To the Graduate Council:

I am submitting herewith a thesis written by Alan Tackie entitled "An Automated Approach to Analyzing Media Contribution to H-Piles Driven into Compacted Fills." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Civil Engineering.

Dr. Earl E. Ingram, Major Professor

We have read this thesis
and recommend its acceptance:

Dr. Edwin G. Burdette

Dr. David W. Goodpasture

Accepted for the Council:

Vice Provost and Dean of Graduate Studies

AN AUTOMATED APPROACH TO ANALYZING MEDIA CONTRIBUTION TO H-PILES DRIVEN INTO COMPACTED FILLS

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Alan D. Tackie
May 2004

Acknowledgements

I would like to express my gratitude to the members of my committee for their support during this endeavor. I am especially thankful to my major professor, Dr. Earl Ingram, for his guidance, constructive criticism, and dedication to my welfare. I am indebted to Dr. Edwin G. Burdette whose guidance and utmost kindness made it possible for me to attend the University of Tennessee. Like a parent away from home, he indeed made my transition from Missouri to Tennessee a delight. I am exceedingly grateful. I am a better person now because of the education I received from Dr. David Goodpasture, Dr. Richard Bennett, Dr. Eric Drumm and my colleagues in the structural engineering department.

I would also like to thank Dr. George Rothenbach whose financial support and confidence in me are immeasurable. My sincerest gratitude also goes to Dr. Roger Parsons for allowing me to work with the Engage Program. I am eternally grateful.

Finally, I would like to thank my parents, siblings and their families for supporting me and praying for me.

Abstract

The purpose of this study is to establish the contribution of a soil medium to the behavior of a laterally displaced pile in a displacement-controlled system. Engineers have often assumed that piles driven into well-compacted fills have a lateral response that correlates closely to those driven into undisturbed soils. An investigation into whether this assertion holds throughout the media cross-section forms the crux of this thesis. HPILE, a pile analysis program, was developed by the author of this thesis to foster a quicker analysis of a laterally loaded pile driven into a given media. HPILE uses strain data collected along the length of a laterally displaced pile under various degrees of static lateral loading to develop soil p-y curves by solving the regulating differential equation. The significant contribution in lateral resistance by a medium led to results that will influence design considerations and subsequently discount the assumption regarding behavioral similarity.

Table of Contents

List of Figures

List of Plates

Chapter One: Introduction

1.1 Background

The behavior of laterally displaced steel H-piles driven into compacted fills supporting integral abutments has been a subject of interest to the Tennessee Department of Transportation (TDOT). When thermal expansion in a bridge supported by integral abutments occurs, the supporting piles experience substantial lateral displacement. The massive longitudinal stiffness of the superstructure overwhelms the moderate lateral bending stiffness of the pile and the resistance of the supporting medium and therefore results in essentially unrestrained movement. Subsequent stresses induced in the supporting piles may be pronounced or insignificant depending on the level of resistance offered by the surrounding media. An investigation of how this displacement propagates throughout the media is performed in this thesis.

When TDOT sponsored research conducted by the University of Tennessee on pile response to applied lateral deflections, two types of media were investigated. In the first, piles were driven and individually tested in undisturbed soil, and similar tests were performed on individual piles in the second compacted fill medium. The piles were part of a study performed on H-piles supporting simulated integral abutments. In a dissertation presented by Ingram (2002), software was developed in Matlab for the sole purpose of analyzing a third media type which was comprised partly of undisturbed soil layered with a compacted fill. A robust application was developed for this investigation around Ingram's software to enable an accelerated analysis of the various data sets.

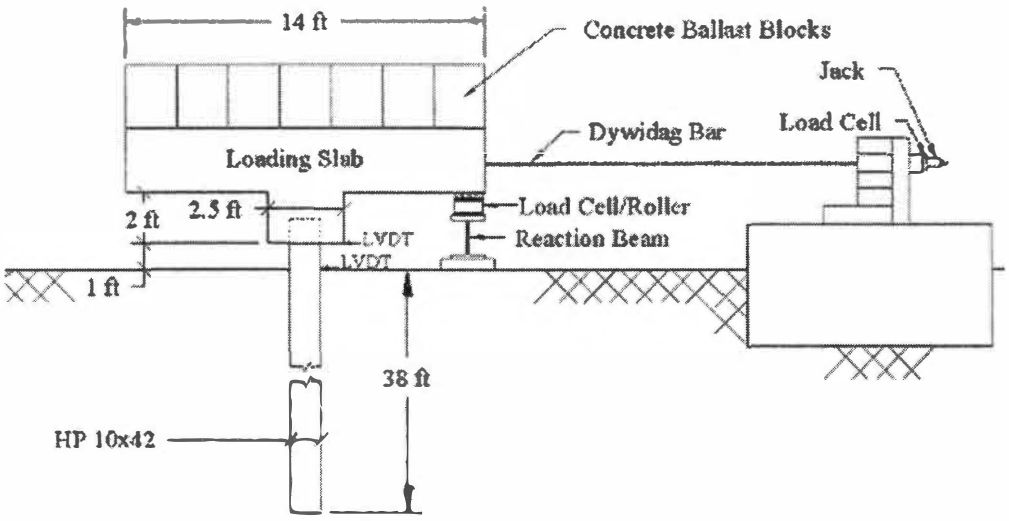

FIGURE 1: Field Test Setup (not to scale)

Field data acquired from the piles were obtained via strain gages placed on the inside faces of each flange projection at various elevations on each pile. Rotational restraint at the top of the pile was consistent with integral abutments (Ingram, 2002). Figure 1 details the test setup.

In accordance with current TDOT expansion limitations for jointless bridges, a maximum lateral displacement of 1 inch was induced such that bending occurred about the strong axis (Wasserman & Walker, 1996). Three different ½-inch deflection tests were followed by three separate 1-inch deflection tests in the same direction as the preceding ½-inch tests. Completion of the six loading sequences was followed by repeated tests in the opposite direction. The twelve deflection tests were performed on twelve separate days. Moment versus depth for the upper 20 feet of pile, horizontal

shear, and lateral displacement of the abutment were acquired for each test by means of strain gages, linear variable differential transformers (LVDT), load cells, and pressure sensors (Ingram, 2002). A comprehensive report detailing field tests and data acquisition is presented in Behavior of Laterally Loaded Piles Supporting Bridge Abutments (Burdette et al., 1999).

1.2 Scope

The primary objective of this thesis was to devise a means of quickly analyzing the data collected and render judgment on the soil's contribution to the behavior of the investigated piles. The approach presented in Ingram's dissertation was thorough but not appropriate for analyzing the voluminous data available on the piles tested. The initial task of this thesis was to provide an automated solution to the pile analysis process, and finally detail the effect of each media on the pile. The belief that media type has little bearing on behavior was investigated by comparing the relative performance of the piles driven into compacted fills to those driven into undisturbed soil. The output of each analysis performed included moment, deflection, shear, soil reaction, and, most importantly p-y, curves.

Chapter Two: Literature Review

A number of computer applications have been developed to analyze piles with most tailored toward investigating specific tasks such as the determination of depth of zero-shear. It is interesting to note that most of the analytical applications developed were based on the work of Lymon C. Reese whose dedication to the study of the beam-column has won the admiration of the engineering community. Roy H. Borden and Robert S. Lawter Jr. measured the load-deflection of piles in sand using the Marchetti Dilatometer (DMT) (Borden & Lawter, 1989). They prescribed minimum segment lengths of 8 inches while using Reese's recommended finite difference solution.

The HPILE application described in this thesis provides variable minimum segment lengths of 2.0, 4.5, and 8.0 inches. The smaller segment lengths were consistent with Ingram's segment choice and also allowed for a better and more refined approximation and resolution of the finite difference solutions. The choice of 8 inches was included to enable any future comparisons of solutions from HPILE to those from DMT. Ingram's dissertation on the behavior of H-piles driven partially into undisturbed soil and partially into compacted fills formed the basis for this thesis. He also advocated Reese's finite difference approach. This approach solved the following differential equation for a beam-column derived by Hetenyi (Hetenyi 1946).

$$E_p I_p \frac{d^4 y}{dy^4} + P_x \frac{d^2 y}{dy^2} - p = 0$$

Where: E_p = Modulus of elasticity of the pile material

I_p = Moment of inertial of the pile in the direction of bending

P_x = Axial load on the pile

p = Soil reaction per unit length of pile

Reese's approach to the differential equation is in the form of the finite difference equation presented below.

$$y_{m-2} R_{m-1} + y_{m-1}(-2R_{m-1} - 2R_m + P_x h^2) + y_m (R_{m-1} + 4R_m + R_{m+1} - 2P_x h^2) + p_m h^4$$

$$+ y_{m+1}(-2R_m - 2R_{m+1} + P_x h^2) + y_{m+2} R_{m+1} = 0 \qquad \text{(Reese, 1997)}$$

Other forms of the differential equation include the following

$$(R_{m-1} / h^2)(y_{m-1} - 2 y_m + y_{m+1}) = M_m$$

$$(R_m / 2h^3)(y_{m-2} - 2 y_{m-1} + 2y_{m+1} + y_{m+2}) = V_m$$

Where: y_m = Deflection at node m

$R_m = E_p I_p$ at node m

P_x = Axial load on the pile

h = Distance between nodes

p_m = Soil reaction per unit length of pile at node m

M_m = Moment at node m

V_m = Shear at node m

While field data provides boundary conditions of moment and shear at the top of the pile, conditions for moment and deflection at the tip of the pile were assumed zero. A much thorough explanation of how Reese's equation was developed is presented in Single Piles and Pile Groups under Lateral Loading (Reese, 1997). Further understanding on the applicability of Reese's solution to piles can be found in Ingram's dissertation, Behavior of Piles Supporting Integral Abutments (Ingram, 2002).

Another similarity among competing pile analysis applications was the use of the Winkler model. This model, used by Ingram as well as by Borden and Lawter Jr.,

idealized the lateral stiffness of the soil as a series of independent nonlinear discrete springs. Using the computer program LTBASE, Borden and Lawter Jr. generated p-y curves at 8-inch intervals along the length of the pile (Borden & Lawter, 1989). The yardstick for measuring the accuracy of the generated p-y curves was how close the computer output came to the DMT measurements. Unlike Borden's data, the data used herein were collected by means of strain gages and therefore were not directly applicable to the use of the LTBASE computer program.

Determining the lateral loads on piles can be a complicated process. The Simple Approach for Lateral Loads on Piles (SALLOP) was developed at Texas A&M University to obtain p-y curves directly from a pressuremeter curve (Briaud, 1997). Again, Reese's work on the beam-column was used as the basis for analysis. Jean-Louis Briaud adopted SALLOP as the tool for establishing the depth of zero-shear (Briaud, 1997).

The use of strain gages as a means of collecting data from piles subjected to stresses warrants applications designed specifically to analyze such strain data. Although other pile analysis programs such as LPILE and COM624P could be used to perform the pile analysis, they require extensive soil data to develop approximate p-y curves from the input. Very few applications provide the most promising method for analysis (Wasserman & Walker, 1996). The development of HPILE was necessary in expediting the analytical process. The ability to reduce the segment lengths and, more importantly, to quickly generate p-y curves for the extremely large amount of data lent credibility to the need for HPILE.

Chapter Three: The Automation Process

3.1 Understanding the Process

The development of HPILE was constrained by a legacy application program written to analyze the media type presented in Ingram's dissertation. The language of choice was Matlab. Matlab has an elegant approach to solving matrices of massive proportions, a feature pertinent to the finite difference solution. A brief synopsis of how Ingram's program operated is presented herein.

3.1.1 The Original Program

Strain data collected from single pile tests were converted to moment values at discrete points along the depth of the pile. The collected moment versus depth and applied lateral load data were then stored in Microsoft Excel files. The files served as preliminary boundary data for solving the finite difference equation. A regression analysis was performed from the preliminary boundary data. Moment values at 4.5-inch depth increments were then evaluated through the function describing the coefficients from the regression analysis. Text files describing moment present in the pile for specific incidents of abutment displacement were saved to serve as final boundary data for the finite difference equation. The text file name was physically inserted into the Matlab source program and eventually executed.

The user at a given prompt needed to know how many segments there were in the saved text file, the modulus of elasticity, and the section modulus of the driven pile. The output yielded moment, shear, deflection, and soil reaction. Graphical plots of the results required additional processes outside the capability of the program. Generation of p-y

curves required individual painstaking analysis of a series of text files corresponding to increments in pile lateral displacement at the abutment level.

3.1.2 Approach to Improvement

An automation of the process performed by Ingram's program required the elimination of those intermediate human intervening activities needed when using Ingram's program. An improvement on the current process required the retrieval of the preliminary boundary data from Excel, performance of regression analysis, generation of the number of analyzed segments, generation of graphs, and performance of a p-y analysis without compromising field data or interfering with the program's source code. Benefits of such actions were the drastic reduction in user error, an increase in analytical speed, and an easy comparison of the piles driven into the different media, which was the ultimate goal.

3.2 New System Recommendation

In order to maintain the integrity of the collected field data, the user's interaction with the Excel spreadsheets was to be avoided and a standardization of the spreadsheets to allow easy data retrieval was recommended. An encapsulation of the source program through the use of a graphical user interface (GUI) was proposed to facilitate an easier system use. Figures A6 and A7 in Appendix A illustrate the GUI menus. Intrinsic information such as number of segments, modulus of elasticity, and section modulus were to form an integral part of the system. The capability to vary segment lengths was

also to be made possible. Finally, the ability to perform p-y analyses and generate graphical output was a must in the new system.

3.3 System Development

A prerequisite to system development was the standardization and arrangement of Excel files. The data were not tampered with, but the sheet tabs were relabeled to allow for an appropriate user choice from the GUI. A preliminary GUI design was performed with an efficient layout of GUI controls. Fields were provided for the user to choose the pile, test date, the Excel file for that test date, and a specific test sheet from the chosen date. The provision of the GUI fields was done to accommodate the different tests performed on a single pile.

The user now has the liberty to choose from a list of saved pile data. A GUI control allows the user to select from a range of pile segment lengths. Default values for the modulus of elasticity and section modulus are provided but can be changed at the user's discretion. The user may elect to perform a p-y analysis by selecting the appropriate choice from a GUI control. A6 in appendix A illustrates all the GUI controls mentioned within this document. An *Analyze* button is provided in the GUI interface to perform the analysis on the selected pile. Appendix C provides the source code for all processes resulting from an execution of the *Analyze* button.

3.4 Performing an Analysis

The user's input from the *Analysis Menu*, the primary user's graphical interface, is first read and assigned to variables. A string concatenation is performed to allow HPILE

to access Excel and import the preliminary boundary data. A regression analysis is then performed to obtain the coefficients of the regression function. The regression function, together with an array of segments generated from the user's choice of segment length, is used to create an array of moment boundary data. The moment array forms the input for the finite difference equation presented by Reese. A coefficient matrix described in Ingram's dissertation and the moment array boundary data are then used in solving the finite difference equation. The resulting matrix solution holds values for the shear, soil reaction, and deflection at equidistant segment lengths spanning the upper 20 feet of the pile. A separation of solution is performed and arrays corresponding to shear, soil reaction, and deflection are generated. These are then saved as ASCII text files for later recall by the graphical display feature of HPILE. Physically scripting the file name into the program's source code is eliminated since the generation of all ASCII text files is made internal to HPILE. The issue of error checking is addressed in a later section.

A user's choice of performing p-y analysis follows an iterative process in which the above analysis is performed several times until all the Excel sheets corresponding to increments in displacements at the abutment level are analyzed. The difference between a regular pile analysis and one involving p-y investigation is the significant number of ASCII text files evaluated. The compiled results are used in producing p-y curves at three specific depths; these depths are 18, 36, and 54 inches below the ground surface.

3.5 Generating Results

Instructions for installing and using the HPILE program are provided in Appendix B. An execution of the *Analyze* button results in a screen display of the HPLOT graphical

user interface. The finite difference solutions saved as ASCII text files are now made available to the HPLOT menu. Figure A7 in appendix A illustrates the HPLOT menu. A drop down menu allows the user to choose from a list of functions. These functions include Moment, Shear, Deflection, Soil Reaction and P-Y plot choices. A feature allowing for a quick review of the plotted points is available in this window. The user also has the capability of producing plots via a printer.

3.6 Error Checking

A series of error checks are performed to maintain system integrity. The checks performed are to ensure that user input and data imported from Excel meet the requirements necessary for a successful pile analysis. Opened Excel files are checked to ensure that the spreadsheets are not empty prior to data extraction. An error message is displayed to alert the user if a chosen Excel file is invalid. Since most data transportation is automated, the propensity for error is minimal.

Chapter Four: Analysis and Findings

4.1 Piles and Media

The media types investigated were undisturbed virgin red clay soil and a well-compacted fill of the same red clay material (Burdette et al., 1999). Piles 1 and 2 were driven into the undisturbed media while piles 3 and 4 were tested in the well-compacted fill. As previously mentioned, pile 5 was driven into undisturbed soil layered with a compacted fill. Results of all analyses performed on pile 5 are presented in Ingram's dissertation. The mimicking of thermal expansion at the level of the abutment through a deflection-controlled test assured that deflection at the pile-abutment interface was constant for piles in each type of media. The deflection-controlled test set the tone for evaluating the effect of the supporting medium's resistance to lateral loading. The evaluation included a comparison of the moment due to lateral displacement at the abutment, the resulting soil reaction, and deflected shape of the pile. The designation of piles 1 and 2 as the control piles was based on the work of Meehan whose investigation of the stiffness of media in the tests considered herein revealed that the compacted fill was less stiff than the undisturbed clay soil (Meehan, 1999).

4.2 Load – Deflection Curves and Moment

Figures A1 and A2 show p-y plots for first and second direction loading of piles 1 and 4 respectively. Pile 1 was loaded in undisturbed red clay soil while pile 4 was loaded in a fill. A comparison of each media's resistance to the repeated loading punctuated by periods of sustained loading revealed that for a given depth of 36 inches and a deflection

of 0.5 inches, the undisturbed media had an average resistance of 1250 lbs/inch while the compacted fill averaged 500 lbs/inch.

The nature of the graphs did not reveal any conclusive evidence that the order of loading influenced the relative stiffness of the media in a particular loading direction. It is important to reiterate that the testing of piles was deflection-controlled and that the moisture content of the medium may have played some role in stiffness fluctuations. Under large deflection, which was the case here, the difference in media stiffness was reflected in the amount of resistance offered to the lateral load induced at the level of the integral abutment. Monthly rainfall data indicated that in the first loading direction of pile 1, testing was performed during a relatively wet month thereby causing the stiffness of the media to be reduced. The second loading direction of pile 1 test was performed in a month with less rainfall and hence a slightly higher media stiffness (Meehan, 1999). Meehan, however, conceded that more testing was needed to accurately correlate pile stiffness with monthly rainfall. Figure A2 indicated that the compacted fill's resistance in the second loading direction was higher than in the first. This higher resistance was attributed to the fact that media stiffness of the compacted fill during the second loading direction was higher. Figure A3 shows the deflected shape of the piles under controlled lateral displacement. At the level of the abutment, both piles were displaced 1 inch. Unlike pile 4, pile 1 rapidly approached infinite slope at a near zero deflection of 0.05 inches due to the stiffness of the undisturbed clay soil. The significant difference in deflected shape indicated that the reaction from the supporting media to pile displacement influenced the deflection below the ground surface. The shear span or media depth below ground surface to where soil reaction was zero was greater in the compacted fill as a

result of the low media stiffness. Reducing the deflection within the media to a near zero value was therefore gradual in the compacted fill and not as rapid as observed in the undisturbed clay soil.

A feel for the stiffness of a given medium is meaningless unless this stiffness is referenced to the stiffness of the embedded pile. A stiff pile in a stiff medium may behave in a similar fashion as a weak pile placed in a not so stiff medium, ceteris paribus, all factors remaining constant. Quantifying the relative stiffness of the two media was based on Effect of Boundary Conditions on Buckling of Friction Piles (Gabr et al., 1994). Based on Terzaghi's subgrade-reaction theory, the constant of horizontal subgrade-reaction k_h assumed to vary linearly with depth was determined as follows

$$k_h = \frac{p}{d * y}$$

Where: p = Soil reaction per unit length of pile

d = Depth from the ground surface

y = Lateral deflection of pile at depth d

The relative stiffness or coefficient of pile-soil compliance α was then defined as

$$\alpha = \sqrt[5]{\frac{k_h}{EI}}$$

Where: E = Modulus of elasticity of the pile material

I = Moment of inertia of the pile in the direction of bending.

For the undisturbed clay soil, an average horizontal subgrade-reaction and relative stiffness of 72.8 lb/in.^{3} and 0.026 in.$^{-1}$ respectively were estimated. The average estimates

reached for the compacted fill were 30.8 lb/in^3 and 0.021 in.$^{-1}$ The small change in relative stiffness is quite considerable given that the evaluation of α is based on the fifth root of the ration of k_h to EI. A significant difference in stiffness was needed to effect a little change in α.

Figure A4 shows the moment graphs for the piles in the undisturbed and compacted fills under first and second loading. The deflection controlled nature of tests reflected in comparable moments in both media at the abutment level. The maximum moment below the ground surface in the undisturbed clay soil was however much higher than that of the compacted fill. The significant difference was attributed to the already established difference in media stiffness. Figure A5 shows that the stiffer medium through a shorter effective depth imposed a much higher load on the deflected pile which, as a result, translated to a higher moment. Also indicated from figures A5 and A4 were the lower soil reaction in the compacted fill and greater depth at which the maximum moment occurred. The significant difference in maximum moment at each media's effective depth reaffirmed that the media did contribute to the behavior of the pile below ground level.

4.3 Conclusions and Recommendations

The significant decrease in the compacted fill's resistance to lateral loading under controlled displacement was noteworthy. The undisturbed clay soil had over twice the resistance of the compacted fill, a fact attributed to the higher relative stiffness of the medium compared to the pile. The maximum moment below ground in the undisturbed clay soil was considerably higher than that in the compacted fill, and the depth of

maximum moment was closer to the ground surface in the undisturbed soil. The higher moment was again attributed to a higher soil reaction, a direct result of the stiffness of the medium. The lateral resistance offered by compacted clay fills on pile-abutment structures experiencing considerable thermal expansion are, therefore, not as large as that which can be expected from undisturbed clay soil. Media contribution below ground surface should be accounted for if the relative stiffness of the supporting medium is high or comparable to that of the undisturbed virgin red clay chosen for the tests herein.

Bibliography

Bibliography

Borden, Roy H. and Lawter, Robert S. Jr., Load-Deflection Response of Piles in Sand: Performance Prediction using the DMT, Transportation Response Records N 1235, 1989 p 79-88.

Briaud-Jean-Louis. "SALLOP: Simple Approach for Laterally Loads on Piles," *Journal of Geotechnical and Geo-environmental Engineering*. V 123 n 10 Oct. 1997, p 958-964

Burdette, Edwin G., Deatherage Harold J., and Goodpasture David W., Behavior of Laterally Loaded Piles Supporting Abutments, Final Report, The University of Tennessee Center for Transportation Research, The University of Tennessee Knoxville, December 1999.

Gabr Mohammed A., Wang Jibai, Kiger Sam A., Effect of Boundary Conditions on Buckling of Friction Piles, *Journal of Engineering Mechanics*. V 120 n 6 June 1994, p 1392-1400

Hanselman D. and Littlefield B., The Student Edition of MATLAB, The Language of Technical Computing, The Math Works Inc., Prentice Hall, NJ., 1997.

Hetenyi M., Beams on Elastic Foundation, The University of Michigan Press, Ann Arbor, 1946

Ingram, Earl E., Behavior of Piles Supporting Integral Abutments, A Dissertation Presented for the Doctor of Philosophy Degree, The University of Tennessee Knoxville, August 2002.

Meehan, Kayon, A Comparison of Load-Deflection Curves for Tests on Simulated Abutments," A Special Problem Presented for the Master of Science Degree, The University of Tennessee Knoxville, December 1999

Reese, Lymon C., and Van Impe, William F., Single Piles and Pile Groups Under Lateral Loading. A. A. Balkema, Brookfield. 2001.

Wasserman, Edward P. and Walker, Houston J., "Integral Abutments for Steel Bridges," Highway Structures Design Handbook, Chapter 5, Vol. II, American Iron and Steel Institute, October 1996.

Appendices

Appendix A: Figures

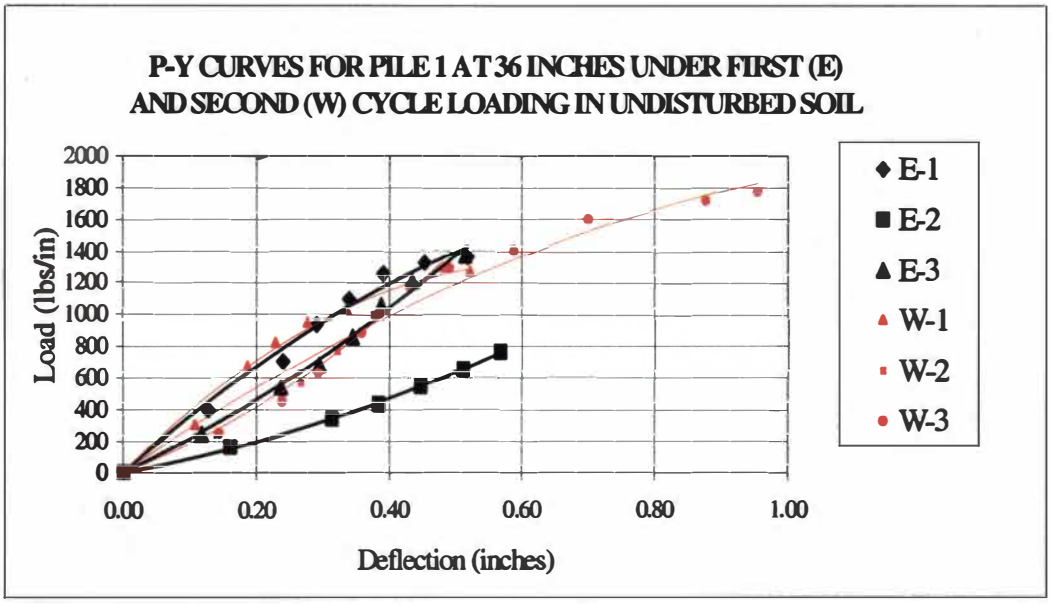

FIGURE A1: P-Y Curves at 36 inches Below Ground Surface (Pile 1)

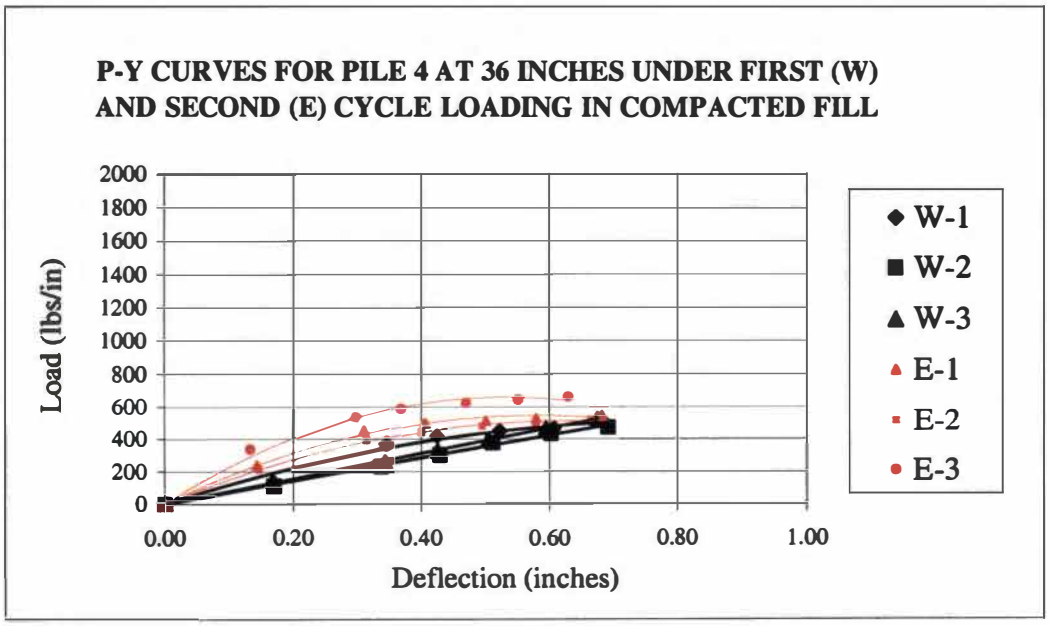

FIGURE A2: P-Y Curves at 36 inches Below Ground Surface (Pile 4)

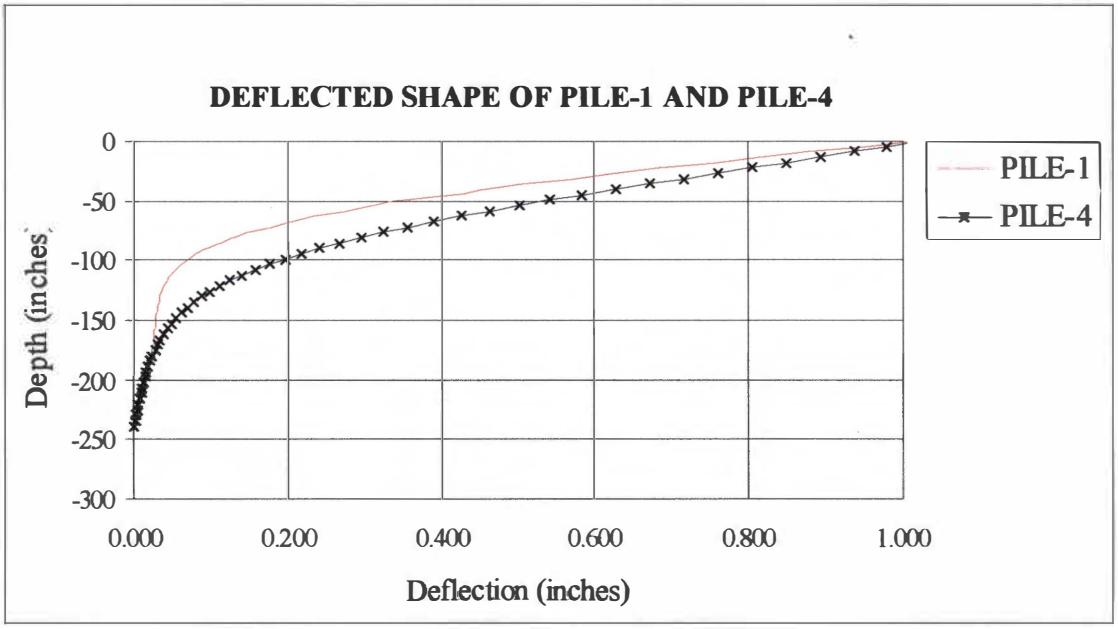

FIGURE A3: Deflected Shape of Piles

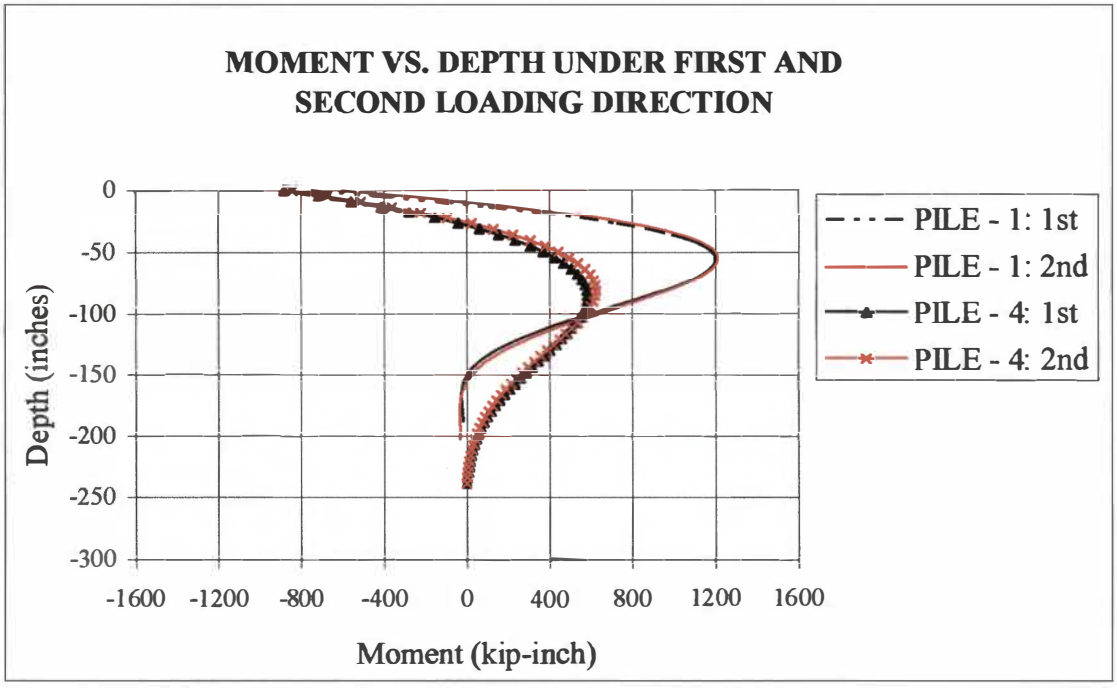

FIGURE A4: Moment Diagrams

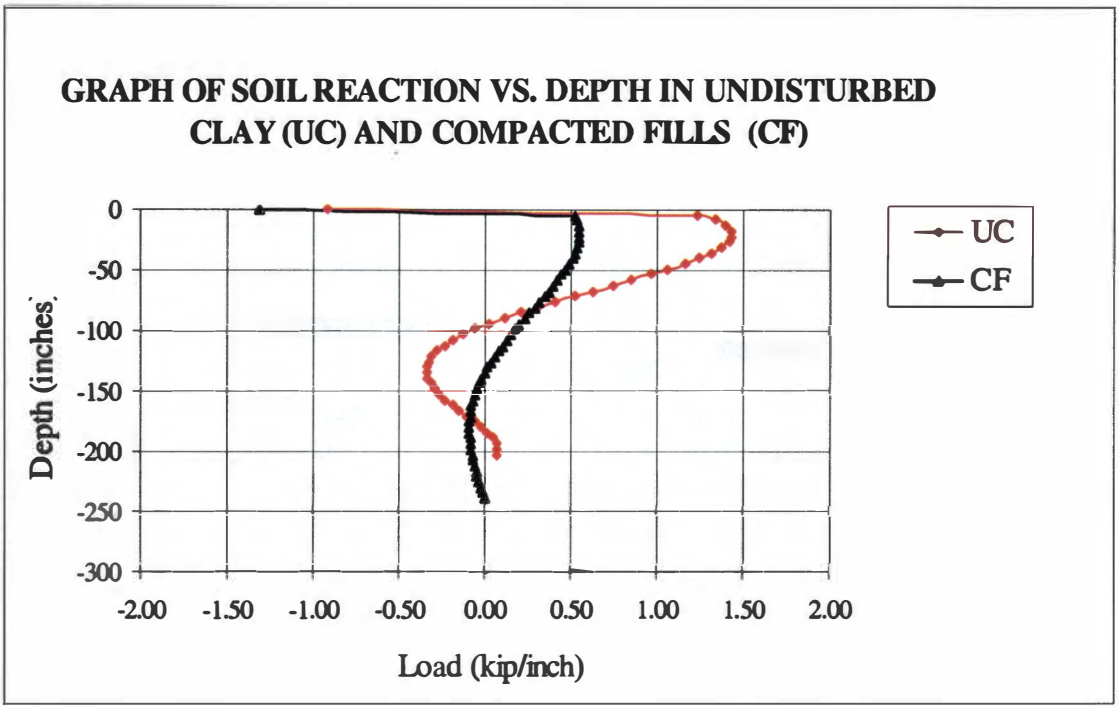

FIGURE A5: Soil Reaction on 1-inch Deflected Pile

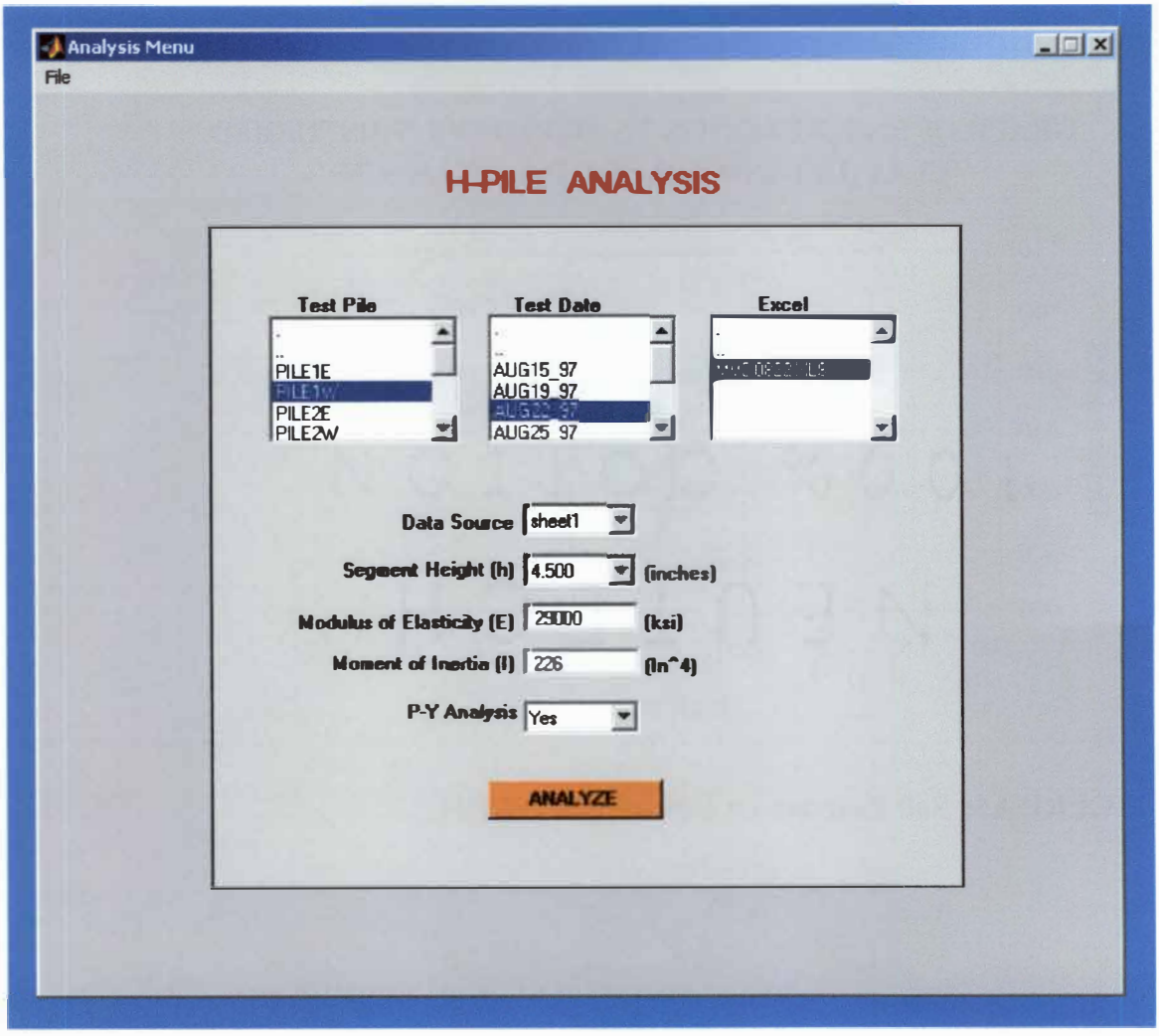

FIGURE A6: HPILE Analysis Menu

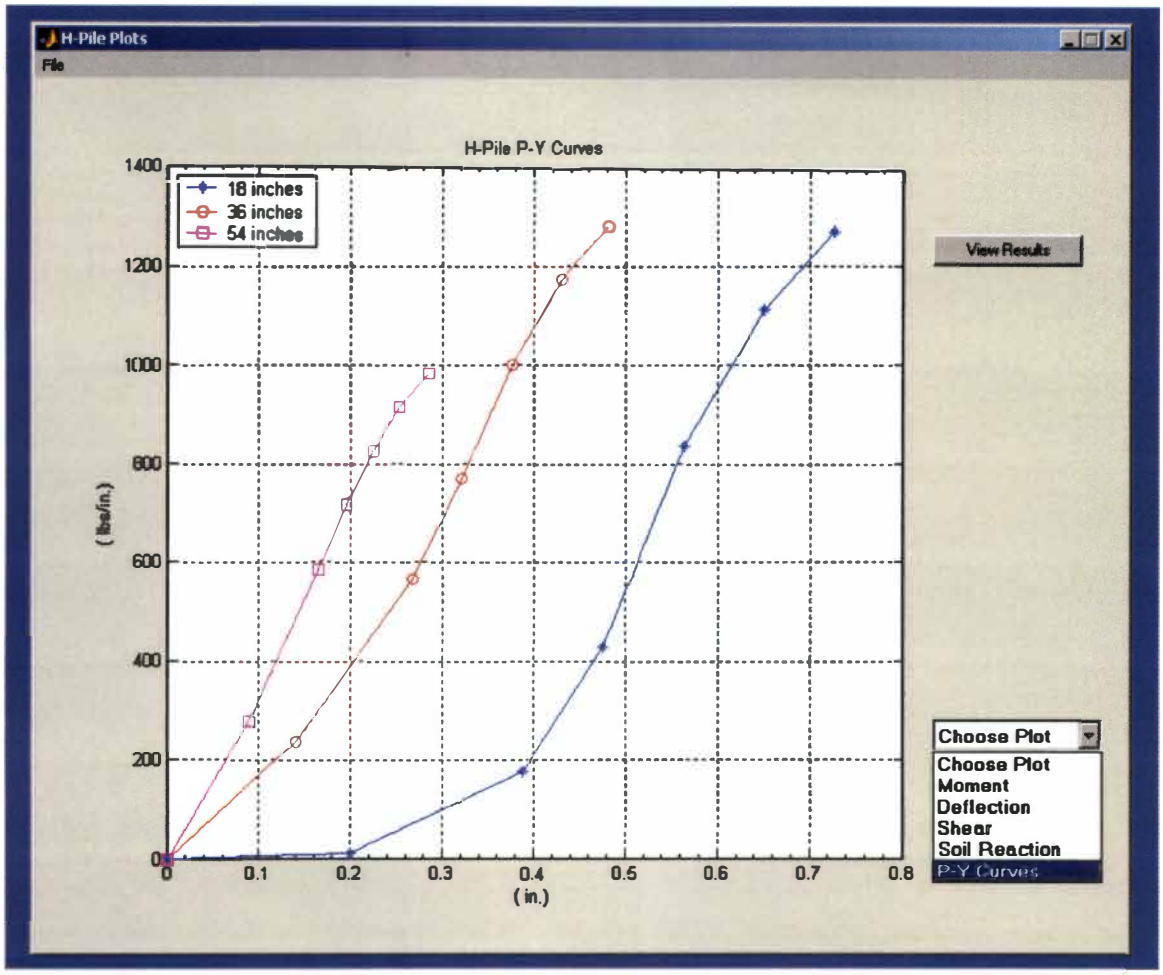

FIGURE A7: HPILE Plot Menu - HPLOT

Appendix B: User's Manual

I System Requirements

HPILE works effectively with MATLAB 6.5, R-13 version on a Windows 95 or higher platform. A minimum storage requirement of 40MB and 32MB RAM is recommended.

II Installation

a) Insert the attached **HPILE Installation CD** Plate into the CD-ROM drive

b) Move or Copy the folder titled **hpile** into C-Drive (C:\) so that the final path name is: **C:\hpile**

c) When copying is completed, open the **hpile** folder and right-click the **piles** folder Select properties from the dropdown list. Uncheck the **read-only** attribute if it is not unchecked and click OK.

d) Open the MATLAB command window and under the file menu, select

Set path

Click Browse to add **hpile** to the path list

Save settings and close the **set path** dialog box

III Using HPILE

Make sure the current directory in the MATLAB environment is set to hpile.

In the command window, type **hpile.**

This opens the HPILE graphical user interface (GUI). From the GUI,

Choose a Pile. Example: **PILE 4E**

Choose a Test Date from the listed dates. Example: Choose **JUL27_99** for **PILE 4E**

Choose a test spreadsheet from the listed spreadsheets. Example: **MOM07279.XLS**

Select a sheet from the drop down list of sheets available. Example: **sheet1**

Choose segment height from the dropdown list. Example: **4.5**

A default **E** of **29000 ksi** is provided for the tested piles. This value may be revised.

The default moment of inertia, **I**, of **226 in^4** may also be revised.

Select **Yes** or **No** for HPILE to perform a p-y analysis. Selecting **Yes** may cause a significant delay in the analytical process. The length of delay is dependent on the computer system processor. Click the **ANALYZE** button when finished

A HPLOT GUI with cleared axes opens at the end of the process. Choose a **MOMENT, SHEAR, DEFLECTION, SOIL REACTION, or P-Y CURVE** plot from the dropdown list. Click the **PLOT** button to display result. If user indicated **No** for a p-y analysis, no result will be displayed when **P-Y CURVE** is chosen.

Click the **View Results** button to review the finite difference solutions. Tabulated results are also presented in the MATLAB command window for printing.

To print plots, go to the file menu on the HPLOT GUI and select **print**.

IV Analyzing piles not cataloged

To analyze non-cataloged piles, open the provided **input template** in the **hpile** folder and edit entries as needed.

Use the **SAVE AS** command in the file dropdown menu of Excel to save the file in the following path: **C:\hpile\piles\NEW\CURRENT**

Run **hpile** as directed above, but select **NEW** from the piles list and **CURRENT** from the date list. Candidate pile for analysis should be displayed in the Excel list

Appendix C: Source Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROGRAM : HPILE ANALYZER v.1                                   %
% AUTHOR  : ALAN TACKIE                                           %
% DATE    : MAY, 2004                                             %
%                                                                 %
% RELATED SOURCES : INGRAM, REESE, MATHWORKS                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function varargout = hpile(varargin)
% HPILE Application M-file for hpile.fig
%    FIG = HPILE launch hpile GUI.
%    HPILE('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.5 13-Sep-2003 21:25:50
%------------------------------------------------------------------------------------------------

% ------------------------ LAUNCH MAIN HPILE GUI -------------------------------------

if nargin == 0

    fig = openfig(mfilename,'reuse');

    % --- Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);

   % --- Change system directory to program's list of piles
   % --- Sort list of piles and fill listbox in GUI with pile names

   cd ('c:\hpile\piles');
   dir_struct = dir('c:\hpile\piles');
   [sorted_names,sorted_index] = sortrows({dir_struct.name}');
   handles.file_names = sorted_names;
   handles.is_dir = [dir_struct.isdir];
   handles.sorted_index = [sorted_index];
   set(handles.listPiles,'String',handles.file_names,'Value',1);

   % --- Save changes to hpile GUI
    guidata(fig, handles);

    if nargout > 0
        varargout{1} = fig;
    end

elseif ischar(varargin{1})
```

```
    try
        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
    catch
        disp(lasterr);
    end
end
%--- END LAUNCH OF MAIN HPILE GUI ----------------------------------


% ---------- HPILE ANALYSIS BEGINS HERE ------------------------------

function varargout = btnAnalyze_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.btnAnalyze.

% --- Piece together user's file path name

 % --- Read Pile Name, Pile Date, Excel File, Test Sheet
 % --- and PY Curve analysis choice from GUI controls to the 'STRING' property
 % --- Read the array index values to their 'VALUE' properties
 % --- Assign user's choices to var1, var2, var3, var4, and py_var


 choice1 = get(handles.listPiles,'String');
 choice1_index = get(handles.listPiles,'Value');
 var1 = choice1{choice1_index(1)};

 choice2 = get(handles.listDate,'String');
 choice2_index = get(handles.listDate,'Value');
 var2 = choice2{choice2_index(1)};

 choice3 = get(handles.listExcelfile,'String');
 choice3_index = get(handles.listExcelfile,'Value');
 var3 = choice3{choice3_index(1)};

 choice4 = get(handles.popTest,'String');
 choice4_index = get(handles.popTest,'Value');
 var4 = choice4{choice4_index(1)};

 choice5 = get(handles.popPYCurves,'String');
 choice5_index = get(handles.popPYCurves,'Value');
 py_var = choice5{choice5_index(1)};


 % --- str and strr contain paths to the excel file and sheet needed for
 % --- program execution
```

```
 str=strcat('c:\hpile\piles\',var1,'\',var2,'\',var3);
  strr=strcat('c:\hpile\piles\',var1,'\',var2,'\',var3,':',var4);


  % --- Delete old text files from subfolders

  delete('c:\hpile\pyfiles\*.*');
  delete('c:\hpile\prgfiles\*.*');


switch py_var
 case 'No'

  % --- Open Excel, add workbook, get necessary matrices from spreadsheet
  % --- Opened Excel application remains invincible as its visible property
  % --- remains set at -1.

  Excel = actxserver('Excel.Application');
  Workbooks = Excel.Workbooks;
  Workbook = invoke(Workbooks,'Open',str);


% --- Use the DDEINIT function to call the spreadsheet from Excel
% --- Assign results of spreadsheet query to myvariable

myvariable = ddeinit('excel',strr);


% --- Check for a NULL condition of opened excel sheet

if myvariable ~= 0

% --- Use the DDEREQ function to extract needed cells to an array

% --- mymatrix: Boundary conditions varied with depth
% --- cellload: Axial loading on pile
% --- topshear: Shear condition at pile cap
% --- topdeflecton: Deflection condition at pile cap

% --- All spreadsheets are formatted to allow specified data abstraction

mymatrix = ddereq(myvariable, 'r3c3:r20c4');  % --- row3,column3 TO row20,column4
cellload = ddereq(myvariable, 'r7c2');        % --- row7,column2
topshear = ddereq(myvariable, 'r3c2');        % --- row3,column2
topdeflection = ddereq(myvariable, 'r6c2');   % --- row6,column2
```

```
% --- Avoid Excel prompt for saving workbook, close workbook, and quit program

Workbook.Saved = 1;
invoke(Workbook, 'Close');
invoke(Excel, 'Quit');

% End process
delete(Excel);


% --- Check for sign on cellload, topshear, topdeflection
% --- Change sign if value is negative

if cellload(1,1)<0;
   cellload(1,1) = cellload(1,1)* -1;
end;

if topshear(1,1)<0;
   topshear(1,1) = topshear(1,1)* -1;
end;

if topdeflection(1,1)<0;
   topdeflection(1,1) = topdeflection(1,1)* -1;
end;


% --- Check for sign on first matrix value
% --- Change sign for proper solution to finite difference equation

if mymatrix(1,1)>0;
   for ii = 1:1:length(mymatrix);
      mymatrix(ii,1) = mymatrix(ii,1) * -1;
   end;
end;

% --- Add additional node at 20ft for bottom boundary condition
% --- Set Moment at depth 20 feet to zero

mymatrix(length(mymatrix)+1 ,1)= 0;
mymatrix(length(mymatrix) ,2)= -20;


% --- Determine the length of imported matrix (mymatrix)
% --- This will allow for reading of last depth item in the matrix
% --- Assign pile length to Pile_Length
```

```
N = length(mymatrix);
Pile_length = mymatrix(N,2);  % --- NB: VALUE READ IS NEGATIVE


% --- Read user's choice of segment height from hpile GUI
% --- Convert string value to numeric and assign to the variable 'h'

choice1 = get(handles.popSegmentheight,'String');
choice1_index = get(handles.popSegmentheight,'Value');
h = str2num(choice1{choice1_index(1)});


% --- Determine number of segments pile is to be divided into based on segment height
     'h'
% --- Round 'n' to the nearest integer using the round function
% --- Negative pile length is first converted to positive inches

n = round((Pile_length * -12)/h);

% --- Calculate total number of nodes based on 'n' and assign value to 'nodes'
% --- nodes is exclusive of last data point which represents conditions at the
% --- end of the pile

nodes = n+5;

% --- Read user's choice of Modulus of Elasticity and Moment of Inertia from hpile GUI
% --- Convert string values to numbers and assign to 'E' and 'I' respectively

str1 = get(handles.txtMOE,'String');
E = str2num(str1);


str2 = get(handles.txtMOI,'String');
I = str2num(str2);



% --- Determine Q based on user's pile choice
% --- Piles 1,2,3 are based on a cap of 65 kips
% --- Piles 4,5 are based on a cap of 90 kips
% --- Set Default to 90 kips

switch var1
 case {'PILE1E','PILE2E','PILE3E','PILE1W','PILE2W','PILE3W'}
   Q = 65 - cellload(1,1);
```

```
 case {'PILE4E','PILE4W','PILE5N'}
   Q = 90 - cellload(1,1);

 otherwise
   Q = 90;
end

% --- Perform a POLYFIT on mymatrix to obtain a regression function
% --- Regression function is of the order 6
% --- POLYFIT(DEPTH, MOMENT)

b_regr_coef = polyfit(mymatrix(:,2),mymatrix(:,1),6);

% --- Find and sort the real roots of b_regr_coef function
% --- to determine the depth of dimished moment

b_root = roots(b_regr_coef);

jj = 1;
for kk=1:1:length(b_root);
if isreal(b_root(kk,1))==1;
   if b_root(kk,1) < 0;
      mom_roots(jj,1)=b_root(kk,1);
      jj = jj+1;
   end
end
end

 mom_roots = sort(mom_roots);

% --- 'no_mom' and 'no_mom2' are the 2nd and 1st zero moment points
% --- from the bottom of the pile

if length(mom_roots) > 2;
   no_mom = mom_roots(length(mom_roots)-2,1);
   no_mom2 = mom_roots(length(mom_roots)-1,1);
else
   no_mom = mom_roots(1,1);
   no_mom2 = -20;
end

% --- If no_mom is below 20ft, set the ZERO moment boundary condition
% --- halfway between 20ft and the next upper zero moment (no_mom2)

if no_mom > -20;
```

```
   zer_pos = ceil(-no_mom.*12./h);
else;
   zer_pos = ceil((-((no_mom2-20).*12)./2)./h);
end;


% ---------- WORK FROM INGRAM'S DISSERTATION ----------
% --- GENERATION OF COEFFICIENT MATRIX FOR HPILE ANALYSIS

% create R vector assuming pile extends above the ground surface
R = zeros(nodes,1);
for ii = 1:1:nodes-2;
  R(ii,1) = E.*I;
end;

% initialize coefficient matrix of appropriate size
coef = zeros(2.*nodes-4,2.*nodes);

% insert non-zero coefficients for finite difference equation
for row1 = 1:1:nodes-4;
  col1 = 2.*row1 + 3;
  coef = add_delta(coef,row1,col1,R(row1+1,1),R(row1+2,1),R(row1+3,1),h,Q);
end;

% insert finite difference coefficients for deflection boundary conditions
coef = add_yb(coef,nodes-3,5);
coef = add_yb(coef,nodes-2,2.*nodes-5);


% insert finite difference coefficients for shear boundary conditions
coef = add_vb(coef,nodes-1,5,R(3,1),h,Q);
coef = add_vb(coef,nodes,2 .* nodes-5,R(nodes-3,1),h,Q);


% insert finite difference coefficients for moment versus depth boundary conditions
for row2 = nodes+1:1:2.*nodes-4;
  col2 = (row2-nodes).*2 + 3;
  coef = add_mb(coef,row2,col2,R((col2-1)./2+1,1),h);
end;

% remove lines corresponding to yk for imaginary nodes in coefficient matrix
% and define A as the reduced coefficient matrix
A = zeros(2.*nodes-4,2.*nodes-4);
A(:,1) = coef(:,1);
A(:,2) = coef(:,3);
```

```
for col3 = 3:1:2.*nodes-6;
  A(:,col3) = coef(:,col3+2);
end;

A(:,2.*nodes-5) = coef(:,2.*nodes-3);
A(:,2.*nodes-4) = coef(:,2.*nodes-1);


% ---------- COEFFIECENT MATRIX GENERATION ENDS HERE ----------
% -------------------------------------------------------------------------------------------

% --- NB: Array 'b' defined below runs from (1 -> n) and from (n+1 ->2n-4)
% --- NB: An array 'bx', a subset of 'b', is defined from (n+1 ->2n-4) and has
%         corresponding terms starting from 1 -> (n-4)
% --- NB: n-terns + (n-4)-terms = 2n-4 terms
% --- NB: (n-4)-terms includes the (n+1) term in array 'b'


% --- Create an array 'b' to hold generated coefficient matrix solution

b = zeros(1,2 .* nodes-4);

% --- Create an array 'bx' to hold generated regression function solution

bx = zeros(1,nodes-4);

% --- Fill bx with incremental depth values for the polyfit function to use

for ii = 2:1:length(bx);
  bx(1,ii) = bx(1,ii-1)- h;
end;


% --- Use 'b-regr_coef' and the polyval function to generate solution 'by'
% --- expressed in kip-ft

by = polyval(b_regr_coef, bx./12);

% --- Fill the 'b' matrix from cells (n+1) to 'zer_pos' with 'by' values

for ii = 1:1:zer_pos;
  b(1,ii+nodes) = by(1,ii).*12;
end;
```

```
% --- Modify 'b' for deflection and shear boundary conditions at the top of pile
% --- It is assumed conditions at the bottom are zero for deflection and shear
b(1,nodes-3)= topdeflection;
b(1,nodes-1)= topshear;


% --- Convert boundary condition data to a column vector

b = b';

% --- Solve finite difference equation

sol = A\b;


% ---------- WORK FROM INGRAM'S DISSERTATION ----------
% ---------- SEPARATION OF SOLUTION BEGINS HERE----------


% seperate sol into a deflection vector and a soil reaction per length vector
jj=1;
for ii = 3:2:length(sol)-3;
  y(jj,1)=sol(ii);
  yk(jj,1)=sol(ii+1);
  jj=jj+1;
end;

% calculate a soil reaction vector
g = yk.*h;
g(1,1) = yk(1,1).*h./2;
g(length(yk),1) = yk(length(yk),1).*h./2;

% calculate a shear vector
v = zeros(length(y),1);
v(1,1)=R(3,1)./(2.*h.^3).*(y(3,1)-2.*y(2,1)+2.*sol(2,1)-sol(1,1))+Q./(2.*h).*(y(2,1)-
    sol(2,1));
v(2,1)=R(4,1)./(2.*h.^3).*(y(4,1)-2.*y(3,1)+2.*y(1,1)-sol(2,1))+Q./(2.*h).*(y(3,1)-
    y(1,1));
for ii = 3:1:length(y)-2;
  v(ii,1)=R(ii+2,1)./(2.*h.^3).*(y(ii+2,1)-2.*y(ii+1,1)+2.*y(ii-1,1)-y(ii-
    2,1))+Q./(2.*h).*(y(ii+1,1)-y(ii-1,1));
end;

% create x vector of depth
x(1,1) = 0;
```

```
for ii = 2:1:length(y);
  x(ii,1) = x(ii-1,1)+h;
end;

% create moment vector from boundary condition data
m = b(nodes+1:length(b),1);

% ---------- SEPARATION OF SOLUTION ENDS HERE ----------


% --- Evaluate Coefficient of Subgrade Reaction (Uh) and Relative Stiffness (Alpha)
% --- Initialize variables

Uh = zeros(length(x),1);
Alpha = zeros(length(x),1);

for vv = 2:1:(length(x)-1);
   Uh(vv) = (1000 .* g(vv))./(h .* x(vv) .* y(vv));
   Alpha(vv) = real((((Uh(vv)./(1000 .*E .* I)).^0.2).*10000000)./10000000);
end

% --- Modify results for bottom of pile conditions

Length = length(x);
m(Length) = 0;
y(Length) = 0;
v(Length) = 0;
g(Length) = 0;


% --- Save results for future use by hpile plot function

Uh = round(Uh);
save c:\hpile\prgfiles\depth.txt x -ascii
save c:\hpile\prgfiles\moment.txt m -ascii
save c:\hpile\prgfiles\deflection.txt y -ascii
save c:\hpile\prgfiles\shear.txt v -ascii
save c:\hpile\prgfiles\reaction.txt g -ascii
save c:\hpile\prgfiles\Subgrade.txt Uh -ascii
save c:\hpile\prgfiles\Rel_Stiffness.txt Alpha -ascii

results(:,1)= x;
results(:,2)= (round(10*m))./10;
results(:,3)= (round(1000*y))./1000;
results(:,4)= (round(1000*v))./1000;
```

```
results(:,5)= (round(1000*g))./1000;
results(:,6)= Uh;
results(:,7)= Alpha

save c:\hpile\prgfiles\results.txt results -ascii


% --- Open plot window and clear axes

openfig('hplots.fig')
cla;




% Display error message if 'myvariable' is empty

else
 %Display error message for no boundary conditions in sheet chosen
 %Close Excel

 msgbox('NO BOUNDARY VALUES EXIST FOR SELECTED EXCEL SHEET. TRY
    SHEETS 1 - 4','H-PILE BOUNDARY DATA ERROR')

 Workbook.Saved = 1;
 invoke(Workbook, 'Close');
 invoke(Excel, 'Quit');

end
% ------------------------------------ END IF ELSE STATEMENT
% ------------------------- END OF PRGRAM WITHWOUT P-Y ANALYSIS ----------



% --- Perform complete pile analysis - Include PY analysis ---------------------------------


case 'Yes'
   % --- Open Excel, add workbook, get necessary matrices from spreadsheet
   % --- Opened Excel application remains invincible as its visible property
   % --- remains set at -1.

   Excel = actxserver('Excel.Application');
   Workbooks = Excel.Workbooks;
   Workbook = invoke(Workbooks,'Open',str);
```

```
myvariable = ddeinit('excel',strr);

Workbook.Saved = 1;
invoke(Workbook, 'Close');
invoke(Excel, 'Quit');

% End process
delete(Excel);


% --- Set sheet number to one
% --- Initialize variables
% --- Check for a NULL condition of opened excel sheet

sheetnumber = 1;
y_index5 = zeros;
y_index9 = zeros;
y_index13 = zeros;
p_index5 = zeros;
p_index9 = zeros;
p_index13 = zeros;
py_results = zeros;
results = zeros;


while myvariable ~= 0


% --- Reset myvariable to process all test sheets for p-y analysis

myvariable = 0;
number = int2str(sheetnumber);
strnumber = strcat('sheet',number);
strr = strcat('c:\hpile\piles\',var1,'\',var2,'\',var3,':',strnumber);

% --- Open Excel, add workbook, get necessary matrices from spreadsheet
% --- Opened Excel application remains invincible as its visible property
% --- remains set at -1.

Excel = actxserver('Excel.Application');
Workbooks = Excel.Workbooks;
Workbook = invoke(Workbooks,'Open',str);

myvariable = ddeinit('excel',strr);
if myvariable ~=0
```

```
% --- Use the DDEREQ function to extract needed cells to an array

% --- mymatrix: Boundary conditions varied with depth
% --- cellload: Axial loading on pile
% --- topshear: Shear condition at pile cap
% --- topdeflecton: Deflection condition at pile cap

% --- All spreadsheets are formatted to allow specified data abstraction

mymatrix = ddereq(myvariable, 'r3c3:r20c4'); % --- row3,column3 TO
 row20,column4
cellload = ddereq(myvariable, 'r7c2'); % --- row7,column2
topshear = ddereq(myvariable, 'r3c2'); % --- row3,column2
topdeflection = ddereq(myvariable, 'r6c2'); % --- row6,column2


% --- Check for sign on cellload, topshear, topdeflection
% --- Change sign if value is negative

if cellload(1,1)<0;
   cellload(1,1) = cellload(1,1)* -1;
end;

if topshear(1,1)<0;
   topshear(1,1) = topshear(1,1)* -1;
end;

if topdeflection(1,1)<0;
   topdeflection(1,1) = topdeflection(1,1)* -1;
end;


% --- Check for sign on first matrix value
% --- Change sign for proper solution to finite difference equation

if mymatrix(1,1)>0;
   for ii = 1:1:length(mymatrix);
      mymatrix(ii,1) = mymatrix(ii,1) * -1;
   end;
end;



% --- Add additional node for bottom boundary condition
% --- Set Moment at depth 20 feet to 0
```

```
mymatrix(length(mymatrix)+1 ,1)= 0;
mymatrix(length(mymatrix) ,2)= -20;

% --- Determine the length of imported matrix (mymatrix)
% --- This will allow for reading of last depth item in the matrix
% --- Assign pile length to Pile_Length

N = length(mymatrix);
Pile_length = mymatrix(N,2);  % --- NB: VALUE READ IS NEGATIVE


% --- Read user's choice of segment height from hpile GUI
% --- Convert string value to numeric and assign to the variable 'h'

choice1 = get(handles.popSegmentheight,'String');
choice1_index = get(handles.popSegmentheight,'Value');
h = str2num(choice1{choice1_index(1)});


% --- Determine number of segments pile is to be divided into based on segment height
  'h'
% --- Round 'n' to the nearest integer using the round function
% --- Negative pile length is first converted to positive inches

n = round((Pile_length * -12)/h);

% --- Calculate total number of nodes based on 'n' and assign value to 'nodes'
% --- nodes is exclusive of last data point which represents conditions at the
% --- end of the pile

nodes = n+5;

% --- Read user's choice of Modulus of Elasticity and Moment of Inertia from hpile
  GUI
% --- Convert string values to numbers and assign to 'E' and 'I' respectively

str1 = get(handles.txtMOE,'String');
E = str2num(str1);


str2 = get(handles.txtMOI,'String');
I = str2num(str2);
% --- Determine Q based on user's pile choice
% --- Piles 1,2,3 are based on a cap of 65 kips
% --- Piles 4,5 are based on a cap of 90 kips
```

```
% --- Set Default to 90 kips

switch var1
   case {'PILE1E','PILE2E','PILE3E','PILE1W','PILE2W','PILE3W'}
      Q = 65 - cellload(1,1);

   case {'PILE4E','PILE4W','PILE5N'}
      Q = 90 - cellload(1,1);

   otherwise
      Q = 90;
end


% --- Perform a POLYFIT on mymatrix to obtain a regression function
% --- Regression function is of the order 6
% --- POLYFIT(DEPTH, MOMENT)

b_regr_coef = polyfit(mymatrix(:,2),mymatrix(:,1),6);

% --- Find and sort the real roots of b_regr_coef function
% --- to determine the depth of dimished moment

b_root = roots(b_regr_coef);

jj = 1;
for kk=1:1:length(b_root);
if isreal(b_root(kk,1))==1;
   if b_root(kk,1) < 0;
      mom_roots(jj,1)=b_root(kk,1);
      jj = jj+1;
   end
end
end

mom_roots = sort(mom_roots);

% --- 'no_mom' and 'no_mom2' are the 2nd and 1st zero moment points
% --- from the bottom of the pile

if length(mom_roots) > 2;
   no_mom = mom_roots(length(mom_roots)-2,1);
   no_mom2 = mom_roots(length(mom_roots)-1,1);
else
   no_mom = mom_roots(1,1);
```

```
   no_mom2 = -20;
end

% --- If no_mom is below 20ft, set the ZERO moment boundary condition
% --- halfway between 20ft and the next upper zero moment (no_mom2)

if no_mom > -20;
   zer_pos = ceil(-no_mom.*12./h);
else;
   zer_pos = ceil((-((no_mom2-20).*12)./2)./h);
end;


% ---------- WORK FROM INGRAM'S DISSERTATION ----------
% --- GENERATION OF COEFFICIENT MATRIX FOR HPILE ANALYSIS


% create R vector assuming pile extends above the ground surface
R = zeros(nodes,1);
for ii = 1:1:nodes-2;
   R(ii,1) = E.*I;
end;

% initialize coefficient matrix of appropriate size
coef = zeros(2.*nodes-4,2.*nodes);

% insert non-zero coefficients for finite difference equation
for row1 = 1:1:nodes-4;
   col1 = 2.*row1 + 3;
   coef = add_delta(coef,row1,col1,R(row1+1,1),R(row1+2,1),R(row1+3,1),h,Q);
end;

% insert finite difference coefficients for deflection boundary conditions
coef = add_yb(coef,nodes-3,5);
coef = add_yb(coef,nodes-2,2.*nodes-5);

% insert finite difference coefficients for shear boundary conditions
coef = add_vb(coef,nodes-1,5,R(3,1),h,Q);
coef = add_vb(coef,nodes,2.*nodes-5,R(nodes-3,1),h,Q);
% insert finite difference coefficients for moment versus depth boundary conditions
for row2 = nodes+1:1:2.*nodes-4;
   col2 = (row2-nodes).*2 + 3;
   coef = add_mb(coef,row2,col2,R((col2-1)./2+1,1),h);
end;
```

```
% remove lines corresponding to yk for imaginary nodes in coefficient matrix
% and define A as the reduced coefficient matrix
A = zeros(2.*nodes-4,2.*nodes-4);
A(:,1) = coef(:,1);
A(:,2) = coef(:,3);

for col3 = 3:1:2.*nodes-6;
   A(:,col3) = coef(:,col3+2);
end;

A(:,2.*nodes-5) = coef(:,2.*nodes-3);
A(:,2.*nodes-4) = coef(:,2.*nodes-1);


% ---------- COEFFIECENT MATRIX GENERATION ENDS HERE ----------


% --- Create an array 'b' to hold generated coefficient matrix solution

b = zeros(1,2 .* nodes-4);

% --- Create an array 'bx' to hold generated regression function solution

bx = zeros(1,nodes-4);

% --- Fill bx with incremental depth values for the polyfit function to use

for ii = 2:1:length(bx);
   bx(1,ii) = bx(1,ii-1)- h;
end;

% --- Use 'b-regr_coef' and the polyval function to generate solution 'by'
% --- expressed in kip-ft

by = polyval(b_regr_coef, bx./12);

% --- Fill the 'b' matrix from cells (n+1) to 'zer_pos' with 'by' values
for ii = 1:1:zer_pos
   b(1,ii+nodes) = by(1,ii).*12;
end;


% --- Modify 'b' for deflection and shear boundary conditions at the top of pile
% --- It is assumed conditions at the bottom are zero for deflection and shear
```

```
b(1,nodes-3)= topdeflection;
b(1,nodes-1)= topshear;


% --- Convert boundary condition data to a column vector

b = b';

% --- Solve finite difference equation

sol = A\b;



% ---------- WORK FROM INGRAM'S DISSERTATION ----------
% ---------- SEPARATION OF SOLUTION BEGINS HERE ----------

% --- Initialize variables

y=0;
yk=0;
g=0;
v=0;
x=0;
m=0;


% seperate sol into a deflection vector and a soil reaction per length vector
jj=1;
for ii = 3:2:length(sol)-3;
   y(jj,1)=sol(ii);
   yk(jj,1)=sol(ii+1);
   jj=jj+1;
end;

% calculate a soil reaction vector
g = yk.*h;
g(1,1) = yk(1,1).*h./2;
g(length(yk),1) = yk(length(yk),1).*h./2;
% calculate a shear vector
v = zeros(length(y),1);
v(1,1)=R(3,1)./(2.*h.^3).*(y(3,1)-2.*y(2,1)+2.*sol(2,1)-sol(1,1))+Q./(2.*h).*(y(2,1)-
 sol(2,1));
v(2,1)=R(4,1)./(2.*h.^3).*(y(4,1)-2.*y(3,1)+2.*y(1,1)-sol(2,1))+Q./(2.*h).*(y(3,1)-
 y(1,1));
```

```
for ii = 3:1:length(y)-2;
   v(ii,1)=R(ii+2,1)./(2.*h.^3).*(y(ii+2,1)-2.*y(ii+1,1)+2.*y(ii-1,1)-y(ii-
 2,1))+Q./(2.*h).*(y(ii+1,1)-y(ii-1,1));
end;

% create x vector of depth
x(1,1) = 0;
for ii = 2:1:length(y);
   x(ii,1) = x(ii-1,1)+h;
end;

% create moment vector from boundary condition data
m = b(nodes+1:length(b),1);

% ---------- SEPARATION OF SOLUTION ENDS HERE ----------


% --- Evaluate Coefficient of Subgrade Reaction (Uh) and Relative Stiffness (Alpha)
% --- Initialize variables

Uh = zeros(length(x),1);
Alpha = zeros(length(x),1);

for vv = 2:1:(length(x)-1);
Uh(vv,1) = (1000 .* g(vv))./(h .* x(vv) .* y(vv));
Alpha(vv,1) = real((((Uh(vv)./(E .* I)).^0.2).*10000000)./10000000);
end

% --- Save Results

if strcmp(var4,strnumber)==1;

   Uh = round(Uh);
   save c:\hpile\prgfiles\depth.txt x -ascii
   save c:\hpile\prgfiles\moment.txt m -ascii
   save c:\hpile\prgfiles\deflection.txt y -ascii
   save c:\hpile\prgfiles\shear.txt v -ascii
   save c:\hpile\prgfiles\reaction.txt g -ascii
   save c:\hpile\prgfiles\Subgrade.txt Uh -ascii
   save c:\hpile\prgfiles\Rel_Stiffness.txt Alpha -ascii

   for ww = 1:1:length(x);
      results(ww,1)= x(ww);
      results(ww,2)= (round(10*m(ww)))./10;
      results(ww,3)= (round(1000*y(ww)))./1000;
```

```
      results(ww,4)= (round(1000*v(ww)))./1000;
      results(ww,5)= (round(1000*g(ww)))./1000;
      results(ww,6)= Uh(ww);
      results(ww,7)= Alpha(ww);
   end

   save c:\hpile\prgfiles\results.txt results -ascii

else
   Length = length(x);
   py_results = zeros(Length,5);
   py_results(:,1)= x;
   py_results(:,2)= m;
   py_results(:,3)= y;
   py_results(:,4)= v;
   py_results(:,5)= g;

   filename = ['c:\hpile\pyfiles\py_results', number, '.txt'];
   eval(['save ', filename , ' py_results -ascii']);

   clc; % --- Clear Matlab WindowCLEAR MATLAB WINDOW

end
% --- END IF STATEMENT FOR FILE SAVE

% --- hvalue determines switch case to evaluate
% --- Save p-y values at depths of 18", 36", and 54"
% --- Return absolute values for plotting purposes
% --- UU, VV, and WW determine depths of 18", 36", and 54"

hvalue = num2str(h);

switch hvalue
   case '2'
      UU=10;
      VV=19;
      WW=28;
   case '4.5'
      UU=5;
      VV=9;
      WW=13;
   case '8'
      UU=3;
      VV=5;
      WW=8;
```

```
  end


  y_index5(sheetnumber +1) = abs(y(UU));
  p_index5(sheetnumber +1) = abs(1000*g(UU)./h);

  y_index9(sheetnumber +1) = abs(y(VV));
  p_index9(sheetnumber +1) = abs(1000*g(VV)./h);

  y_index13(sheetnumber +1) = abs(y(WW));
  p_index13(sheetnumber +1) = abs(1000*g(WW)./h);


  % --- Increment sheet number
  sheetnumber = sheetnumber + 1;

  end
  % --- END IF STATEMENT


  Workbook.Saved = 1;
  invoke(Workbook, 'Close');
  invoke(Excel, 'Quit');

  % End process
  delete(Excel);


  % Display error message if 'myvariable' is empty
  % Display error message for no boundary conditions in sheet chosen

  if sheetnumber == 1;
     msgbox('NO BOUNDARY VALUES EXIST FOR SELECTED EXCEL SHEET.
    TRY SHEETS 1 - 4','H-PILE BOUNDARY DATA ERROR')
     Workbook.Saved = 1;
     invoke(Workbook, 'Close');
     invoke(Excel, 'Quit');
     % End process
     delete(Excel);


end
end

% ---  END WHILE STATEMENT
```

```
% --- Provide error message if user choice fails switch case statement

if sheetnumber == 1;
     msgbox('NO BOUNDARY VALUES EXIST FOR SELECTED EXCEL SHEET.
    TRY SHEETS 1 - 4','H-PILE BOUNDARY DATA ERROR')
else

   % --- Save py_analyses results for the hplot GUI

   save c:\hpile\prgfiles\y_index5.txt y_index5 -ascii
   save c:\hpile\prgfiles\y_index9.txt y_index9 -ascii
   save c:\hpile\prgfiles\y_index13.txt y_index13 -ascii

   save c:\hpile\prgfiles\p_index5.txt p_index5 -ascii
   save c:\hpile\prgfiles\p_index9.txt p_index9 -ascii
   save c:\hpile\prgfiles\p_index13.txt p_index13 -ascii

   % --- Open plot window and clear axes

   openfig('hplots.fig')
   cla;
   clc;
   %clear all;

end
end

% ------------------------------- END MAIN CASE STATEMENT-------------------------
% -------------------------- END OF PROGRAM WITH P-Y ANALYSIS ------------------
```

```
% --- HPILE FUNCTIONS FOR GUI CONTROL FUNCTIONS

function varargout = listPiles_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.listPiles.

% --- Read pile choice and assign result to 'var'

choice = get(handles.listPiles,'String');
choice_index = get(handles.listPiles,'Value');
var = choice{choice_index(1)};

% --- Create path name, change directory, and read contents of directory.
% --- Assign sorted directory list to the 'listDate' and save GUI

str=strcat('c:\hpile\piles\',var);
cd(str);
dir_struct = dir(str);
[sorted_names,sorted_index] = sortrows({dir_struct.name}');
handles.file_names = sorted_names;
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = [sorted_index];
set(handles.listDate,'String',handles.file_names,'Value',1)

%guidata(fig, handles);


function varargout = listDate_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.listDate.

% --- Read pile choice and assign result to 'var'
% --- Read date choice and assign result to 'var1'

choice1 = get(handles.listPiles,'String');
choice1_index = get(handles.listPiles,'Value');
var1 = choice1{choice1_index(1)};

choice2 = get(handles.listDate,'String');
choice2_index = get(handles.listDate,'Value');
var2 = choice2{choice2_index(1)};

% --- Create path name, change directory, and read contents of directory.
% --- Assign sorted directory list to the 'listExcelfile' listbox and save GUI

str=strcat('c:\hpile\piles\',var1,'\',var2);
cd(str);
```

```
dir_struct = dir(str);
[sorted_names,sorted_index] = sortrows({dir_struct.name}');
handles.file_names = sorted_names;
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = [sorted_index];
set(handles.listExcelfile,'String',handles.file_names,'Value',1)

%guidata(hpile, handles);


function submenuExit_Callback(hObject, eventdata, handles)

% --- Close Matlab

exit;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROGRAM : HPILE ANALYZER    -    HPLOT  SOURCE CODE           %
% AUTHOR  : ALAN TACKIE                                         %
% DATE    MAY, 2004                                             %
%                                                               %
% RELATED SOURCES : INGRAM, REESE, MATHWORKS                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% --- LAUNCH HPLOT GUI ------------------------------------------

function varargout = hplots(varargin)

%   HPLOTS Application M-file for hplots.fig
%   HPLOTS, by itself, creates a new HPLOTS or raises the existing
%   singleton*.
%
%   H = HPLOTS returns the handle to a new HPLOTS or the handle to
%   the existing singleton*.
%
%   HPLOTS('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in HPLOTS.M with the given input arguments.
%
%   HPLOTS('Property','Value',...) creates a new HPLOTS or raises the
%   existing singleton*.  Starting from the left, property value pairs are
%   applied to the GUI before hplots_OpeningFunction gets called.  An
%   unrecognized property name or invalid value makes property application
%   stop.  All inputs are passed to hplots_OpeningFcn via varargin.
%
%   *See GUI Options - GUI allows only one instance to run (singleton).
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help hplots

% Last Modified by GUIDE v2.5 21-Jun-2003 12:17:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @hplots_OpeningFcn, ...
                   'gui_OutputFcn',  @hplots_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
```

```
                'gui_Callback',     []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

%--------------------------------------------------------------------------------------------

% --- Executes just before hplots is made visible.

function hplots_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for hplots

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = hplots_OutputFcn(hObject, eventdata, handles)


% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function popPlots_CreateFcn(hObject, eventdata, handles)

if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
%----------------------END LAUNCH OF HPLOT GUI ------------------------------------
```

```
% -------------------------- HPLOT BEGINS HERE -------------------------------------------

% --- Executes on button press in plot_button.

function plot_button_Callback(hObject, eventdata, handles)


% --- Obtain GUI handles, read plot choice and assign value to 'userchoice'

handles = gcf;
guidata(handles);
handles = guihandles(handles);

choice = get(handles.popPlots,'String');
choice_index = get(handles.popPlots,'Value');
userchoice = choice{choice_index(1)};
close(gcf);


% --- Open 'hplots' GUI
% --- Clear axes
% --- Obtain current handles

openfig('hplots.fig');
cla;
handles = gcf;
guidata(handles);
handles = guihandles(handles);
% --- Change current directory to prfiles, which contains processed results
% --- from the analysis menu

cd ('c:\hpile\prgfiles');

% --- Import hpile analyzed data into the plot function
% --- Open text files, scan for results and close text file

depth = fopen('depth.txt');
x = fscanf(depth,'%g');
x = -1.*x;              % --- Convert depth values to negative for plotting purposes
fclose(depth);

moment = fopen('moment.txt');
m = fscanf(moment,'%g');
fclose(moment);
deflection = fopen('deflection.txt');
```

```
y = fscanf(deflection,'%g');
fclose(deflection);
shear = fopen('shear.txt');
v = fscanf(shear,'%g');
fclose(shear);


reaction = fopen('reaction.txt');
g = fscanf(reaction,'%g');
fclose(reaction);

% --- Evaluate node depths for graph legend

node5label = strcat(num2str(x(2,1)*-4),' inches');
node9label = strcat(num2str(x(2,1)*-8),' inches');
node13label = strcat(num2str(x(2,1)*-12),' inches');


% --- Generate plots based on user's choice from the hplot GUI

% --- Clear axes
% --- Plot choice versus depth
% --- Make the x-axis tick marks visible
% --- Set the plot title appropriately
% --- Set the appropriate units for the x-axis
% --- Set the grid property on


switch userchoice

   case 'Moment'
      clc;
      plot(m,x)
      set(handles.plotaxes,'XMinorTick','on')
      title('H-Pile Moment Vs. Depth')
      xlabel('( kip-in.)')
      ylabel('( in.)')
      grid on

   case 'Deflection'
      clc;
      axes(handles.plotaxes)
      plot(y,x)
      set(handles.plotaxes,'XMinorTick','on')
      title('H-Pile Deflection Vs. Depth')
```

```
    xlabel('( in.)')
    ylabel('( in.)')
    grid on

case 'Shear'
    clc;
    axes(handles.plotaxes)
    plot(v,x)
    set(handles.plotaxes,'XMinorTick','on')
    title('H-Pile Shear Vs. Depth')
    xlabel('( kips.)')
    ylabel('( in.)')
    grid on

case 'Soil Reaction'
    clc;
    axes(handles.plotaxes)
    plot(g,x)
    set(handles.plotaxes,'XMinorTick','on')
    title('H-Pile Soil Reaction Vs. Depth')
    xlabel('( kips.)')
    ylabel('( in.)')
    grid on

case 'P-Y Curves'
    p_index5 = fopen('p_index5.txt');
    p5 = fscanf(p_index5,'%g');
    fclose(p_index5);
   y_index5 = fopen('y_index5.txt');
    y5 = fscanf(y_index5,'%g');
    fclose(y_index5);

    p_index9 = fopen('p_index9.txt');
    p9 = fscanf(p_index9,'%g');
    fclose(p_index9);

    y_index9 = fopen('y_index9.txt');
    y9 = fscanf(y_index9,'%g');
    fclose(y_index9);

    p_index13 = fopen('p_index13.txt');
    p13 = fscanf(p_index13,'%g');
    fclose(p_index13);

    y_index13 = fopen('y_index13.txt');
```

```
      y13 = fscanf(y_index13,'%g');
      fclose(y_index13);
      clc;
      axes(handles.plotaxes)
      plot(y5,p5,'-b*',y9,p9,'-ro',y13,p13,'-ms')
      set(handles.plotaxes,'XMinorTick','on')
      title('H-Pile P-Y Curves')
      xlabel('( in.)')
      ylabel('( lbs/in.)')
      % --- Set legend values. 8" segment length has no depth results at 18", 36", and 54"
      % --- Value 2 sets position of legend to the upper left corner

      if abs(x(2))==8;
         legend('16 inches','32 inches','56 inches',2)
         grid on
      else
         legend('18 inches','36 inches','54 inches',2)
         grid on
      end

   case 'Choose Plot'  % --- Default clears screen and axes
      clc;
      cla;


end
% --- Executes on button press in btnResults.
function btnResults_Callback(hObject, eventdata, handles)



% --- Get handles of current GUI
% --- Save handles

% --- Change current directory to prfiles, which contains processed results
% --- from the analysis menu
cd ('c:\hpile\prgfiles');

handles = gcf;
guidata(handles);
handles = guihandles(handles);
% --- Open and read results.txt
% --- Read contents, assign contents to 'results' and close file channel

Resultstr = fopen('results.txt');
```

```
results = fscanf(Resultstr,'%g %g %g %g %g',[5 inf]); % It has infinite rows and 5
columns
results = results';
fclose(Resultstr);

% --- Convert numeric results to string using num2str function

results = num2str(results);


% --- Open view results GUI, optain current handle, set its
% --- 'String' property to 'result'

openfig('viewresults.fig');
handles = gcf;
guidata(handles);
handles = guihandles(handles);
set(handles.listResults,'String',results,'Value',1);


% --- Save GUI

guidata(gcf);
handles = guihandles(handles);
function submenuClose_Callback(hObject, eventdata, handles)

% --- Close window

close;


% --------------------------------------------------------------------
function submenuPrint_Callback(hObject, eventdata, handles)

% --- Print graphical output without GUI controls
% --- Clear screen

print -noui;
clc;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROGRAM : HPILE ANALYZER    - PRINTRESULTS  SOURCE CODE     %
% AUTHOR  : ALAN TACKIE                                        %
% DATE    : MAY, 2004                                          %
%                                                              %
% RELATED SOURCES : INGRAM, REESE, MATHWORKS                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% --- Executes on button press in btnPrint.

function btnPrint_Callback(hObject, eventdata, handles)

% --- Open result files, read required data, and plot results in the main
% --- GUI. This should allow for easy printing while in the matlab
% --- application GUI

depth = fopen('depth.txt');
x = fscanf(depth,'%g');
fclose(depth);
moment = fopen('moment.txt');
m = fscanf(moment,'%g');
fclose(moment);

deflection = fopen('deflection.txt');
y = fscanf(deflection,'%g');
fclose(deflection);

shear = fopen('shear.txt');
v = fscanf(shear,'%g');
fclose(shear);

reaction = fopen('reaction.txt');
g = fscanf(reaction,'%g');
fclose(reaction);


% --- CLEAR SCREEN FOR RESULT DISPLAY

 clc;
```

```
% Print a table of depth, deflection, moment, shear, shear differential, and soil reaction

fprintf('\n  Depth     Def.      Moment      Shear      Soil React.');
fprintf('\n  (in)      (in)      (k-in)      (kips)      (kips) ');
fprintf('\n  -----     ------    --------    -------    -----------');

for ii = 1:1:length(y);
  fprintf('\n  %5.1f   %8.5f   %7.1f    %6.2f      %7.4f',x(ii),y(ii),m(ii),v(ii),g(ii));
end;
fprintf('\n\n');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROGRAM : HPILE ANALYZER    - CALLBACK FUNCTIONS          %
% AUTHOR  : INGRAM                                           %
% DATE    : AUGUST, 2002                                     %
%                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function m=add_delta(cm,row,col,r1,r2,r3,h,Q)

% Add to coefficient matrix
% Input: cm = coefficient matrix
%       row = line number of addition
%       col = column of first entry (ym-2)
%       r1 = Rm-1 EI for section of pile
%       r2 = Rm EI for section of pile
%       r3 = Rm+1 EI for section of pile
%       h = length of segment
%       Q = axial load on pile
% Output: coefficients for ym-2 * rm-1 + ... = 0

m=cm;

m(row,col-4) = r1;
m(row,col-2) = -2.*r1 - 2.*r2 + Q.*h.^2;
m(row,col) = r3 + 4.*r2 + r1 - 2.*Q.*h.^2;
m(row,col+1) = h.^4;
m(row,col+2) = -2.*r3 - 2.*r2 + Q.*h.^2;
m(row,col+4) = r3;

return;



function m=add_mb(cm,row,col,r1,h)

% Add moment boundry condition to coefficient matrix
% Input: cm = coefficient matrix
%       row = line number of addition
%       col = column entry
%       r1 = EI of pile
%       h = length of segment
% Output: coefficients for r/h^2 (ym-1 - 2ym + ym+1) = Mm

m=cm;
```

```
m(row,col-2) = r1./(h.^2);
m(row,col) = -2.*r1./(h.^2);
m(row,col+2) = r1./(h.^2);

return;


function m=add_vb(cm,row,col,r1,h,Q)

% Add shear boundry condition to coefficient matrix
% Input: cm = coefficient matrix
%        row = line number of addition
%        col = column entry
%        r1 = EI of pile
%        h = length of segment
%        Q = axial load on pile
% Output: coefficients for R/3h^3 (ym-2 - 2ym-1 +2 ym+1 - ym+2) + Q/2h (ym-1
%                                  +ym+1) = V


m=cm;

m(row,col-4) = -r1./(2.*h.^3);
m(row,col-2) = 2.*r1./(2.*h.^3) - Q./(2.*h);
m(row,col+2) = -2.*r1./(2.*h.^3) + Q./(2.*h);
m(row,col+4) = r1./(2.*h.^3);

return;


function m=add_yb(cm,row,col)

% Add pile displacement boundry condition to coefficient matrix
% Input: cm = coefficient matrix
%        row = line number of addition
%        col = column entry
% Output: coefficients for yt = delta_pile

m=cm;

m(row,col) = 1;

return;
```

Vita

Alan Derek Tackie was born in Accra, Ghana, West Africa on November 18, 1975. He was raised in Accra the capital, where he attended Bishop Bowers from first to sixth grade. From there, he went to the Accra Academy and graduated from the 7-year program in 1995. He enrolled at College of the Ozarks in Branson, Missouri and received a B.S. in computer science and mathematics in 2001. He intends to pursue a PhD in the fall of 2004.