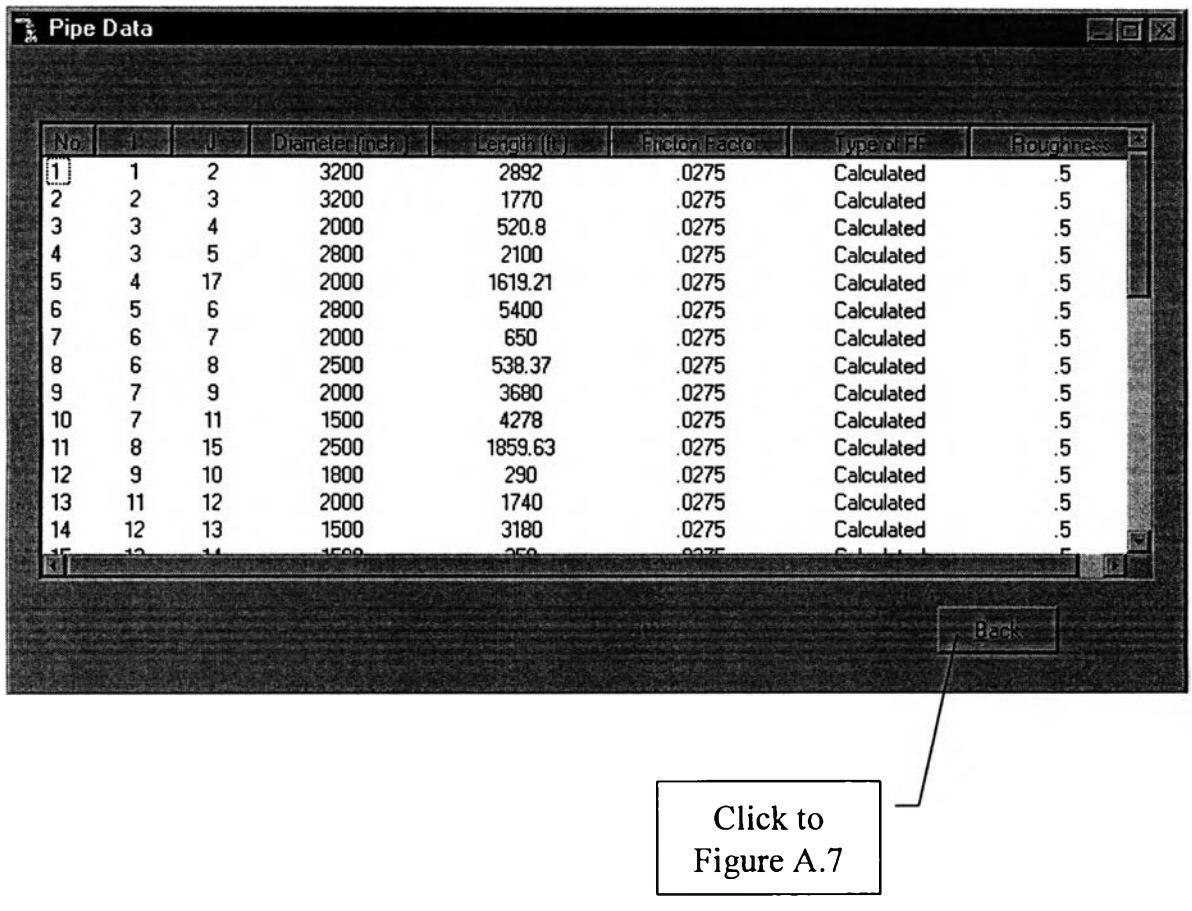

Figure A.8 -Check pipe data-

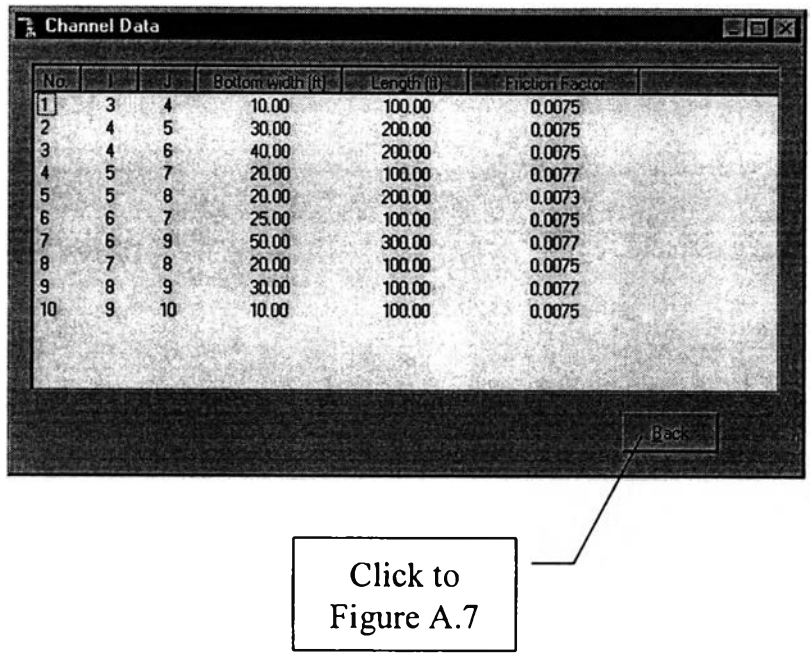

Figure A.9 -Check channel data-

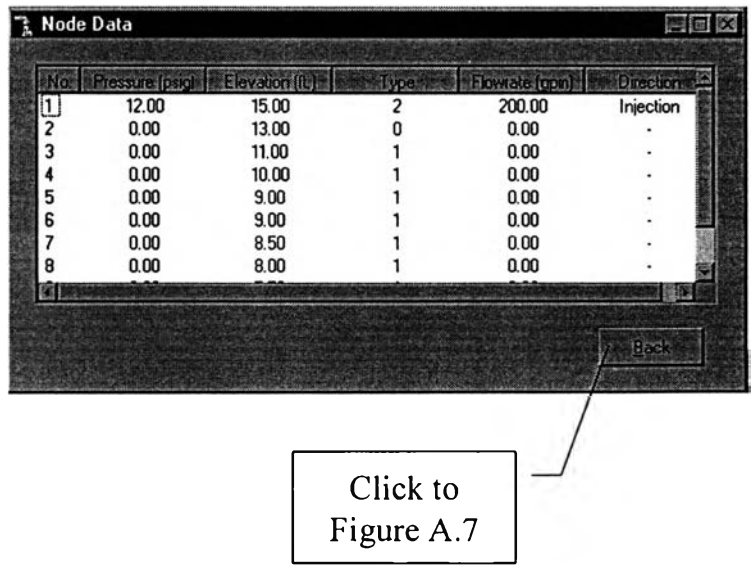

Figure A.10 -Check node data-

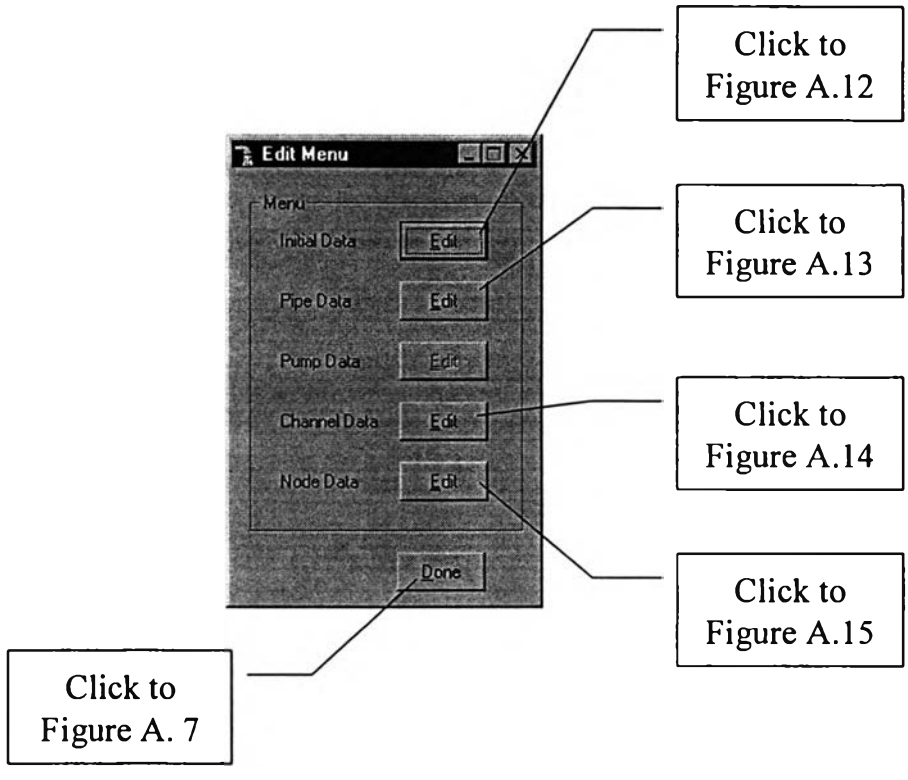

Figure A.11 -Edit data menu-

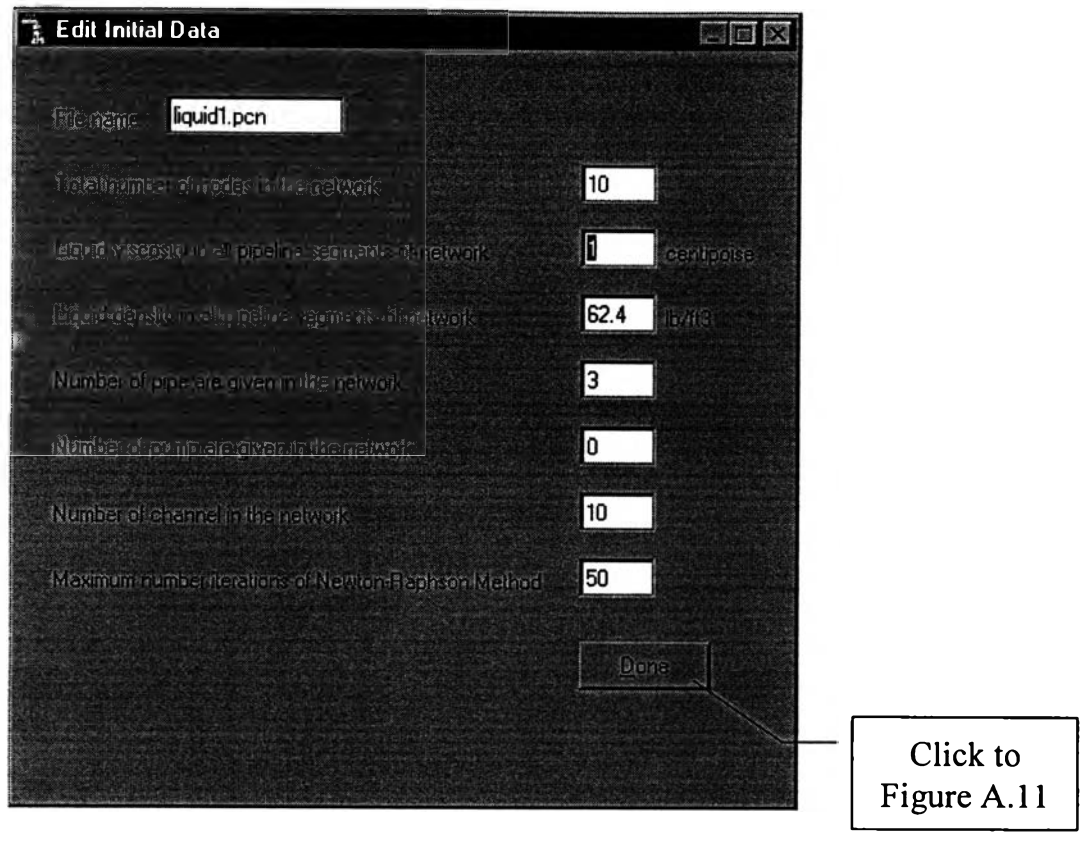

Figure A.12 -Edit initial data-

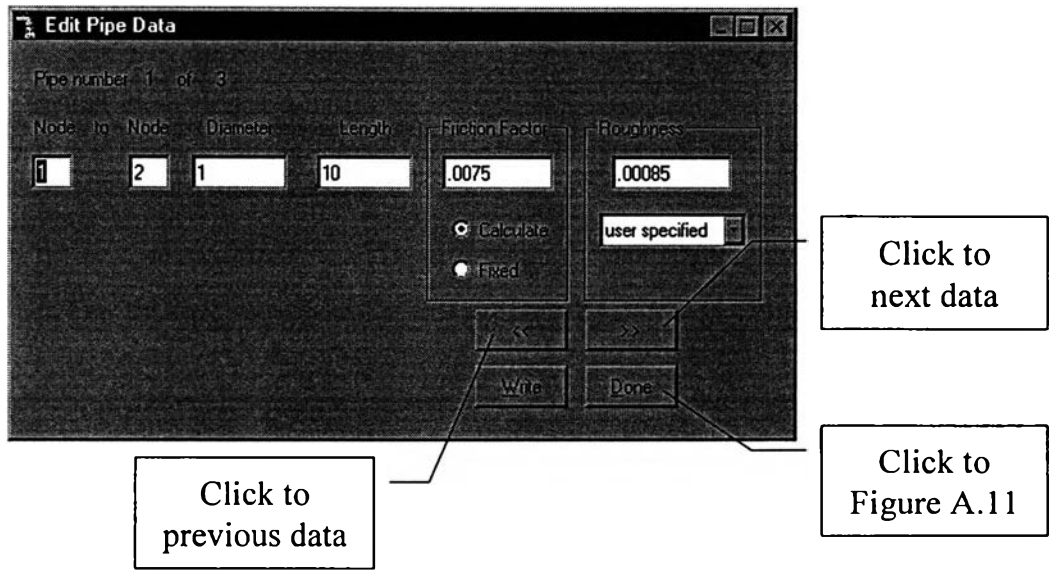

Figure A.13 -Check pipe data-

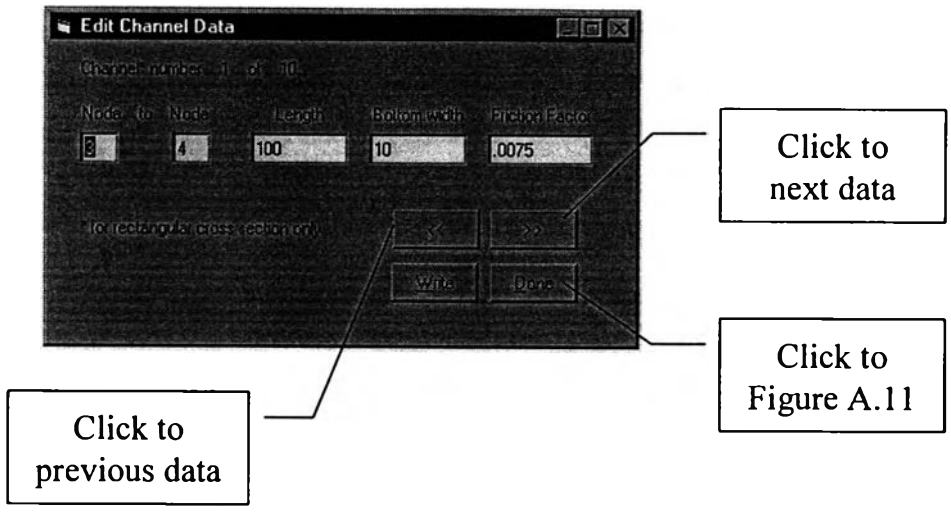

Figure A.14 -Edit channel data-

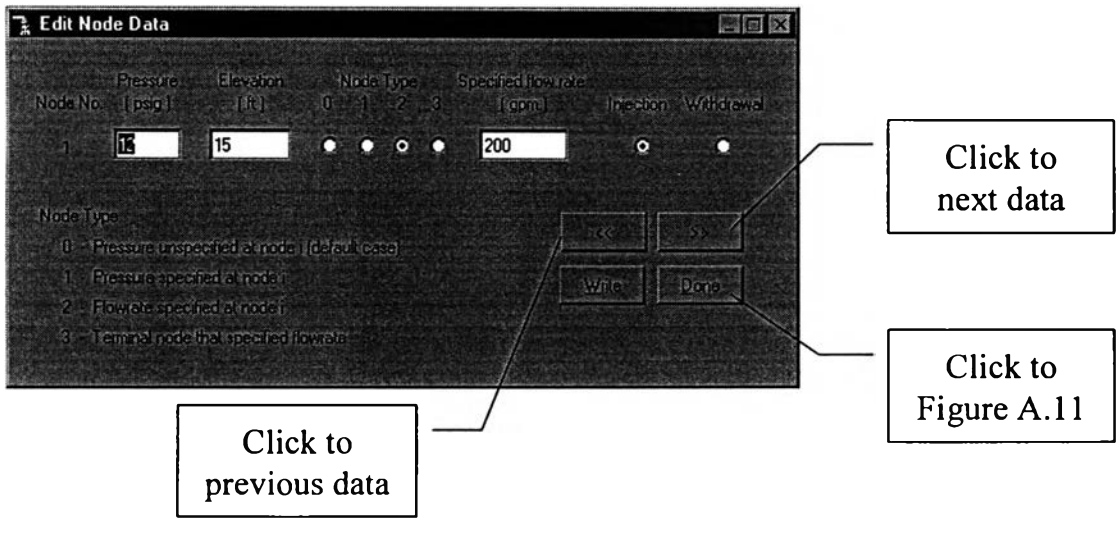

Figure A.15 -Edit node data-

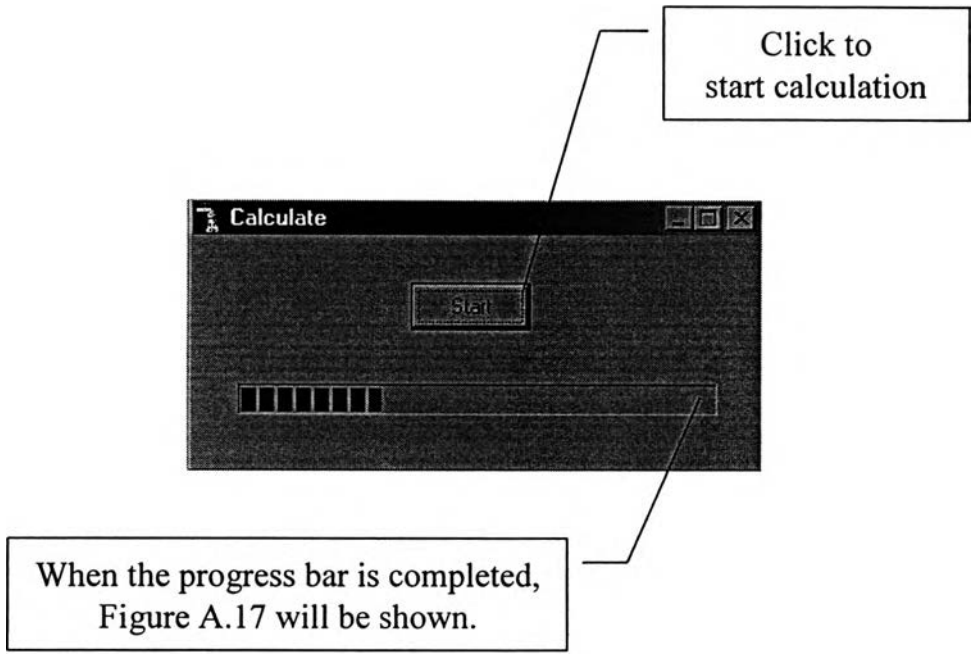

Figure A.16 -Calculate-

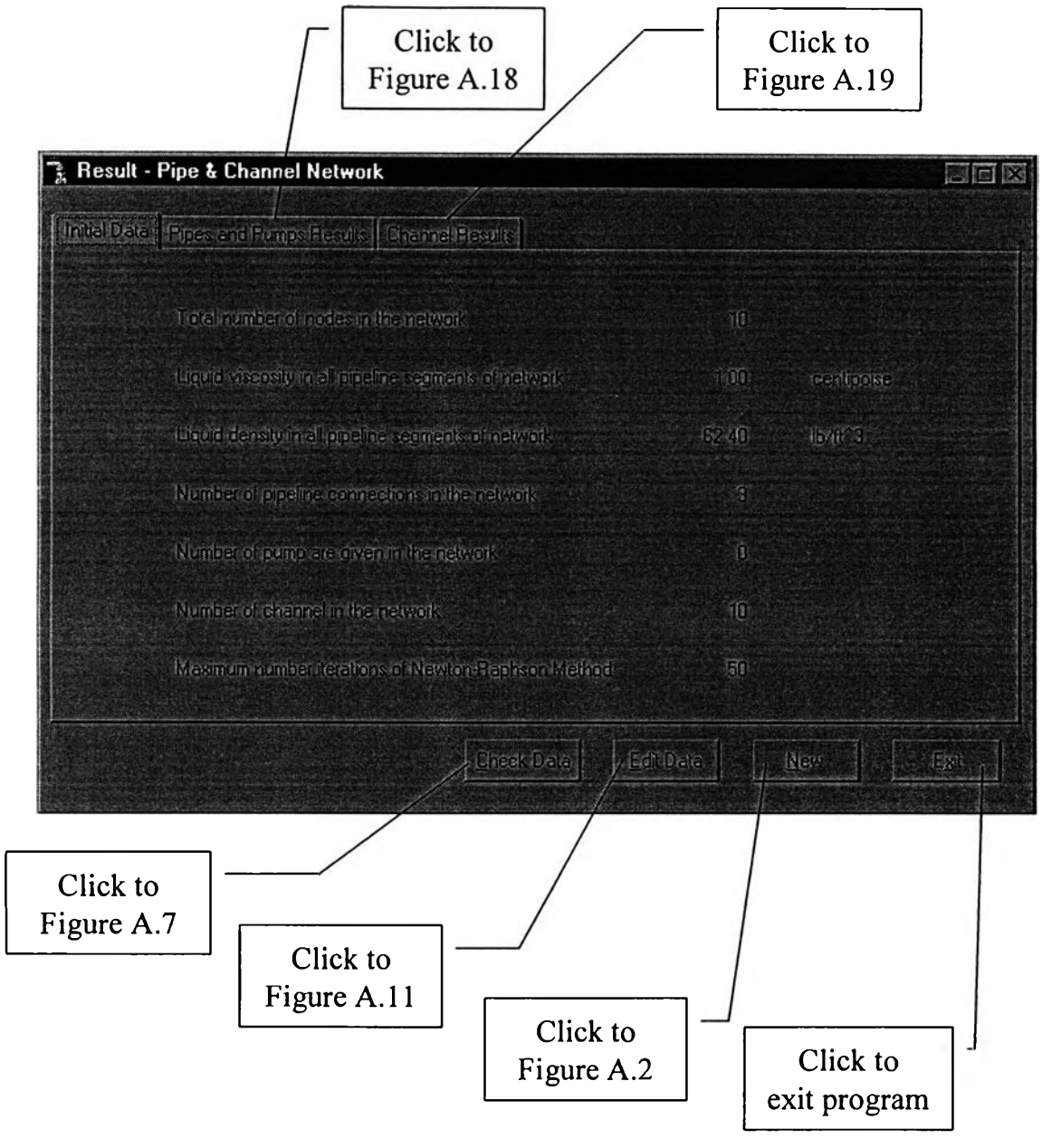

Figure A.17 -View result (Initial data)-

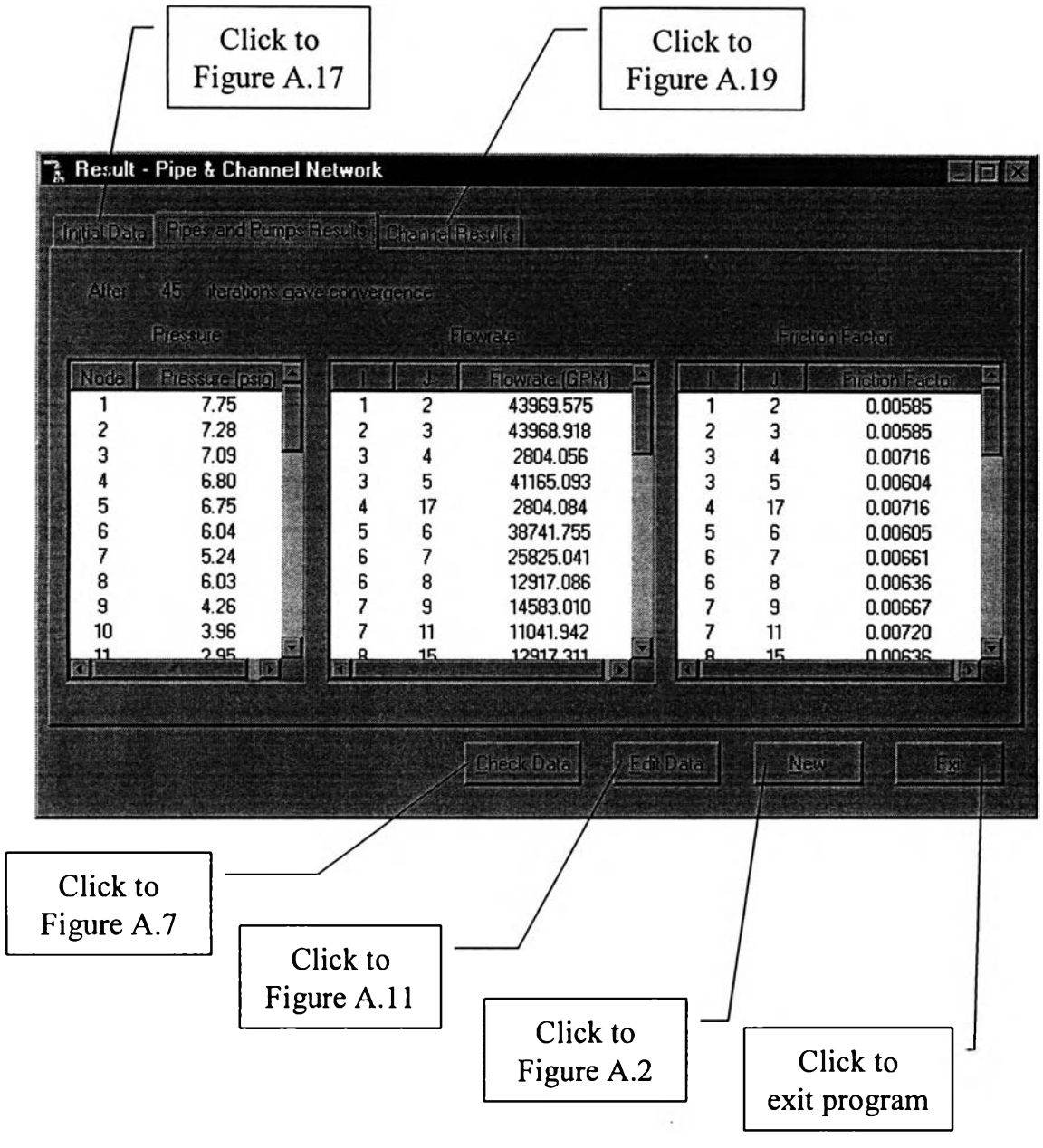

Figure A.18 -View result (pipe and pump results)-

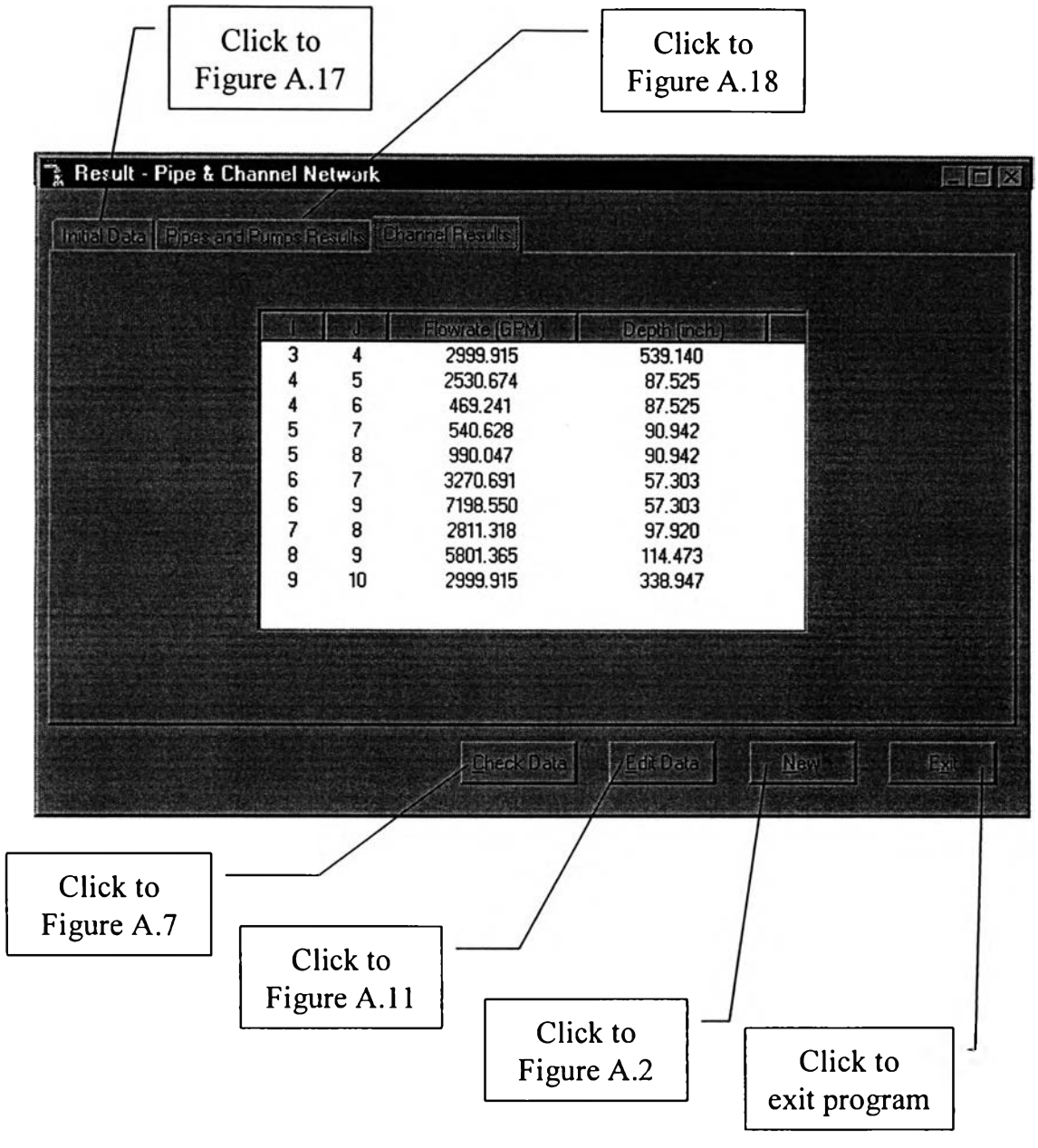

Figure A.19 -View result (Channel results)-

APPENDIX B

SOURCE CODE FOR CALCULATION MODULE

```
Dim Beta As Double, Delta As Double, Dens As Double
Dim Pi As Double, Pj As Double, W As Double
Dim X As Double, Y As Double, Converge As Double, Viscos As Double
Dim Aijchk As Double, tempA As Double, tempB As Double, ED As Double
Dim Rij As Double, Pii As Double, Factor As Double, Pivot As Double
Dim TB As Double, LL As Double
Dim AbsIJ As Integer, Counter As Integer, FR As Integer
Dim I As Integer, II As Integer, Iter As Integer, ITmax As Integer
Dim J As Integer, JJ As Integer, M As Integer, Tm As Integer
Dim JL As Integer, JH As Integer, Npl As Integer, Nband As Integer
Dim ILR As Integer, IrowK As Integer, K As Integer, Kst As Integer
Dim ID As Integer, ini_byte As Integer, dat_byte As Integer
Dim at_byte As Integer, curpos As Integer, N As Integer, Npipe As
Integer
Dim Npump As Integer, NC As Integer, Nchnnl As Integer, wid As
Integer
Dim TypeC1 As Boolean
Dim TypeC3 As Boolean
Dim Inidat As Inidat
Dim IniByte As IniByte
Dim NodeDat() As NodeDat
Dim PipeDat() As PipeDat
Dim Pumpdat() As Pumpdat
Dim Chnnldat() As Chnnldat
Dim PressDat() As PressDat
Dim FlowDat() As FlowDat
Dim ChannelDat() As ChannelDat
Dim FFDat() As FFDat

Private Sub Command1_Click()

    Open App.Path & "\" & Keep.Text1.Text For Binary As #1
    Open App.Path & "\Press.dat" For Binary As #2
    Open App.Path & "\Flow.dat" For Binary As #3
    Open App.Path & "\FF.dat" For Binary As #4
    Open App.Path & "\OpenFlow.dat" For Binary As #5

' Read the initial data
    Get #1, 1, Inidat
    Get #1, 29, IniByte

' Read the pipe data
    ini_byte = IniByte.iniPipe
    ReDim PipeDat(Inidat.Npi) As PipeDat
    dat_byte = Len(PipeDat(1))
    at_byte = IniByte.iniPipe
    For I = 1 To Inidat.Npi
        at_byte = dat_byte * (I - 1) + ini_byte
        Get #1, at_byte, PipeDat(I)
    Next

' Read the pump coefficient data
```

```
    If Inidat.Npm > 0 Then
        ini_byte = IniByte.iniPump
        ReDim Pumpdat(Inidat.Npm) As Pumpdat
        dat_byte = Len(Pumpdat(1))
        at_byte = IniByte.iniPump
        For I = 1 To Inidat.Npm
            at_byte = dat_byte * (I - 1) + ini_byte
            Get #1, at_byte, Pumpdat(I)
        Next I
    End If

' Read the Channel data
    If Inidat.Nch > 0 Then
        ini_byte = IniByte.iniChnnl
        ReDim Chnnldat(Inidat.Nch) As Chnnldat
        dat_byte = Len(Chnnldat(1))
        at_byte = IniByte.iniChnnl
        For I = 1 To Inidat.Nch
            at_byte = dat_byte * (I - 1) + ini_byte
            Get #1, at_byte, Chnnldat(I)
        Next I
    End If

' Read the node data
    ini_byte = IniByte.iniNode
    ReDim NodeDat(Inidat.N) As NodeDat
    dat_byte = Len(NodeDat(1))
    at_byte = IniByte.iniNode
    For I = 1 To Inidat.N
        at_byte = dat_byte * (I - 1) + ini_byte
        Get #1, at_byte, NodeDat(I)
    Next I

' Start to calculate
          N = Inidat.N
     Viscos = Inidat.Vis
       Dens = Inidat.Dens
      Npipe = Inidat.Npi
      Npump = Inidat.Npm
     Nchnnl = Inidat.Nch
      ITmax = Inidat.Imax
         NC = Npipe + Npump + Nchnnl

ReDim PressDat(N) As PressDat
ReDim FlowDat(NC) As FlowDat
ReDim ChannelDat(NC) As ChannelDat
ReDim FFDat(Npipe) As FFDat

ReDim C(N, N) As Integer, FRi(Npipe) As Integer, _
      FRj(Npipe) As Integer, III(NC) As Integer, Irow(N) As Integer,
_
      Jhigh(N) As Integer, JJj(NC) As Integer, Jlow(N) As Integer, _
      NDowChn(N) As Integer, t(N) As Integer, _
      TypeFF(N, N) As Integer, TOpen(N) As Integer

ReDim A(N, N) As Double, AF(N) As Double, Alpha(N, N) As Double, _
      B(N, N) As Double, D(N, N) As Double, Dep(N, N) As Double, _
      Dp(N) As Double, DPP(N) As Double, e(N, N) As Double, _
```

```
      F(N + 1, N + 1) As Double, F_F(N, N) As Double, _
      floww(N) As Double, FRij(N * N) As Double, L(N, N) As Double, _
      Otype(N, N) As Double, P(N) As Double, Q(N, N) As Double, _
      Qij(N * N) As Double, Sigma(N) As Double, Tch(N, N) As Double, _
      v(N) As Double, Wch(N, N) As Double, Z(N) As Double

ReDim Re(N, N) As Double, Fij(N, N) As Double, Qin(N) As Double

' Set all parameter as zero
    For I = 1 To N
            P(I) = 0
            t(I) = 0
            v(I) = 0
            Z(I) = 0
        Sigma(I) = 0
           Dp(I) = 0
          DPP(I) = 0
           AF(I) = 0
        TOpen(I) = 0
        For J = 1 To N
                A(I, J) = 0
                B(I, J) = 0
                C(I, J) = 0
                D(I, J) = 0
                e(I, J) = 0
                L(I, J) = 0
              F_F(I, J) = 0
            Alpha(I, J) = 0
        Next J
    Next I

' Prepare pipe coefficient and set connection matrix of pipe(c = 1)
    For Counter = 1 To Npipe
            D(PipeDat(Counter).Ni, PipeDat(Counter).Nj) = PipeDat
(Counter).D
            L(PipeDat(Counter).Ni, PipeDat(Counter).Nj) = PipeDat
(Counter).L
          F_F(PipeDat(Counter).Ni, PipeDat(Counter).Nj) = PipeDat
(Counter).FF
            e(PipeDat(Counter).Ni, PipeDat(Counter).Nj) = PipeDat
(Counter).Rough
       TypeFF(PipeDat(Counter).Ni, PipeDat(Counter).Nj) = PipeDat
(Counter).TFF
            C(PipeDat(Counter).Ni, PipeDat(Counter).Nj) = 1
    Next

' Prepare pump coefficient and set connection matrix of pump (c = 2)
    If Npump > 0 Then
      For Counter = 1 To Npump
         A(Pumpdat(Counter).Ni, Pumpdat(Counter).Nj) = Pumpdat
(Counter).A
         B(Pumpdat(Counter).Ni, Pumpdat(Counter).Nj) = Pumpdat
(Counter).B
         C(Pumpdat(Counter).Ni, Pumpdat(Counter).Nj) = 2
      Next
    End If
```

```
' Prepare channel data
    If Nchnnl > 0 Then
      For Counter = 1 To Nchnnl
         C(Chnnldat(Counter).Ni, Chnnldat(Counter).Nj) = 3
        Wch(Chnnldat(Counter).Ni, Chnnldat(Counter).Nj) = Chnnldat
(Counter).W
         L(Chnnldat(Counter).Ni, Chnnldat(Counter).Nj) = Chnnldat
(Counter).L
        F_F(Chnnldat(Counter).Ni, Chnnldat(Counter).Nj) = Chnnldat
(Counter).FF
        Dep(Chnnldat(Counter).Ni, Chnnldat(Counter).Nj) = Chnnldat
(Counter).Dep
        Tch(Chnnldat(Counter).Ni, Chnnldat(Counter).Nj) = Chnnldat
(Counter).Tch
      Next
    End If
' Prepare Node data
    For Counter = 1 To N
         Z(NodeDat(Counter).NNo) = NodeDat(Counter).Z
         P(NodeDat(Counter).NNo) = NodeDat(Counter).P
         If NodeDat(Counter).Type <> 0 Then
            t(NodeDat(Counter).NNo) = NodeDat(Counter).Type
            If NodeDat(Counter).Type = 2 Or NodeDat(Counter).Type =
3 Then
                If NodeDat(Counter).Dir = 1 Then
                v(NodeDat(Counter).NNo) = NodeDat(Counter).Flow
                Else: v(NodeDat(Counter).NNo) = -NodeDat
(Counter).Flow
                End If
            End If
         End If
    Next

' Compute constant conversion unit from the flow rate (British Unit)
      Beta = Dens / 144
       Pii = 3.14159265
    Factor = 32 * 12 ^ 5 / (Pii ^ 2 * 144 * 32.2 * (7.48 * 60) ^ 2)

' Compute constant conversion unit from the flow rate (SI)
  '    Beta = 9.81 * Dens / (1.01325 * 10 ^ 5)
  '     Pii = 3.14159265
  '  Factor = 32 * 10 ^ 15 / (Pii ^ 2 * (3600) ^ 2 * 1.01325 * 10 ^
5)

' Compute Bandwidth
    Nband = 0
    For I = 1 To N
        For J = 1 To N
            If (C(I, J) <> 0) Then
                If Abs(I - J) > Nband Then
                    AbsIJ = Abs(I - J)
                    wid = AbsIJ
                End If
            End If
        Next J
        Nband = wid
    Next I
```

```
' Set lower and upper limit of each column
    For I = 1 To N

        If 1 > I - Nband Then
            Jlow(I) = 1
        Else: Jlow(I) = I - Nband
        End If
        If N < I + Nband Then
            Jhigh(I) = N
        Else: Jhigh(I) = I + Nband
        End If
        JL = Jlow(I)
        JH = Jhigh(I)
        TypeC1 = False: TypeC3 = False
        For J = JL To JH
          If I <> J Then
                D(J, I) = D(I, J)
                e(J, I) = e(I, J)
                L(J, I) = L(I, J)
              F_F(J, I) = F_F(I, J)
              Wch(J, I) = Wch(I, J)
           TypeFF(J, I) = TypeFF(I, J)
            If C(I, J) = 1 Then
                C(J, I) = C(I, J)
              Alpha(J, I) = Factor * Dens * L(J, I) / (D(J, I) ^ 5)
            End If
            If C(I, J) = 3 Then C(J, I) = C(I, J)
          End If

          If C(I, J) = 1 Or C(I, J) = 2 Then TypeC1 = True
          If C(I, J) = 3 Then TypeC3 = True

          If TypeC1 Then
                    TOpen(I) = 0
          End If
          If TypeC3 Then
                    TOpen(I) = 2
          End If
          If TypeC1 And TypeC3 Then
                    TOpen(I) = 1
          End If


        Next J
    Next I

' ********* Start the Iteration ***********
For Iter = 1 To ITmax

' Set all element of F(i,j) = 0
    Npl = N + 1
      For I = 1 To N
        For J = 1 To Npl
            F(I, J) = 0
        Next J
      Next I
' Set lower and upper limit of each row
    For I = 1 To N
```

```
        JL = Jlow(I)
        JH = Jhigh(I)
' Checking type of node
    If t(I) <> 1 Then
            F(I, Npl) = -v(I)

        For J = JL To JH

        If J <> I Then
            If C(J, I) = 1 Then
                Y = P(J) - P(I) + Beta * (Z(J) - Z(I))
                If t(I) = 3 Then
                    F(I, I) = 1
                    F(I, J) = -1
                    F(I, Npl) = v(I) * Abs(v(I)) * Alpha(J, I) * F_F
(J, I) + Y
                Else
                    If Y <> 0 Then
                      If Y < 0 Then
                            F(I, J) = 0.5 * Sqr(-1 / (Alpha(J, I) *
F_F(J, I) * Y))
                            F(I, I) = F(I, I) - 0.5 * Sqr(-1 /
(Alpha(J, I) * F_F(J, I) * Y))
                          F(I, Npl) = F(I, Npl) + Sqr(-Y / (Alpha(J,
I) * F_F(J, I)))
                      Else
                            F(I, J) = 0.5 * Sqr(1 / (Alpha(J, I) *
F_F(J, I) * Y))
                            F(I, I) = F(I, I) - 0.5 * Sqr(1 / (Alpha
(J, I) * F_F(J, I) * Y))
                          F(I, Npl) = F(I, Npl) - Sqr(Y / (Alpha(J,
I) * F_F(J, I)))
                      End If
                    End If
                End If

            ElseIf C(J, I) = 2 Then
                W = P(J) - P(I) + A(J, I) + Beta * (Z(J) - Z(I))
                If W > 0 Then
                    If P(J) <= P(I) Then
                        F(I, J) = 0.5 * Sqr(1 / (B(J, I) * W))
                        F(I, I) = F(I, I) - 0.5 * Sqr(1 / (B(J, I) *
W))
                      F(I, Npl) = F(I, Npl) - Sqr(W / B(J, I))
                    Else
                      F(I, Npl) = F(I, Npl) - Sqr(A(J, I) / B(J, I))
                    End If
                End If
            ElseIf C(I, J) = 2 Then
                W = P(I) - P(J) + A(I, J) + Beta * (Z(I) - Z(J))
                If W > 0 Then
                    If P(I) <= P(J) Then
                        F(I, J) = 0.5 * Sqr(1 / (B(I, J) * W))
                        F(I, I) = F(I, I) - 0.5 * Sqr(1 / (B(I, J) *
W))
                      F(I, Npl) = F(I, Npl) + Sqr(W / B(I, J))
                    Else
                      F(I, Npl) = F(I, Npl) + Sqr(A(I, J) / B(I, J))
```

```
                End If
            End If
        End If
    End If
    Next J
Else
    F(I, I) = 1
End If
Next I

' Use the subroutine  SGEM(Speacial Guessian Elimination Method) to
solved
' the element of coefficient in banded matrix
GoSub SGEM

    For I = 1 To N
        AF(I) = 1 / 3
        Dp(I) = F(I, N + 1)
        Dp(I) = Dp(I) * AF(I)
         P(I) = P(I) + Dp(I)
    Next I

' Check for convergence
    Converge = 0
    For I = 1 To N
        Delta = Dp(I)
        If Abs(Delta) > Converge Then
            Converge = Abs(Delta)
        End If
    Next I

' Calculate flow in all pipes
    For I = 1 To N
        floww(I) = 0
        For J = 1 To N
            Q(I, J) = 0
        Next J
    Next I

    For I = 1 To N
        JL = Jlow(I)
        JH = Jhigh(I)
        For J = JL To JH
            If J <> I Then
        ' compute flowrate in case of pipeline
                If C(I, J) = 1 Then
                    X = (P(I) - P(J) + Beta * (Z(I) - Z(J))) /
(Alpha(I, J) * F_F(I, J))
                  If X <> 0 Then
                    If X > 0 Then
                        Q(I, J) = Sqr(X)
                    Else: Q(I, J) = -Sqr(-X)
                    End If
                  End If
        ' in case of pump
                ElseIf C(J, I) = 2 Then
                    W = P(J) - P(I) + A(J, I) + Beta * (Z(J) - Z(I))
```

```
                    Pi = P(I) + Beta * Z(I)
                    Pj = P(J) + Beta * Z(J)
                    If W > 0 Then
                        If Pj > Pi Then
                            Q(I, J) = -Sqr(A(J, I) / B(J, I))
                        Else: Q(I, J) = -Sqr(W / B(J, I))
                        End If
                    End If


                ElseIf C(I, J) = 2 Then
                    W = P(I) - P(J) + A(I, J) + Beta * (Z(I) - Z(J))
                    Pi = P(I) + Beta * Z(I)
                    Pj = P(J) + Beta * Z(J)
                    If W > 0 Then
                        If Pi > Pj Then
                            Q(I, J) = Sqr(A(I, J) / B(I, J))
                        Else: Q(I, J) = Sqr(W / B(I, J))
                        End If
                    End If
                End If
            End If
        Next J
    Next I

' Calculate net flow at every node
    For I = 1 To N
        JL = Jlow(I)
        JH = Jhigh(I)
        For J = JL To JH
            If Abs(Q(I, J)) <> 0 Then
               floww(I) = floww(I) + Q(I, J)
            End If
        Next
    Next

    If Nchnnl > 0 Then GoSub CalChannel:


'Check convergence
    If Converge < 0.00001 Then
        curpos = 1
        For I = 1 To N
            PressDat(I).Ni = I
            PressDat(I).P = P(I)
            Put #2, curpos, PressDat(I)
            curpos = Seek(2)
        Next
        Exit For 'Iter
    End If

    GoSub ApproxFF


    ProgressBar1.Value = Iter * 100 / ITmax

Next Iter
```

```
' *******  end of iteration *******

    Tm = 0
    FR = 0
    For I = 1 To N
        For J = I To N
        If C(I, J) = 1 Then
            Tm = Tm + 1
            III(Tm) = I
            JJj(Tm) = J
            FR = FR + 1
            FRi(FR) = I
            FRj(FR) = J
            FRij(FR) = F_F(I, J)
        End If
        If C(I, J) = 3 Then
            Tm = Tm + 1
            III(Tm) = I
            JJj(Tm) = J
        End If

        Next J
    Next I  .

    If Converge > 0.00001 Then
        Label3.Caption = "iterations gave no convergence"
        MsgBox "The iterations gave no convergence ", 48,
"Convergence report"
    ElseIf Iter = 1 Then
           Label3.Caption = "iteration gave convergence"
        Else: Label3.Caption = "iterations gave convergence"
    End If

    curpos = 1
    For I = 1 To N
        PressDat(I).Ni = I
        PressDat(I).P = P(I)
        Put #2, curpos, PressDat(I)
        curpos = Seek(2)
    Next

' Put the calculated data to files
    curpos = 1
    For M = 1 To Tm
        If C(III(M), JJj(M)) = 1 Or C(III(M), JJj(M)) = 2 Then
            FlowDat(M).Ni = III(M)
            FlowDat(M).Nj = JJj(M)
            FlowDat(M).Q = Q(III(M), JJj(M))
            curpos = Seek(3)
            Put #3, curpos, FlowDat(M)
        ElseIf C(III(M), JJj(M)) = 3 Then
            ChannelDat(M).Ni = III(M)
            ChannelDat(M).Nj = JJj(M)
            ChannelDat(M).Q = Q(III(M), JJj(M))
            ChannelDat(M).Dep = Dep(III(M), JJj(M))
            curpos = Seek(5)
            Put #5, curpos, ChannelDat(M)
```

```
        End If
    Next
    curpos = 1
    For M = 1 To FR
        FFDat(M).Ni = FRi(M)
        FFDat(M).Nj = FRj(M)
        FFDat(M).Fij = FRij(M)
        Put #4, curpos, FFDat(M)
        curpos = Seek(4)
    Next
    Close #1, #2, #3, #4, #5
    ProgressBar1.Value = 100
    If Iter > ITmax Then Iter = ITmax
    Label2.Caption = Iter

Calculate.Visible = False

'Unload Me
ViewResult2.Visible = True

Exit Sub

'**************  End of main program **********************

SGEM:
' start L-U Decomposition loop
    Kst = 1
    For K = 1 To N
        ILR = K + Nband
        If ILR > N Then
            ILR = N
        End If
        Pivot = 0
        J = Kst
        For I = K To ILR
            If Abs(F(I, J)) > Abs(Pivot) Then
                Pivot = F(I, J)
                Irow(K) = I
            End If
        Next I
        If Pivot = 0 Then
            MsgBox "Some pivot = 0, Cannot proceed anymore." + Chr
(13) + Chr(13) _
                            + "Please check your data.", 48, "Error"
            Close #1, #2, #3, #4, #5
            Unload Me
            Piping6.Visible = True
            Exit Sub
        End If
' Normalize pivot row element
        Npl = N + 1
        IrowK = Irow(K)
        For J = K To Npl
            F(IrowK, J) = F(IrowK, J) / Pivot
        Next J
        If K <> IrowK Then
            For J = K To Npl
                Tm = F(K, J)
```

```
                F(K, J) = F(IrowK, J)
                F(IrowK, J) = Tm
            Next
        End If
        II = K + 1
        JJ = Kst
        For I = II To ILR
            Aijchk = -F(I, JJ)
          For J = JJ To Npl
            F(I, J) = F(I, J) + Aijchk * F(JJ, J)
          Next J
        Next I
        Kst = Kst + 1

    Next K
' End of L-U Decompositon Loop

' Back substitution
    II = N
    For I = 2 To N
        II = II - 1
        ID = II
        TB = F(II, Npl)
        ID = ID + 1
      For JJ = ID To N
          TB = TB - F(II, JJ) * F(JJ, Npl)
      Next JJ
          F(II, Npl) = TB
    Next I

Return
Exit Sub

ApproxFF:

    Pii = 3.14159265
    For I = 1 To N
        For J = 1 To N
              Q(I, J) = 0
             Re(I, J) = 0
            Fij(I, J) = 0
        Next J
    Next I

    For I = 1 To N
        JL = Jlow(I)
        JH = Jhigh(I)
        For J = JL To JH
           If J <> I Then
             If TypeFF(J, I) = 1 Or TypeFF(I, J) = 1 Then
                If C(I, J) = 1 Then
                    Y = (P(I) - P(J) + Beta * (Z(I) - Z(J))) /
(Alpha(I, J) * F_F(I, J))
                  If Y <> 0 Then
                    If Y > 0 Then
                        Q(I, J) = Sqr(Y)
                    Else: Q(I, J) = -Sqr(-Y)
                    End If
```

```
            'for British unit
            Re(I, J) = 158.99 * Dens * Abs(Q(I, J)) / (Pii * Viscos
* D(I, J))
            ' for SI
            '       Re(I, J) = 111.111 * Dens * Abs(Q(I, J)) / (Pii
* Viscos * D(I, J))

                ' for RE > 4000
                   If Re(I, J) > 4000 Then
                       ED = 0.269 * 12 * e(I, J) / D(I, J)
                       Rij = 2.185 / Re(I, J)
                       tempA = Log(ED + (14.5 / Re(I, J)))
                       tempB = Log(ED - (Rij * tempA))
                       Fij(I, J) = (-1.737 * tempB) ^ (-2)
                   ElseIf Re(I, J) < 2000 Then
                       Fij(I, J) = 16 / Re(I, J)
                   Else: Fij(I, J) = F_F(I, J)
                   End If
                 End If
              Else: Fij(I, J) = F_F(I, J)
              End If
            Else: Fij(I, J) = F_F(I, J)
            End If
          End If
        Next J
    Next I
    ' Return value Fij to FF
    For I = 1 To N
        JL = Jlow(I)
        JH = Jhigh(I)
        For J = JL To JH
            If J <> I Then
                If C(J, I) = 1 Then
                    F_F(J, I) = Fij(J, I)
                End If
            End If
        Next J
    Next I

Return
'*********  End of subroutine ApproxFF ***************


' *********** Subroutine Calculate Depth ***************
CalDepth:

    dZ = Abs(Z(I) - Z(J))
    W = Wch(I, J)
    flo = Q(I, J) * 0.1337 / 60

    FF = F_F(I, J)
    LL = L(I, J)
    Dept = Dep(I, J)
    T1 = 4 * (32.2) * dZ * W ^ 3
    T2 = 0
    T3 = -1 * 4 * LL * FF * flo ^ 2
    T4 = -1 * 2 * LL * FF * W * flo ^ 2
```

```
    A1 = T2 / T1
    A2 = T3 / T1
    A3 = T4 / T1
    QQ = 1 / 9 * (-3 * A2 + 0)
    RR = 1 / 54 * (0 + 27 * A3 + 0)
    gramm = QQ ^ 3 - RR ^ 2
    If gramm > 0 Then
        Xx = RR / QQ ^ (3 / 2)
        Beeta = Atn(-Xx / Sqr(-Xx * Xx + 1)) + 2 * Atn(1)
         dep1 = -2 * Sqr(QQ) * Cos(Beeta / 3) - 0
         dep2 = -2 * Sqr(QQ) * Cos(Beeta / 3 + 2 / 3 * Pii) - 0
         dep3 = -2 * Sqr(QQ) * Cos(Beeta / 3 + 4 / 3 * Pii) - 0
        If dep1 > 0 Then
            Dep(I, J) = dep1 * 12
        ElseIf dep2 > 0 Then
            Dep(I, J) = dep2 * 12
        ElseIf dep3 > 0 Then
            Dep(I, J) = dep3 * 12
        End If
    Else
        del = (Abs(RR) + Sqr(-1 * gramm)) ^ (1 / 3)
        Dep(I, J) = (-1 * Sgn(RR) * (del + QQ / del) - 0) * 12
    End If

Return
' **************  End fo Subroutine CalDepth ***************

' ************** Subroutine CalChannel ************************
CalChannel:
' Define the flow in the upstream channel
    For I = 1 To N
        If TOpen(I) = 1 And floww(I) < 0 Then
            For J = Jlow(I) To Jhigh(I)
                If C(I, J) = 3 Then
                    Q(I, J) = Abs(floww(I))
                    GoSub CalDepth:
                    GuessD = Dep(I, J)
                End If
            Next
        End If
    Next

' Calculate the flow rate and depth at each channel.
    For I = 1 To N
        Qin(I) = 0
        NDowChn(I) = 0
        For J = Jlow(I) To Jhigh(I)
          If C(I, J) = 3 Then
            If TOpen(I) = 2 Then
              If Z(I) > Z(J) Then
                Otype(I, J) = -1
                NDowChn(I) = NDowChn(I) + 1
              End If
            End If
          End If
        Next J
    Next I
```

```
    For I = 1 To N
        JL = Jlow(I)
        JH = Jhigh(I)
        If TOpen(I) = 2 And NDowChn(I) > 1 Then
            For J = JL To JH
               If C(I, J) = 3 Then
                 If Z(I) < Z(J) Then
                    floww(I) = floww(I) + Q(J, I)
                 End If
               End If
            Next
            Qin(I) = floww(I)
            GoSub Branching:
        ElseIf NDowChn(I) = 1 Then
            For J = JL To JH
               If C(I, J) = 3 Then
                 If Z(I) < Z(J) Then
                    floww(I) = floww(I) + Q(J, I)
                 End If
               End If
            Next
            For J = JL To JH
               If C(I, J) = 3 Then
                 If Z(I) > Z(J) Then
                    Q(I, J) = Abs(floww(I))
                    GoSub CalDepth:
                    GuessD = Dep(I, J)
                 End If
               End If
            Next
        End If
    Next I

Return


' ************** Subroutine Branching ***************************
Branching:
    Qkeep = Qin(I)
    Dkeep = GuessD / 3
    FixI = I
    NDC = NDowChn(FixI)
    ijk = 0
Do
    Qkeep = Qin(FixI)
    NDC = NDowChn(FixI)
    For J = JL To JH
        If Otype(FixI, J) = -1 Then

            If NDC > 1 Then
              Dep(FixI, J) = Dkeep
              Num = 32.2 * (Z(FixI) - Z(J)) * 4 * (Wch(FixI, J) _
                    * Dep(FixI, J) / 12) ^ 3
              Denom = 2 * L(FixI, J) * F_F(FixI, J) _
                      * (2 * Dep(FixI, J) / 12 + Wch(FixI, J))
              Q(FixI, J) = Sqr(Num / Denom) * 60 / 0.1337
              Qkeep = Qkeep - Q(FixI, J)
```

```
                NDC = NDC - 1
            Else:
                Q(FixI, J) = Qkeep
                GoSub CalDepth:
                Dlast = Dep(FixI, J)
            End If
        End If
    Next J
    If Abs(Dlast - Dkeep) < 0.00001 Then
        Exit Do
    Else
        Sum = 0
        For J = JL To JH
            If Otype(FixI, J) = -1 Then
                Sum = Sum + Dep(FixI, J)
            End If
        Next
        Dkeep = Sum / NDowChn(FixI)
    End If
    ijk = ijk + 1
    If ijk = 100 Then Exit Do

Loop Until Abs(Dlast - Dkeep) < 0.00001

Return


End Sub
```

CURRICULUM VITAE

Name : Mr. Athapol Kitiyanan

Birth Date : July 20, 1976

Nationality : Thai

University Education:

1994-1997 Bachelor's Degree of Science in Chemical Technology, Chulalongkorn University