NEAR-SURFACE SEISMIC VELOCITY DATA: A COMPUTER PROGRAM FOR ANALYSIS

G. I. Alaminiokuma+*, E. D. Uko+, and C. Israel-Cookey++
+Department of Physics and ++Department of Mathematics,
Rivers State University of Science and Technology,
P.M. B. 5080, Port Harcourt, Nigeria.

(Submitted: 23 June, 2006; Accepted: 18 October, 2007)

Abstract

A computer program (NESURVELANA) has been developed in Visual Basic Computer programming language to carry out a near surface velocity analysis. The method of analysis used includes: Algorithms design and Visual Basic codes generation for plotting arrival time (ms) against geophone depth (m) employing the Least-Squares approximation to fit the best lines, computing velocities from the reciprocals of the slopes of the lines and determining thickness for multi-layer cases. The program was debugged and test-run on Microsoft Windows XP using a set of two near-surface seismic velocity data acquired by the Up-hole technique within the South-Central Niger Delta, Nigeria. The first set of data showed a weathered two-layer case, where the layer has a thickness of 6.0 m and velocity of 431 m/s and a consolidated layer velocity of 1845 m/s. The second set of data was a weathered three-layer case, where the thicknesses were 4.3 m and 7.7 m with the weathered and second layer with velocities of 513 m/s and 1132 m/s respectively and a consolidated layer velocity of 1756 m/s for the third layer. These results are comparable with those obtained by other researchers for the Niger Delta, Nigeria. The resulting information from this near-surface velocity analysis is essential for static corrections for the accurate mapping of the underlying structures for oil and gas exploration, and geotechnical engineering for foundation works in building houses, bridges, dams and construction of highways.

Keywords: *Algorithm design, visual basic codes, least-squares approximation, weathered layer thickness and velocity.*

Introduction

The analysis of near-surface velocity data in the Niger Delta has become indispensable in the search for underground water, oil and gas. It is essential in determining the time delays needed for static corrections during seismic reflection data processing as highlighted by (Akpabio and Onwusiri, 2004; Eze, *et al.* 2003), and in geotechnical engineering for the establishment of bedrock for foundation works in building houses, bridges, dams and construction of highways as studied by (Okwueze, et al., 1992; Uko, et al., 1992) among others. Such analysis by computer programs (software) minimizes slowness and time wastage thereby greatly lessening strenuous iterative routines, eliminates unnecessary repetitions involved in making modifications, allows for updating of previous work in the light of experience, and make results more reliable.

In these days of fast computers, powerful software have been extensively used by the major oil exploration companies such as Nigerian Agip Oil Company, Total Petroleum Nig. Ltd., and Shell Petroleum Development Company Nig. Ltd., Mobil Producing Nigerian Unltd. etc. However, the software available in these

**Corresponding author*

companies have largely remained classified and unpublished.

In line with this trend, one of such computer programs (NESURVELANA) has been developed in this work in Visual Basic Computer programming language to carry out this analysis.

Theory and Model

The Visual Basic computer programming Language is a graphical-based language with its Integrated Development Environment (IDE) allowing Programmers to create, run and debug programs conveniently without being a Windows programming expert (Bradley and Millspaugh, 2002; Halvorson, 2005 and Balena, 2005). It eliminates the need for the programmer to write codes that generates the Form, codes for all the Form's properties, codes for Form placement on the screen, codes to create foreground and background colours and so on. Its event-driven programming, that is codes that responds to events or notifications, allows the user to dictate the order of programming (Deitel et al, 1999). The implementation of this program by this language follows the standard procedures used in any software development: Algorithm design, Program Composition, Debugging and Testing and Storage and Maintenance (Chapra and Canale, 1998).

The Algorithm has been designed as follows:

Step 1: Input values for T (ms) axis and Z (m) axis respectively.

Step 2: Plot the points scatter for T (ms) on the vertical axis against Z (m) on the horizontal axis.

Step 3: Identify the number of layers (lines on graph) plotted.

Step 4: Draw the lines of best fit for T-Z graph

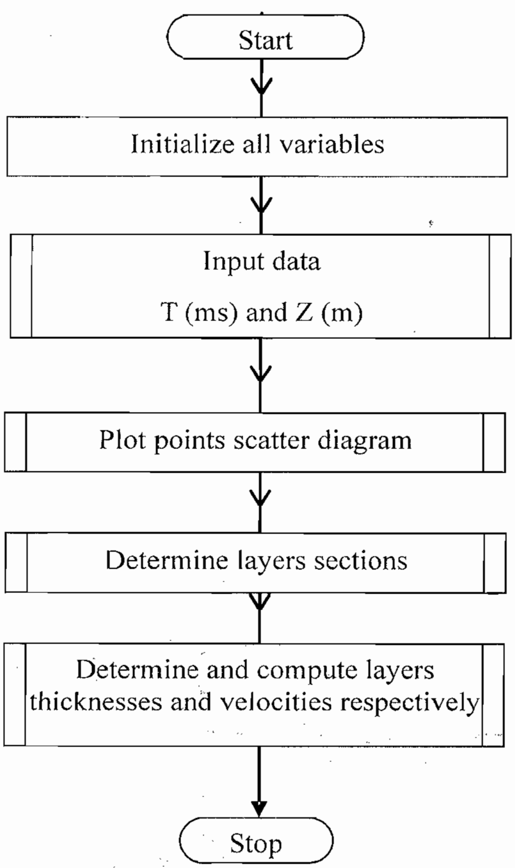

Fig. 1: Main flowchart

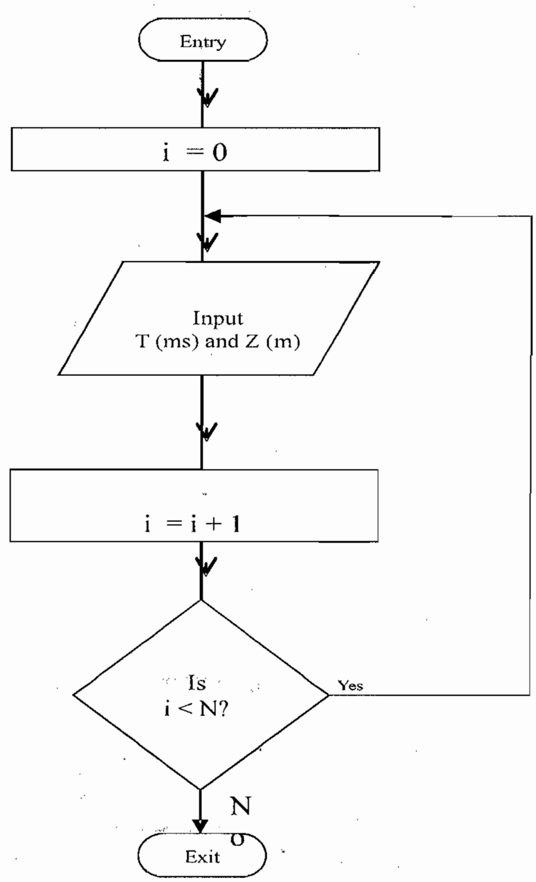

Fig. 2: Input data routine flowchart

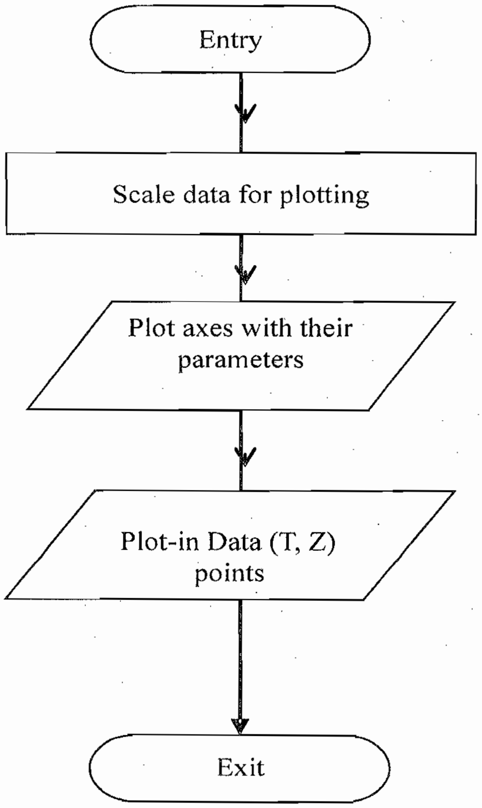

Fig. 3: Plot points scatter diagram routine flowchart

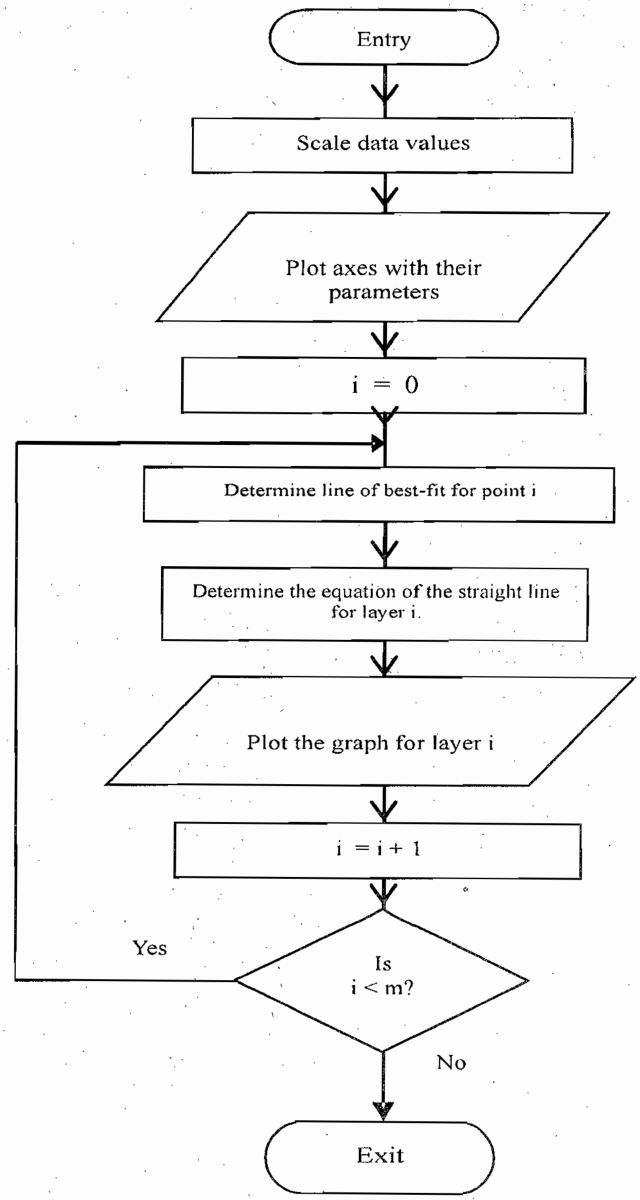

Fig. 4: Plot T-Z graph sub-routine flowchart

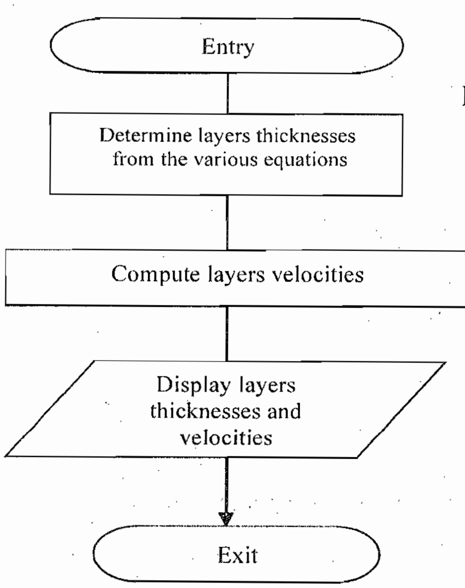

Fig. 5: Sub-routine flowchart to display results

Step 5: Determine the layers thicknesses (depths) and compute layers velocities.
Step 6: Display T-Z plot, layers thicknesses and velocities.
Step 7: Stop.
The flow charts for this Algorithm are shown in Figures 1, 2, 3, 4, and 5.

This program is based mainly on the theory of the "Up-hole" seismic method and its application to the near-surface velocity analysis. In this method, a deep hole of about 60 m or more is drilled and an Up-hole tool, 24-channel geophone cable, is placed in the hole with the first few receivers from the ground surface at closer intervals to ensure that the velocity of the weathering layer was adequately recorded for accurate velocity computations and thickness determination (Figure 6). An energy source (dynamite) placed in a hole at a depth of 1.5 m at an offset distance of 5.0 m from the well is detonated near the ground surface. A seismograph on the surface measures the arrival times of the

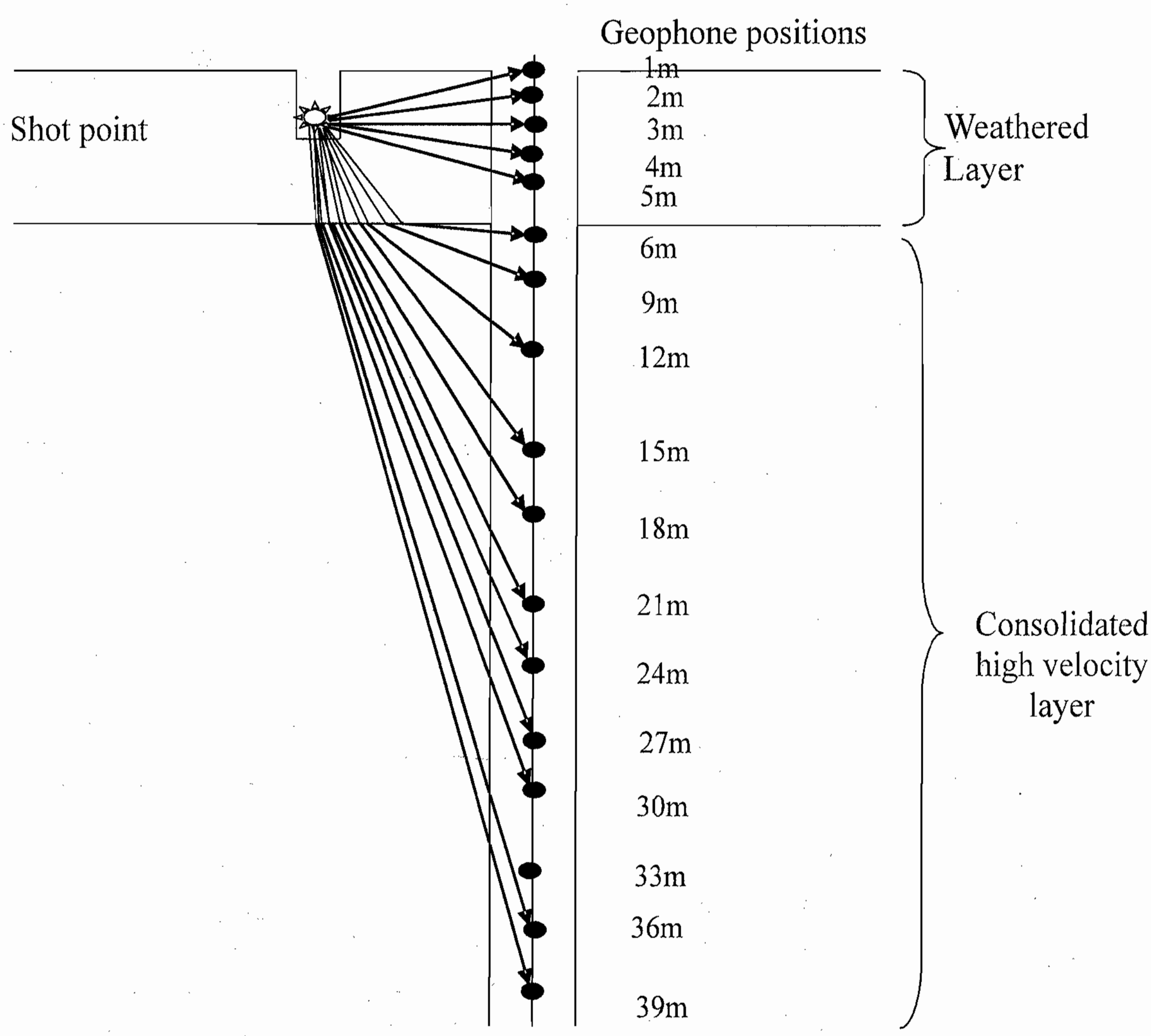

Fig. 6: Theoretical ray paths for up-hole seismic method

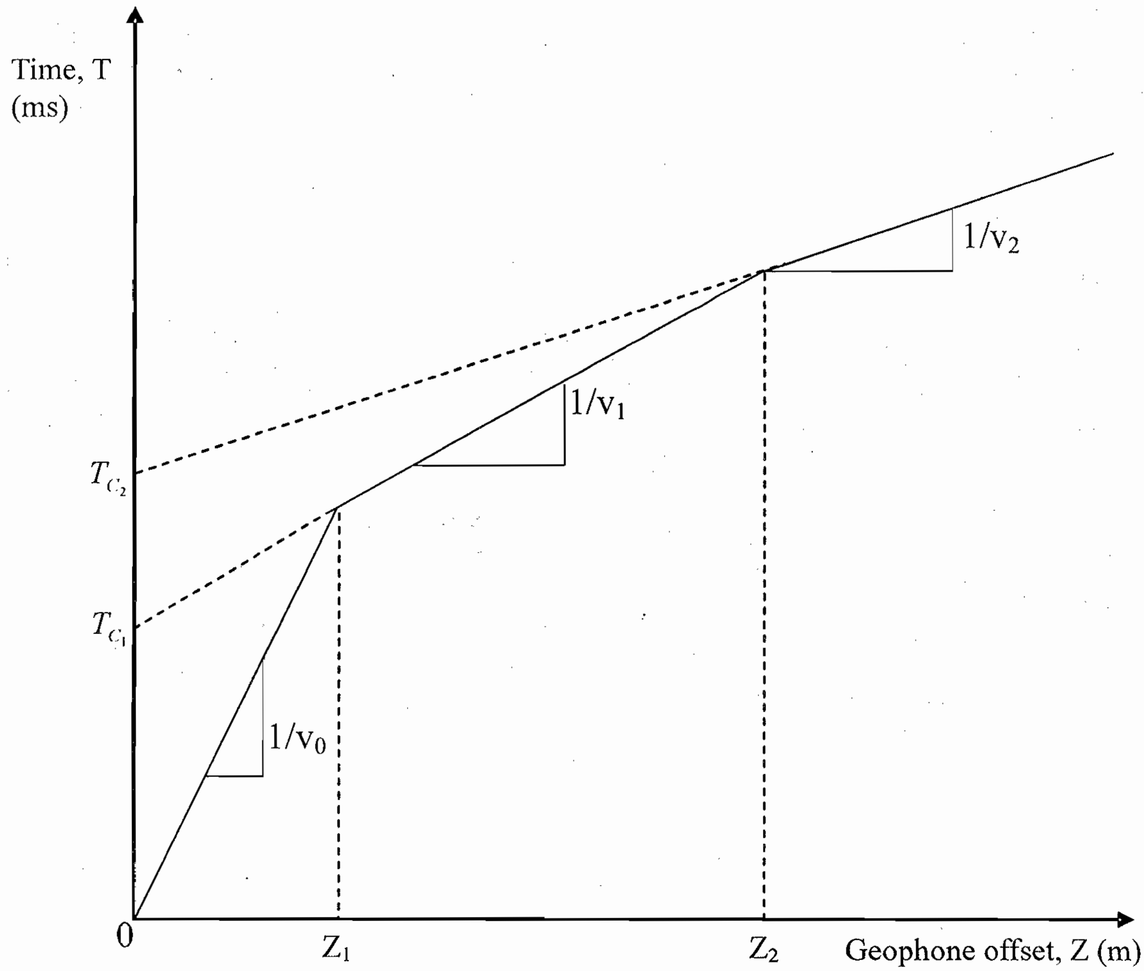

Fig. 7: Typical time depth graph for a weathered three-layer case model

generated sound energy to the detectors down the hole. These arrival times already corrected for the slant travel and source paths are then plotted against the geophone offsets (depths) according to equations 1, 2, 3, and 4 and (Figure 7).

The Time-Depth plot is mathematically given as:

$$T_n = \left(\frac{1}{V_i}\right) Z_n + T_{c_i} \quad (1)$$

The best lines were correctly fitted employing the Least-squares approximation (Spiegel, 1992) for slope and intercept.

The slopes of the various layers were computed using the formula:

$$Slope = \frac{\Delta T_i}{\Delta Z_i} (s/m) \quad (2)$$

The reciprocals of the slopes so computed gave the velocities of the various layers (Slotnick, 1959):

$$V_i = \frac{1}{\text{Slope}} = \frac{\Delta Z_i}{\Delta T_i} (m/s) \quad (3)$$

The thicknesses were determined by automatically solving the equations of the various lines simultaneously at the point where any two lines meet as given by the equation:

$$Z_m = \frac{T_{(i+1)} - T_{c_i}}{\left(\frac{1}{v_i} - \frac{1}{v_{(i+1)}}\right)} \quad (4)$$

where

$\left(\frac{1}{v_i}\right)$ = slopes of the various lines intercepts on the T-axis,
i = 0, 1, 2..., n = 1, 2, 3..., m = 1, 2, 3...
Z = geophone offset in meters (m)
T = arrival time in millisecond (ms).

The model assumes a flat layer, an isotropic and homogeneous medium and an increase of velocity with depth. The formulae have also assumed straight ray paths for the rays within the LVL. It has also been constrained that all the input data to the program must all be positive values, which implies that negative values are ignored. All values less than that of the previous entry should be ignored and that all axes must start from zero, that is, the Time (T) Depth (Z) curve must start from (0,0). The program codes in Visual Basic Programming language are listed in the Appendix.

Results

The process of data input and interpretation is completed within a few minutes of implementation and the results displayed. Once supplied with the input data (Tables 1 and 2), the program, NESURVELANA, automatically generates the layers thicknesses and velocities. Figures 8 and 9 show the results of the application of the program

Add or Modify Input Data for Dept (X-axis) and Time ...

ZAxis	TAxis
1	2.2
6	13.8
9	15.2
12	17.1
15	18.8
21	21.7
24	22.3
27	25.2
30	27.1
33	28.3
39	32.1
45	35.3
48	37.1
51	38.6
54	40.1
57	42

Add | Edit | Delete | Refresh | Close

Record: 1

Table 1: Input data for a weathered two-layer case

Add or Modify Input Data for Dept (X-axis) and Time ...

ZAxis	TAxis
1	2.1
3	6
4	8.2
7	11.4
10	14.1
13	15.7
19	19.2
22	20.8
25	22.9
28	25.7
31	26.3
34	27.6
37	29.6
40	31.6
46	34.7
49	36.5
52	37.8
55	39.4
58	40.7

Add | Edit | Delete | Refresh | Close

Record: 1

Table 2: Input data for a weathered three-layer case

to two sets of near-surface seismic velocity data acquired across the South-Central Niger Delta.

For the weathered two-layer case, the thickness obtained was 6.0 m and velocity of 431 m/s and a consolidated layer velocity of 1845 m/s while for the weathered three-layer case; the

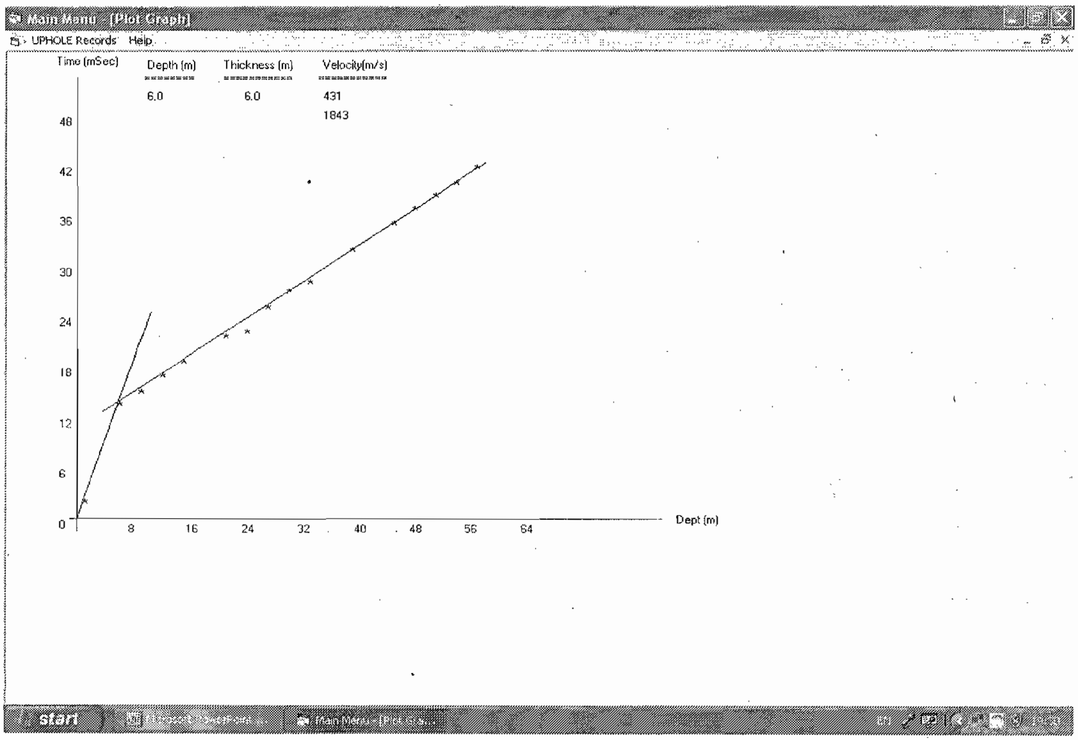

Fig. 8: Results for a weathered two-layer case

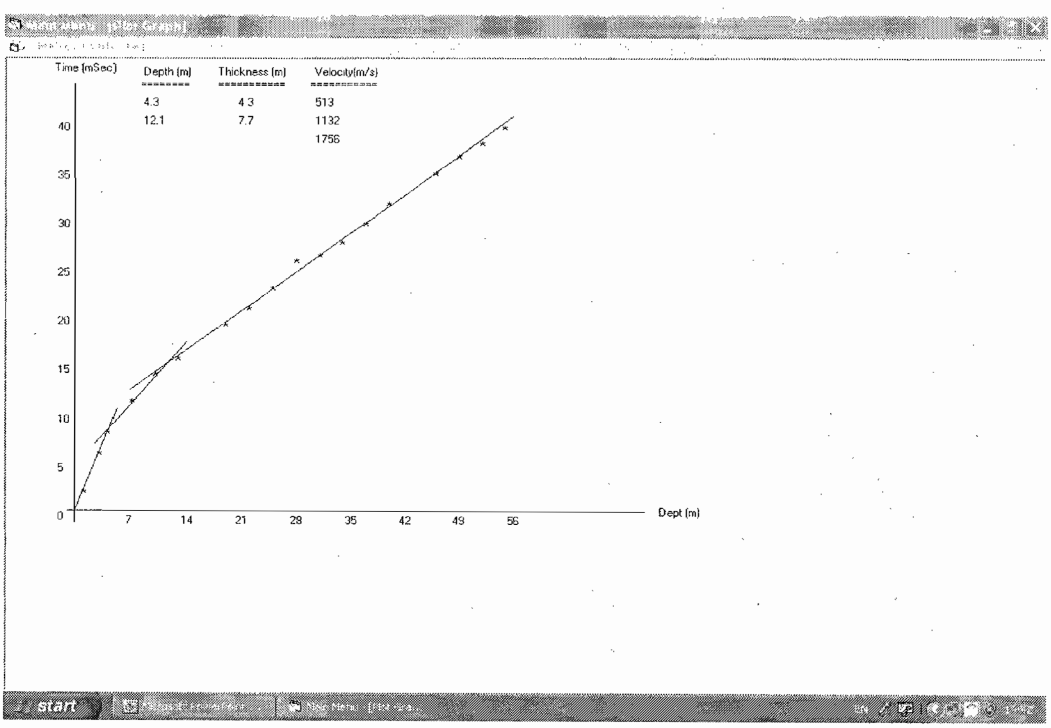

Fig. 9: Results for a weathered three-layer case

thicknesses are 4.3 m and 7.7 m with velocities of 513 m/s and 1132 m/s respectively and a consolidated layer velocity of 1756 m/s. These results represent a test-run of the program on a Microsoft Windows XP. It can also run on MS Windows 98, Windows 2000, Windows Me, Windows XP and future versions.

Discussion and Conclusion

The Computer program, NESURVELANA, has been presented for easy, fast and accurate analysis of the near-surface velocity data acquired by the Up-hole seismic method in the Niger Delta, Nigeria. It has been simplified to allow for easy understanding and adoption by any earth scientist with minimal programming experience. The basic mathematical equations employed are simple and elegant and the sub-routines in the program codes are devoid of "spaghetti-like" codes or "jumping around" due to indiscriminate branching. By its nature of implementation, NESURVELANA allows the user to be in control of the interpretation. It is flexible, that is, user-interactive and does not restrict the user on determining the expected results.

In conclusion, the easy and quick nature of implementation and the closeness of the results to those of other researchers on the weathered layer in the Niger Delta indicate that the program is a reliable measure for the analysis of the near-surface seismic data acquired in the Niger Delta, Nigeria.

References

Akpabio, I. O. and Onwusiri, H. N. (2004): The Computation of LVL correction parameters in parts of western Niger Delta (OML 62): Global Journal of Geological Sciences, 2 (1), 45 50

Balena, F. (2005): Programming Microsoft Visual Basic 2005: The Language, 2nd New Edition, Microsoft Press, U.S.A.

Bradley, J. C. and Millspaugh, A. C. (2002): Programming in Visual Basic 6.0. Update Edition, McGraw-Hill Companies, New York.

Chapra, S. C. and.Canale, R. P. (1998): Numerical methods for Engineers: with programming and software Applications. 3rd Edition, McGraw-Hill Companies, London.

Cohoon, J. P. and Davidson, J. W. (1999): C++ Design: An Introduction to Programming and Object-Oriented Design, 2nd Edition, McGraw-Hill Companies, London.

Deitel, H. M., Deitel, P. J. and Nieto, J. R. (1999): Visual Basic 6, How to Program. Prentice Hall, Upper Saddle River, New Jersey.

Eze, C. L., Okwueze, E. E., and Uko, E. D. (2003): The Velocity Thickness Characteristics of the Mangrove Swamp Low Velocity Layer (LVL) South Central Niger Delta, Nigeria. Global Journal of Pure and Applied Sciences, 9, (3), 369-374.

Halvorson, M. (2005): Microsoft Visual Basic 2005 Step by Step (Step by Step (Microsoft)), Microsoft Press, U.S.A.

Okwueze, E. E., and A. O. Offong (1992): Seismic Refraction Investigation of the Foundation Conditions of the Rock/Soil in Parts of Akwa-Ibom State of Nigeria, Nig., Journ. of Physics, 4, 114-124.

Slotnick, M. M. (1959): Lessons in Seismic Computing. The Society of Exploration Geophysicists, Tulsa, Oklahoma.

Spiegel, M. R. (1992): Theory and Problems of Statistics, 2nd Revised Ed. S.I.Ed. (Schaum's Outline Series), McGraw-Hill Book Co., London.

Uko, E. D., Ekine, A. S., Ebeniro, J. O., and Ofoegbu. C. O. (1992): Weathering Structure of the East Central Niger Delta, Nigeria, Geophysics, 57 (9): 1228-1233.

Appendix
(Visual Basic Program Codes)

Codes for Plotting the Time-Depth Graph

```
Private Sub Inputpmt_Click()
 frmPlotValue.Show
End Sub

Private Sub MDIForm_Load()
'Dim db As Connection
 Set db = New Connection
 db.CursorLocation = adUseClient
                              db.Open
      "PROVIDER=Microsoft.Jet.OLEDB.3.51;Data
      Source=C:\NESURVELANA\TZData.mdb;"

 Set dataValue = New Recordset
  dataValue.Open "select [ZAxis],[TAxis]
      from PlotValue", db, adOpenStatic,
      adLockOptimistic

End Sub

Private Sub mnuexit_Click()
 End
End Sub
Private Sub mnuPlot_Click()
'frmPlotGraph.Show
Dim x1 As Integer
Dim x2 As Integer
Dim y1 As Integer
Dim y2 As Integer
Dim i As Integer
Dim Ymax As Double, Ymin As Double
Dim Xmax As Double, Xmin As Double
Dim X As Integer, Y As Integer
Dim T(200), D(200), slope(10),
      intercept(10), ddx(10)
flag = 0
If (plotwinflag) Then
 Unload frmPlotGraph
 frmPlotGraph.Show
End If
'A = Array(0, 2.3, 13, 15, 16, 18, 20, 53)
'B = Array(0, 14, 15.2, 17, 19, 23, 35, 85)
' Get Maximun and Minimum values
'n = 7
On Error GoTo DataErr
 With dataValue
  .MoveFirst
  While Not dataValue.EOF
                A (i) =
      dataValue.Fields("Zaxis").Value
                B (i) =
      dataValue.Fields("Taxis").Value
  .MoveNext
  i = i + 1
  Wend
  n = dataValue.RecordCount - 1
 End With
A(n + 1) = A(n)
Ymax = B(0): Ymin = B(0)
Xmax = A(0): Xmin = A(0)
For i = 0 To n
 If Ymax < B(i) Then
  Ymax = B(i)
 End If
 If Ymin > B(i) Then
  Ymin = B(i)
 End If
 If Xmax < A(i) Then
  Xmax = A(i)
 End If
 If Xmin > A(i) Then
  Xmin = A(i)
 End If
Next i
'frmTest.WindowState = "2-maximized"
'frmPlotGraph.WindowState = 2
'frmPlotGraph.Appearance = 0
y1 = 100: y2 = 7600
x1 = 1000: x2 = 9000
X = 7000
intervalY = Int(Ymax / 8) + 1
intervalX = Int(Xmax / 8) + 1

'Plotter lines
'==============

frmPlotGraph.Line (x1, y1 + 300)-(x1,
      y2)
frmPlotGraph.Line (x1 - 100, y2 - 200)-
      (x2 + 400, y2 - 200)
'(y2 - y1))
'msg = "  "
'Print
```

```
'frmPlotGraph.Line (X, Y)-(X, Y)

'Placement of Parameters
'========================
X = 1000
j = 1
For i = 8 To 0 Step -1
   frmPlotGraph.Line (700, X)-(700, X)
   X = X + 800
   'msg = msg + "      "
   'Print msg;
   frmPlotGraph.Print i * intervalY
   'frmPlotGraph.CurrentX 400
   'frmPlotGraph.CurrentY 100
   frmPlotGraph.Line (X - 100, 7450)-(X -
      100, 7450)
   'X = X + 800
   'msg = msg + "    "
   'frmPlotGraph.Print msg;
    If i <> 0 Then frmPlotGraph.Print j *
      intervalX: j = j + 1
Next i
frmPlotGraph.Line (700, y1 - 50)-(700, y1
      - 50)
frmPlotGraph.Print "Time (Sec)"
frmPlotGraph.Line (x2 + 600, y2 - 300)-
      (x2 + 600, y2 - 300)
frmPlotGraph.Print "Dept (m)"
sumA = 0: sumB = 0: sumAB = 0: sumAA
      = 0
'Check for points availability
If counter > 0 Then
 bpp(counter) = n + 1
Else: bpp(0) = n + 1
End If
j = 0
For m = 0 To counter
'point of best fit analysis
'=================================
For i = 0 + j To bpp(m) - 1
 sumAB = sumAB + A(i) * B(i)
 sumA = sumA + A(i)
 sumB = sumB + B(i)
 sumAA = sumAA + (A(i) ^ 2)
Next i
slope(m) = (bpp(m) * sumAB - sumA *
      sumB) / (bpp(m) * sumAA - sumA ^
      2)
intercept(m) = (sumB * sumAA - sumA *
      sumAB) / (bpp(m) * sumAA -
      sumA ^ 2)
i = 0 + j
intercept(0) = 0
A(i) = 0
B(i) = intercept(m)
stpval = 0
For i = 1 + j To bpp(m)
 stpval = stpval + intervalX
 If stpval > A(bpp(m) - 1) Then
  stpval = A(bpp(m))
 End If
 A(i) = stpval
 B(i) = slope(m) * A(i) + intercept(m)
Next i


'Scale XY values
'=================
For i = 0 + j To bpp(m)
 D(i) = A(i) * 800 / intervalX
 T(i) = B(i) * 800 / intervalY
Next i

'Ploting of Graph
'=================
For i = 0 + j To bpp(m) - 2
   frmPlotGraph.Line (x1 + D(i), y2 - T(i)
      - 200)-(x1 + D(i + 1), y2 - T(i + 1) -
      200)
Next i
'Print Results Analysis
frmPlotGraph.Line (x1 + (3000 * (m +
      1)), y2 - 1500)-(x1 + (3000 * (m +
      1)), y2 - 1500)
frmPlotGraph.Print "Intercept T = ";
      Format(intercept(m), "0.00")
frmPlotGraph.Line (x1 + (3000 * (m +
      1)), y2 - 1000)-(x1 + (3000 * (m +
      1)), y2 - 1000)
frmPlotGraph.Print "Velocity V = ";
      Format((1 / slope(m)), "0.0000")
plotwinflag = 1
'dataValue.Close
j = bpp(m)
Next m
frmPlotGraph.Line (x1 + 1000, y1)-(x1 +
      1000, y1)
```

```
frmPlotGraph.Print "Depth (m)"; Spc(5);
     "Thickness (m)"; Spc(5);
     "Velocity(m/s)"
frmPlotGraph.Print Spc(26);
     "=========="; Spc(5);
     "============"; Spc(4);
     "==============="
For m = 0 To counter - 1
ddx(m) = (intercept(m) - intercept(m + 1)) /
     (slope(m + 1) - slope(m))
frmPlotGraph.Line (x1 + 500, y2 - 1000)-
     (x1 + 500, y2 - 1000)
frmPlotGraph.Print "zm = ";
     Format(ddx(m), "###0.00")
frmPlotGraph.Line (x1 + 1000, y1 + 500)-
     (x1 + 1000, y1 + 500)
frmPlotGraph.Print Format(intercept(m +
     1), "###0.00"); Spc(15);
     Format(ddx(m), "###0.00");
     Spc(12); Format((1# / slope(m + 1)),
     "###0.00")
Next m
Exit Sub
DataErr:
 MsgBox Err.Description
End Sub
Private Sub mnuPlotpt_Click()
 f2value = plotscatter()
 Fvalue = Plotgraph()
End Sub

Private Sub mnuSections_Click()
 frmFitSection.Show
End Sub
Private Sub mnuxy_Click()
Dim x1 As Integer
Dim x2 As Integer
Dim y1 As Integer
Dim y2 As Integer
Dim i As Integer
Dim Ymax As Double, Ymin As Double
Dim Xmax As Double, Xmin As Double
Dim X As Integer, Y As Integer
Dim T(200), D(200)
flag = 1
If (plotwinflag) Then
 Unload frmPlotGraph
 frmPlotGraph.Show
End If
'A = Array(0, 2.3, 13, 15, 16, 18, 20, 53)
'B = Array(0, 14, 15.2, 17, 19, 23, 35, 85)
i = 0
On Error GoTo DataErr
 With dataValue
  .MoveFirst
  While Not dataValue.EOF
                                A ( i ) =
     dataValue.Fields("Zaxis").Value
                                B ( i ) =
     dataValue.Fields("Taxis").Value
  A2(i) = A(i)
  B2(i) = B(i)
  .MoveNext
  i = i + 1
  Wend
  n = dataValue.RecordCount - 1
 End With

' Get Maximun and Minimum values
'n = 7
Ymax = B(0): Ymin = B(0)
Xmax = A(0): Xmin = A(0)
For i = 0 To n
 If Ymax < B(i) Then
  Ymax = B(i)
 End If
 If Ymin > B(i) Then
  Ymin = B(i)
 End If
 If Xmax < A(i) Then
  Xmax = A(i)
 End If
 If Xmin > A(i) Then
  Xmin = A(i)
 End If
Next i
y1 = 100: y2 = 7600
x1 = 1000: x2 = 9000
X = 7000
intervalY = Int(Ymax / 8) + 1
intervalX = Int(Xmax / 8) + 1

'Scale XY values
For i = 0 To n
 D(i) = A(i) * 800 / intervalX
 T(i) = B(i) * 800 / intervalY
Next i

'Plotter Guide
```

```
'==============
frmPlotGraph.Line (x1, y1 + 300)-(x1, y2)
frmPlotGraph.Line (x1 - 100, y2 - 200)-(x2
      + 400, y2 - 200)

'Placement of Parameters
'=======================
X = 1000
j = 1
For i = 8 To 0 Step -1
  frmPlotGraph.Line (700, X)-(700, X)
  X = X + 800
  frmPlotGraph.Print i * intervalY
   frmPlotGraph.Line (X - 100, 7450)-(X -
      100, 7450)
    If i <> 0 Then frmPlotGraph.Print j *
      intervalX: j = j + 1
Next i
frmPlotGraph.Line (700, y1 - 50)-(700, y1
      - 50)
frmPlotGraph.Print "Time (Sec)"
frmPlotGraph.Line (x2 + 600, y2 - 300)-
      (x2 + 600, y2 - 300)
frmPlotGraph.Print "Dept (m)"

'Ploting of Graph Scattered Points
'==============================
      ====
For i = 0 To n
   frmPlotGraph.Line (x1 + D(i) - 20, y2 -
      T(i) - 260)-(x1 + D(i) - 20, y2 - T(i) -
      260)
  frmPlotGraph.FontSize = 13
  frmPlotGraph.Print "*";
  frmPlotGraph.FontSize = 8
  frmPlotGraph.Print A(i); ","; B(i)
Next i
plotwinflag = 1
'dataValue.Close
Exit Sub
DataErr:
 MsgBox Err.Description
End Sub
Private Sub mnuPlot_Click_manual()
  'frmPlotGraph.Show
Dim x1 As Integer
Dim x2 As Integer
Dim y1 As Integer
Dim y2 As Integer
Dim i As Integer
Dim Ymax As Double, Ymin As Double
Dim Xmax As Double, Xmin As Double
Dim X As Integer, Y As Integer
Dim T(200), D(200)
flag = 0
If (plotwinflag) Then
 Unload frmPlotGraph
 frmPlotGraph.Show
End If
'A = Array(0, 2.3, 13, 15, 16, 18, 20, 53)
'B = Array(0, 14, 15.2, 17, 19, 23, 35, 85)

' Get Maximun and Minimum values
'n = 7
On Error GoTo DataErr
 With dataValue
  .MoveFirst
  While Not dataValue.EOF
                        A ( i )      =
      dataValue.Fields("Zaxis").Value
                        B ( i )      =
      dataValue.Fields("Taxis").Value
   .MoveNext
   i = i + 1
  Wend
  n = dataValue.RecordCount - 1
 End With

Ymax = B(0): Ymin = B(0)
Xmax = A(0): Xmin = A(0)
For i = 0 To n
 If Ymax < B(i) Then
  Ymax = B(i)
 End If
 If Ymin > B(i) Then
  Ymin = B(i)
 End If
 If Xmax < A(i) Then
  Xmax = A(i)
 End If
 If Xmin > A(i) Then
  Xmin = A(i)
 End If
Next i
'frmTest.WindowState = "2-maximized"
'frmPlotGraph.WindowState = 2
'frmPlotGraph.Appearance = 0
y1 = 100: y2 = 7600
```

```
x1 = 1000: x2 = 9000
X = 7000
intervalY = Int(Ymax / 8) + 1
intervalX = Int(Xmax / 8) + 1

sumA = 0: sumB = 0: sumAB = 0: sumAA = 0
'Check for points availability
For i = 0 To n
 sumAB = sumAB + A(i) * B(i)
 sumA = sumA + A(i)
 sumB = sumB + B(i)
 sumAA = sumAA + (A(i) ^ 2)
Next i
slope = ((n + 1) * sumAB - sumA * sumB) / ((n + 1) * sumAA - sumA ^ 2)
intercept = (sumB * sumAA - sumA * sumAB) / ((n + 1) * sumAA - sumA ^ 2)
i = 0
A(i) = intercept
B(i) = 0
stpval = 0
For i = 1 To n
 stpval = stpval + intervalX
 A(i) = stpval
 B(i) = slope * A(i) + intercept
 'frmPlotGraph.Print A(i), B(i)
Next i
'frmPlotGraph.Print A(0), B(0), sumAB, sumB, sumA, sumAA
'Scale XY values
For i = 0 To n
 D(i) = A(i) * 800 / intervalX
 T(i) = B(i) * 800 / intervalY
Next i
'Plotter Guide
'==================
frmPlotGraph.Line (x1, y1 + 300)-(x1, y2)
frmPlotGraph.Line (x1 - 100, y2 - 200)-(x2 + 400, y2 - 200)
'(y2 - y1))
'msg = "   "
'Print
'frmPlotGraph.Line (X, Y)-(X, Y)
'Placement of Parameters
'===========================
X = 1000
j = 1
For i = 8 To 0 Step -1
 frmPlotGraph.Line (700, X)-(700, X)
 X = X + 800
 'msg = msg + "     "
 'Print msg;
 frmPlotGraph.Print i * intervalY
 'frmPlotGraph.CurrentX 400
 'frmPlotGraph.CurrentY 100
 frmPlotGraph.Line (X - 100, 7450)-(X - 100, 7450)
 'X = X + 800
 'msg = msg + "     "
 'frmPlotGraph.Print msg;
  If i <> 0 Then frmPlotGraph.Print j * intervalX: j = j + 1
Next i
frmPlotGraph.Line (700, y1 - 50)-(700, y1 - 50)
frmPlotGraph.Print "Time (Sec)"
frmPlotGraph.Line (x2 + 600, y2 - 300)-(x2 + 600, y2 - 300)
frmPlotGraph.Print "Dept (m)"

'Ploting of Graph
'==================
For i = 0 To n - 1
  frmPlotGraph.Line (x1 + D(i), y2 - T(i) - 200)-(x1 + D(i + 1), y2 - T(i + 1) - 200)
  'X = X - 500
  'Y = 2 * X
  'msg = msg + "    "
  'Print msg;
  'Print "*"; X; Y
Next i
'Print Results Analysis
frmPlotGraph.Line (x2, y2 - 1500)-(x2, y2 - 1500)
frmPlotGraph.Print "Intercept T = "; Format(intercept, "0.00")
frmPlotGraph.Line (x2, y2 - 1000)-(x2, y2 - 1000)
frmPlotGraph.Print "Velocity V = "; Format((1 / slope), "0.0000")
plotwinflag = 1
'dataValue.Close
Exit Sub
DataErr:
 MsgBox Err.Description
End Sub
```