Option Explicit

Private Sub Command1_Click()
    Option Explicit
    Dim numnp As Integer, numel As Integer, n As Integer
    Dim i As Integer, j As Integer, k As Integer, ii As Integer
    Dim nqpts As Integer, ib As Integer, Lu As Integer, iprint As Integer
    Dim k1 As Integer, L0 As Integer, L1 As Integer
    Dim s1(2, 2) As Double, s2(2, 2) As Double, s3(2, 2) As Double
    Dim sq(2) As Double, dndx(2) As Double
    Dim gpts(6) As Double, gwts(6) As Double
    Dim diff As Double, dx As Double, dq As Double, du As Double
    Dim rl As Double, xi As Double, dxds As Double, dsdx As Double
    Dim xl As Double, xg As Double, s As Double, sw As Double

    Dim numnp As Integer, numel As Integer, n As Integer
    Dim l As Integer, ll As Integer
    Dim i As Integer, j As Integer, k As Integer, ii As Integer
    Dim nqpts As Integer, ib As Integer, Lu As Integer, iprint As Integer
    Dim k1 As Integer, L0 As Integer, L1 As Integer
    Dim s1(2, 2) As Double, s2(2, 2) As Double, s3(2, 2) As Double
    Dim sq(2) As Double, dndx(2) As Double
    Dim gpts(6) As Double, gwts(6) As Double
    Dim diff As Double, dx As Double, dq As Double, du As Double
    Dim rl As Double, xi As Double, dxds As Double, dsdx As Double
    Dim xl As Double, xg As Double, s As Double, sw As Double

Rem**********************************************************************
Rem  Gaussian   Quadrature Points and Weights                           *
Rem**********************************************************************
    gpts(1) = 0.93246951: gpts(2) = 0.66120938: gpts(3) = 0.23861018
    gpts(4) = -gpts(3): gpts(5) = -gpts(2): gpts(6) = -gpts(1)
    gwts(1) = 0.17132449: gwts(2) = 0.36076157: gwts(3) = 0.46791393
    gwts(4) = gwts(3): gwts(5) = gwts(2): gwts(6) = gwts(1)

    Picture1.Cls
    Picture2.Cls

Rem**********************************************************************
Rem  OPEN INPUT AND OUTPUT FILES                                      *
Rem**********************************************************************
    Call OpenFile
    Input #1, numnp
    numel = numnp - 1
    Picture1.Print
    Picture1.Print "Number of nodes   ": numel
Sub Print
  Print "Number of elements ": numel
End Sub

ReDim npbc(1 To numnp) As Integer
ReDim xord(1 To numnp) As Double
ReDim yord(1 To numnp) As Double
ReDim u(1 To numnp) As Double
ReDim q(1 To numnp) As Double
ReDim sk(1 To numnp, 3) As Double

l = 1
Do While Not EOF(1)
  Input #1, n
  Input #1, npbc(n), xord(n), u(n), q(n)
  ll = n - l
  If ll > 0 Then
    diff = n + 1 - l
    dx = (xord(n) - xord(l - 1)) / diff
    du = (u(n) - u(l - 1)) / diff
    dq = (q(n) - q(l - 1)) / diff
    For i = l To n - 1
      npbc(i) = npbc(i - 1)
      xord(i) = xord(i - 1) + dx
      u(i) = u(i - 1) + du
      q(i) = q(i - 1) + dq
      l = l + 1
    Next i
  End If
  l = l + 1
Loop
Close #1

Rem============================
Rem      ECHO INPUT DATA      =
Rem============================
Call PrintHeader
For i = 1 To numnp
  Print #2, i, npbc(i), FormatNumber(xord(i), 5), FormatNumber(u(i), 5),
  FormatNumber(q(i), 5)
Next i

Rem=================================
Rem      BEGIN INITIALIZATION      =
Rem=================================
nqpts = 6
ib = 3
Lu = 1
For i = 1 To numnp
  For j = 1 To ib
    sk(i, j) = 0
  Next j
Next i

Rem=================================
Rem      FORMATION OF SK MATRIX      =
Rem=================================
For i = 1 To numel
  rl = xord(i + 1) - xord(i)
  xi = xord(i)
  dxds = rl / 2
  dsdx = 1 / dxds
  For j = 1 To 2
    For k = 1 To 2
      s1(j, k) = 0
      s2(j, k) = 0
      s3(j, k) = 0
    Next k
  Next j
sq(j) = 0
Next j

Rem=============================================================================
Rem BEGIN GAUSSIAN QUADRATURE =
Rem=============================================================================
For j = 1 To nqpts
  xl = (rl / 2) + (rl * gpts(j)) / 2
  xg = xi + xl
  s = gpts(j)
  sw = gwts(j)
  dndx(1) = dsn(1, s) * dsdx
  dndx(2) = dsn(2, s) * dsdx
  For k = 1 To 2
    For l = 1 To 2
      s1(k, l) = *********** BLEEP **********
      s2(k, l) = s2(k, l) + sn(k, s) * b(xg) * dndx(l) * sw * dxds
      s3(k, l) = *********** BLEEP **********
    Next l
    sq(k) = sq(k) + sn(k, s) * d(xg) * gwts(j) * dxds
  Next k
Next j

Rem=============================================================================
Rem PLACE ELEMENT MATRICES INTO GLOBAL MATRIX =
Rem=============================================================================
k1 = *********** BLEEP **********
L0 = 2
For k = 1 To 2
  L0 = L0 - 1
  k1 = k1 + 1
  L1 = L0
  For l = 1 To 2
    L1 = L1 + 1
    sk(k1, L1) = sk(k1, L1) + s1(k, l) - s2(k, l) - s3(k, l)
  Next l
  q(k1) = q(k1) + sq(k)
Next k
Next i

Rem=============================================================================
Rem ALL ELEMENTS ARE NOW ASSEMBLED IN GLOBAL MATRIX EQUATION. =
Rem KNOWN VALUES OF U(I) MUST BE ACCOUNTED FOR. =
Rem=============================================================================
For i = 1 To numnp
  If npbc(i) <> 0 Then
    sk(i, 2) = sk(i, 2) * (1E+35)
    q(i) = u(i) * sk(i, 2)
  End If
Next i

For i = 1 To numnp
  u(i) = q(i)
Next i

Call LDU(sk, u, Lu, numnp, ib, 101, 3)

Rem=============================================================================
Rem OUTPUT DATA =
Rem=============================================================================
Print #2, "Output Data"
Print #2, "-----------------------------------------------"
Print #2, "Node  x    u  "
Print #2, "-----------------------------------------------"
For i = 1 To numnp
  Print #2, i, FormatNumber(xord(i), 5), FormatNumber(u(i), 5)
Next i
Close #2
Picture1.Print
Picture1.Print "ODE2 analysis complete"
Program ODE2

Private Sub Command2_Click()
End Sub

Public Function sn(i, x) As Double
If i = 1 Then
  sn = *********** BLEEP **********
Else
  sn = 0.5 * x + 0.5
End If
End Function

Public Function dsn(i, x) As Double
If i = 1 Then
  dsn = *********** BLEEP **********
Else
  dsn = 0.5
End If
End Function

Public Function a(x) As Double
a = x ^ 2
End Function

Public Function b(x) As Double
b = 0
End Function

Public Function c(x) As Double
c = -1
End Function

Public Function d(x) As Double
d = 0
End Function

Public Sub LDU(z() As Double, y() As Double, Lu, numeq, ib, idim, jdim)
Dim nem1 As Integer, ibd2 As Integer, idiag As Integer
Dim i As Integer, j As Integer, k As Integer, k1 As Integer, k2 As Integer
Dim jend As Integer, jbgn As Integer
Dim jrow As Integer, jcol As Integer, irow As Integer
Dim fac As Double
If numeq > idim Then
  Picture1.Print "NUMBER OF EQUATIONS EXCEEDS IDIM"
End If
If ib > jdim Then
  Picture1.Print "BAND WIDTH EXCEEDS JDIM"
End If
nem1 = numeq - 1
ibd2 = (ib - 1) / 2

priv  BEGIN FORWARD ELIMINATION
idiag = ibd2 + 1
For i = 1 To nem1
  jend = numeq - i
  If jend > ibd2 Then jend = ibd2
  For j = 1 To jend
    jrow = 1 + j
    fac = z(jrow) / z(irow)
    For k = i + 1 To jend
      z(k + jrow) = z(k + jrow) - fac * z(k + irow)
    Next
  Next
Next
End Sub
}
jcol = ibd2 - j + 1

If Lu <> 0 Then
    fac = z(jrow, jcol) / z(i, idiag)
    z(jrow, jcol) = fac
    For k = 1 To jend
        k1 = idiag + k
        k2 = jcol + k
        z(jrow, k2) = z(jrow, k2) - fac * z(i, k1)
    Next k

End If

y(jrow) = y(jrow) - z(jrow, jcol) * y(i)

Next j
Next i

Rem BEGIN BACK SUBSTITUTION

y(numeq) = y(numeq) / z(numeq, idiag)

jbgn = idiag + 1

For i = 1 To nem1
    irow = numeq - i
    jend = idiag + i
    If jend > ib Then jend = ib
    jrow = irow
    For j = jbgn To jend
        jrow = jrow + 1
        y(irow) = y(irow) - z(irow, j) * y(jrow)
    Next j
    y(irow) = y(irow) / z(irow, idiag)
Next i

End Sub

Public Sub DrawPlot(x() As Double, y() As Double, numnp)
    Dim xscale As Double, yscale As Double, offset As Double
    Dim xleft As Double, xright As Double, ybottom As Double, ytop As Double
    Dim xmin As Double, xmax As Double, ymin As Double, ymax As Double
    Dim xp1 As Double, xp2 As Double, yp1 As Double, yp2 As Double
    Dim xdiff As Double, ydiff As Double
    Dim i As Integer
    xscale = 1000
    yscale = 1000
    offset = 0.2
    xleft = -offset * xscale
    xright = (1 + offset) * xscale
    ybottom = -offset * yscale
    ytop = (1 + offset) * yscale
    Picture2.DrawWidth = 2
    Picture2.Scale (xleft, ytop)-(xright, ybottom)
    Picture2.Line (0, 0)-(xright - offset * xscale / 2, 0)
    Picture2.Line (0, 0)-(0, ytop - offset * yscale / 2)
    xmax = 0
    ymax = 0
    xmin = 1E+34
    ymin = 1E+34
    For i = 1 To numnp
        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
    xdiff = xmax - xmin
    ydiff = ymax - ymin
    ydiff = ymax - ymin

For i = 1 To numnp
  xp1 = ((x(i) - xmin) / xdiff) * xscale
  yp1 = ((y(i) - ymin) / ydiff) * yscale
  If i < numnp Then
    'Draw lines connecting data points
    xp2 = ((x(i + 1) - xmin) / xdiff) * xscale
    yp2 = ((y(i + 1) - ymin) / ydiff) * yscale
    Picture2.DrawLine (xp1, yp1)-(xp2, yp2)
  End If
  'Draw small circles around data points
  Picture2.Circle (xp1, yp1), xscale / 200
Next i

'Draw xmin and xmax
Picture2.CurrentX = 0
Picture2.CurrentY = 0 - yscale * offset / 4
Picture2.Print x(1)
Picture2.CurrentX = xscale
Picture2.CurrentY = 0 - yscale * offset / 4
Picture2.Line (xscale, 0)-(xscale, -yscale * offset / 6)
Picture2.CurrentX = xscale
Picture2.CurrentY = 0 - yscale * offset / 4
Picture2.Print x(numnp)

'Draw ymin and ymax
Picture2.CurrentX = 0 - xscale * 3 * offset / 4
Picture2.CurrentY = 0
Picture2.Print FormatNumber(ymin, 3)
Picture2.CurrentX = 0 - xscale * 3 * offset / 4
Picture2.CurrentY = yscale
Picture2.Print FormatNumber(ymax, 3)
Picture2.CurrentX = 0 - xscale * 3 * offset / 4
Picture2.CurrentY = yscale
Picture2.Line (0, yscale)-(-xscale * offset / 6, yscale)
End Sub

Public Sub PrintHeader()
  Print #2, " **************************************************************"
  Print #2, " **
  Print #2, " ***           ODE2 – The University of Memphis***
  Print #2, " **
  Print #2, " **************************************************************"
  Print #2, "Input Data"
  Print #2, "---------------------------------------------------------------------"
  Print #2, "Node         Code              x             u             q 
  Print #2, "---------------------------------------------------------------------"
End Sub

Public Sub OpenFile()
On Error GoTo ErrorHandler
Dim file As String, gAppDir As String, inputfile As String, outputfile As String
file = Text1.Text
gAppDir = App.Path
inputfile = gAppDir & "\" & file & ".in"
Open inputfile For Input As #1
outputfile = gAppDir & "\" & file & ".out"
Picture1.Cls
Picture1.Print "Input file: ", inputfile
Picture1.Print "Output file: ", outputfile
Open outputfile For Output As #2
End Sub

ErrorHandler:
Select Case Err.Number
  Case 53
    MsgBox "File not Found. Try Again"
    file = InputBox("Enter input file name *.in")
gAppDir = App.Path
inputfile = gAppDir & "\" & file & ".in"
Text1.Text = file
End Select
Resume
End Sub