Integrating AutoCAD with Finite Element Code to Model Underground Construction

By

Timothy Y. Lai
and
Ronaldo I. Borja

CIFE Technical Report #111
January, 1998

STANFORD UNIVERSITY

If you would like to contact the authors please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

SUMMARY
CIFE TECHNICAL REPORT #111

Title: *Integrating AutoCAD with Finite Element Code to Model Underground Construction*

Authors: Timothy Y. Lai, Ronaldo I. Borja (PI)

Publication Date: January, 1998

Funding Sources:
- CIFE Seed Research
- Title of Research Project: *Integrating 4D CAD Model with Field Monitoring Program for Underground Construction*

1. Abstract:

The use of AutoCAD to allow easy interaction with a 3D nonlinear finite element (FE) excavation code DIG-DIRT written by the PI is presented. DIG-DIRT currently has many advanced features such as the element birth/death option to model the placement/removal of new/old materials, fluid flow option, consolidation with free-surface seepage, nonlinear soil behavior, and geometric nonlinearity options. However, DIG-DIRT has limited input/output capabilities. The current AutoCAD model allows the user to input the problem, run DIG-DIRT, and view the results (displacement, pore pressures, stresses/ strains) all from *within* AutoCAD. The current model also allows the user to view the output results at different time steps during the excavation sequence. This is essential since the process of excavation involves sequential removal of earth materials and installation of new structural members. Enhancing DIG-DIRT, by adding a user-friendly interface such as AutoCAD, will make the advanced features accessible to the design and construction community.

2. Subject:

This project links a nonlinear FE excavation code, written in Fortran, with the graphical software, AutoCAD. Failures in underground construction are not uncommon. Therefore, an excavation tool that can predict failures is essential. The FE excavation code (DIG-DIRT) capable of predicting such failures is already in place. A graphical user interface (GUI) that will facilitate the input and output of information to and from DIG-DIRT will enhance DIG-DIRT for the purpose of preventing underground construction failures.

3. Objectives/Benefits:

The objective of this project is to develop a graphical user interface so that the advanced features of DIG-DIRT will be made accessible to the design and construction community. The algorithms used in both the FE program and the GUI is now well established. Therefore, the value added from the project comes directly from the transfer of technology.

4. Methodology:

The software used to interface DIG-DIRT is AutoCAD. Specifically, the AutoLISP programming language within AutoCAD together with AutoCAD's other macro and customization features were used for the project.

5. Results:

From the project, a graphical user interface was developed for DIG-DIRT that allows the user to input the problem, run DIG-DIRT, and view the results (displacement, pore pressures, stresses/ strains) all from *within* AutoCAD. Software allowing the input and output of information from DIG-DIRT were created. All AutoLISP files needed to run the interface were generated. Finally, AutoCAD macro and custom files to supplement the AutoLISP files were written.

A 3-year long project sponsored by the National Science Foundation has resulted from this seed grant. The title of the project is: "Finite element analysis of strain localization in excavations," Principal Investigator: Ronaldo I. Borja; Project Period: 1 October 1997- 30 September 2000; Sponsoring Program: Geomechanical, Geotechnical and Geoenvironment Systems (NSF: Civil and Mechanical Systems).

6. Research Status:

All the essential tasks needed to run an underground construction simulation is in place. There are three main areas for future work. These are:

- Improvements to GUI: The way input and output of information to DIG-DIRT can still be improved. More output capabilities and a more object oriented input phase can be put in place.

- Field Monitoring/Performance Feedback: Enhancing DIG-DIRT to incorporate these features will allow the geotechnical engineer a way to analyze underground construction more accurately and in real time.

- Use in Classroom Teaching/Research: The newly developed GUI, together with DIG-DIRT, will be tested in the foundation engineering class of the PI next winter (1998). Feedback from students will be gathered and incorporated. Using DIG-DIRT with the GUI in research is ongoing.

Abstract

The use of AutoCAD to allow easy interaction with a 3D nonlinear finite element (FE) excavation code DIG-DIRT written by the PI is presented. DIG-DIRT currently has many advanced features such as the element birth/death option to model the placement/removal of new/old materials, fluid flow option, consolidation with free-surface seepage, nonlinear soil behavior, and geometric nonlinearity options. However, DIG-DIRT has limited input/output capabilities. The current AutoCAD model allows the user to input the problem, run DIG-DIRT, and view the results (displacement, pore pressures, stresses/ strains) all from *within* AutoCAD. The current model also allows the user to view the output results at different time steps during the excavation sequence. This is essential since the process of excavation involves sequential removal of earth materials and installation of new structural members. Enhancing DIG-DIRT, by adding a user-friendly interface such as AutoCAD, will make the advanced features accessible to the design and construction community.

Acknowledgments

This project was sponsored by a seed project award from the Center For Integrated Facilities Engineering (CIFE), Stanford University. A 3-year long project sponsored by the National Science Foundation has resulted from this seed grant. The title of the project is: "Finite element analysis of strain localization in excavations," Principal Investigator: Ronaldo I. Borja; Project Period: 1 October 1997- 30 September 2000; Sponsoring Program: Geomechanical, Geotechnical and Geoenvironment Systems (NSF: Civil and Mechanical Systems).

The authors would like to thank CIFE for the use of their computer equipment and software to make the project possible, and the Civil Engineering Department of Stanford University for granting access to the Advanced Computing Environment (ACE) Lab at Stanford University.

Table of Contents

1. Introduction

Construction of multi-story high-rise buildings with several floors of basements consists of excavating several meters of soil beneath the ground surface. The problem of underground construction and excavation involves sequential removal of earth materials and installation of new structural members. Construction of the substructure foundations generally gives construction contractors the most difficulties. The problem lies in the prediction of ground movements during the excavation process. Ground movements can be caused by several factors including, but not limited to, soil condition, ground water table, and adjacent building surcharges. This process affects not only the project but also all the surrounding buildings within the project area. Therefore, it is essential to be able to predict the ground movements and probable failures that may occur. With the sequential nature of the process, the geotechnical engineer must be able to design the braces, struts, and soil/rock anchors that must be used as the excavation proceeds.

The need for a system that allows the geotechnical engineer to analyze a given condition and formulate sound, real-time engineering decisions cannot be overstated. A properly analyzed and designed bracing system is a prerequisite for a successful construction and completion of a project. However, if improper calculations are performed, failure could occur, which could result in delays or even cancellation of the construction project. Figure 1 shows an active mode failure on the sidewalk that resulted from the construction of a 30-story high rise office building, called Chatham House, in the financial district of Manila, Philippines. The bracing system included standard steel I-beams braced against the core of the building, which was supplemented with rock anchors holding back the soldier piles. The real cause of failure was not determined but it was suspected that the rock anchors failed, leaving the standard bracing system to support the soil on its own. Luckily, the failure was not catastrophic and the standard bracing system was able to prevent total collapse (see Figs. 2 and 3). Construction of the project has been continued to completion (see Fig. 4). If the engineer had a calibrated numerical model, he/she could have performed "what if" scenarios as each layer of soil is

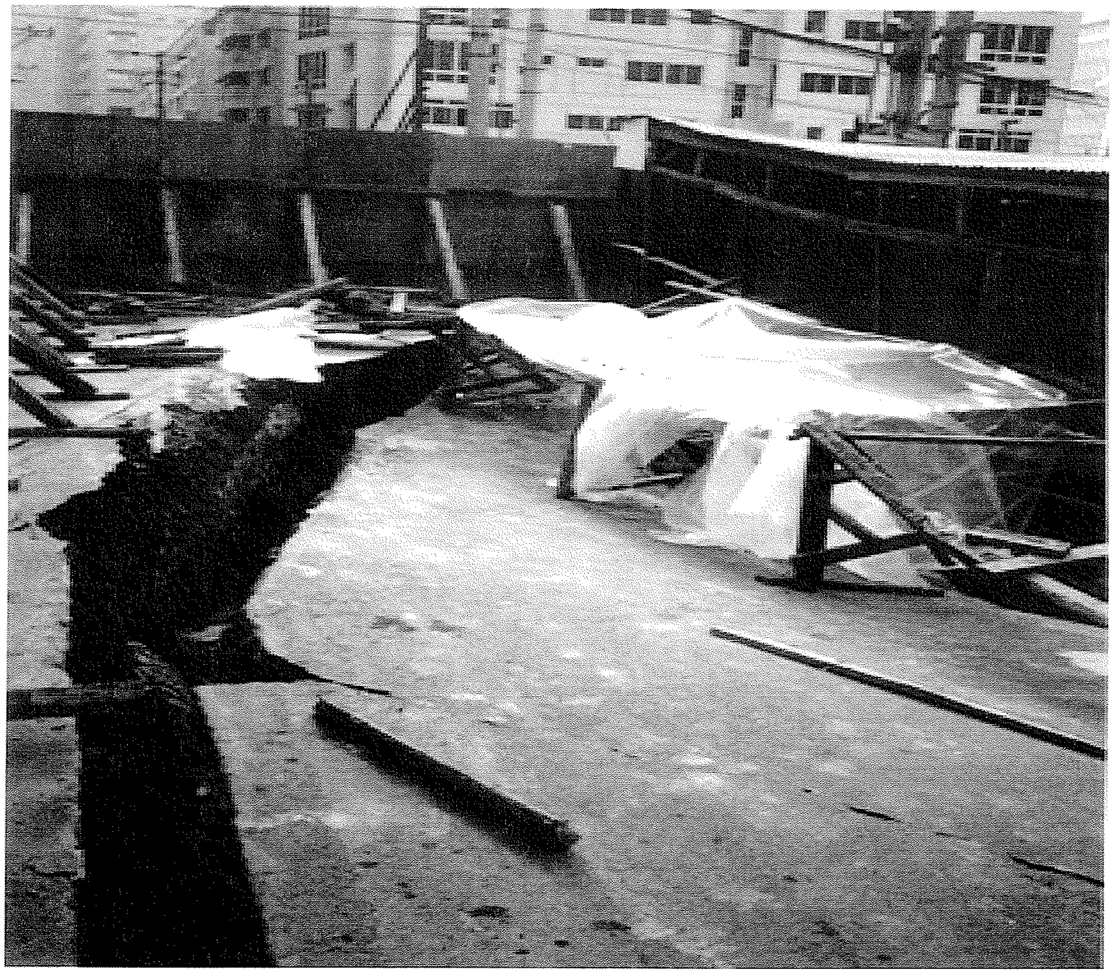


Figure 1. Active mode failure as seen from street level.

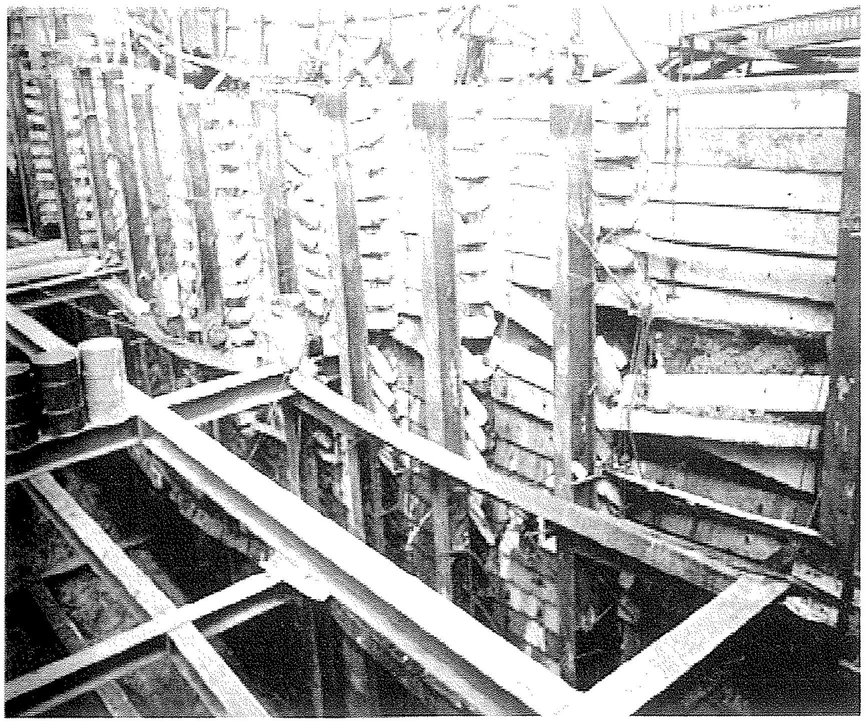


Figure 2. Steel bracing system.

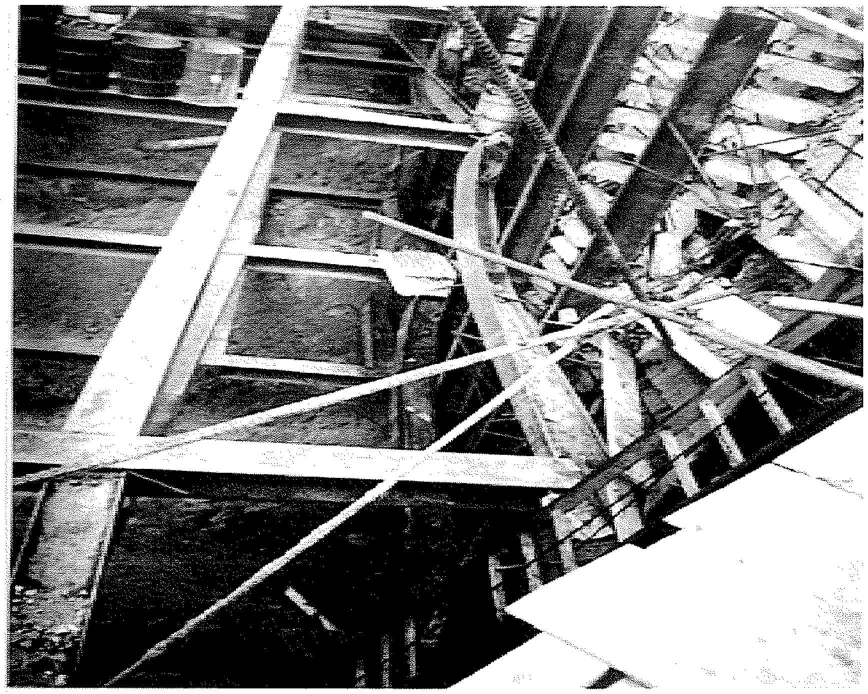


Figure 3. I-beams buckle but able to hold.

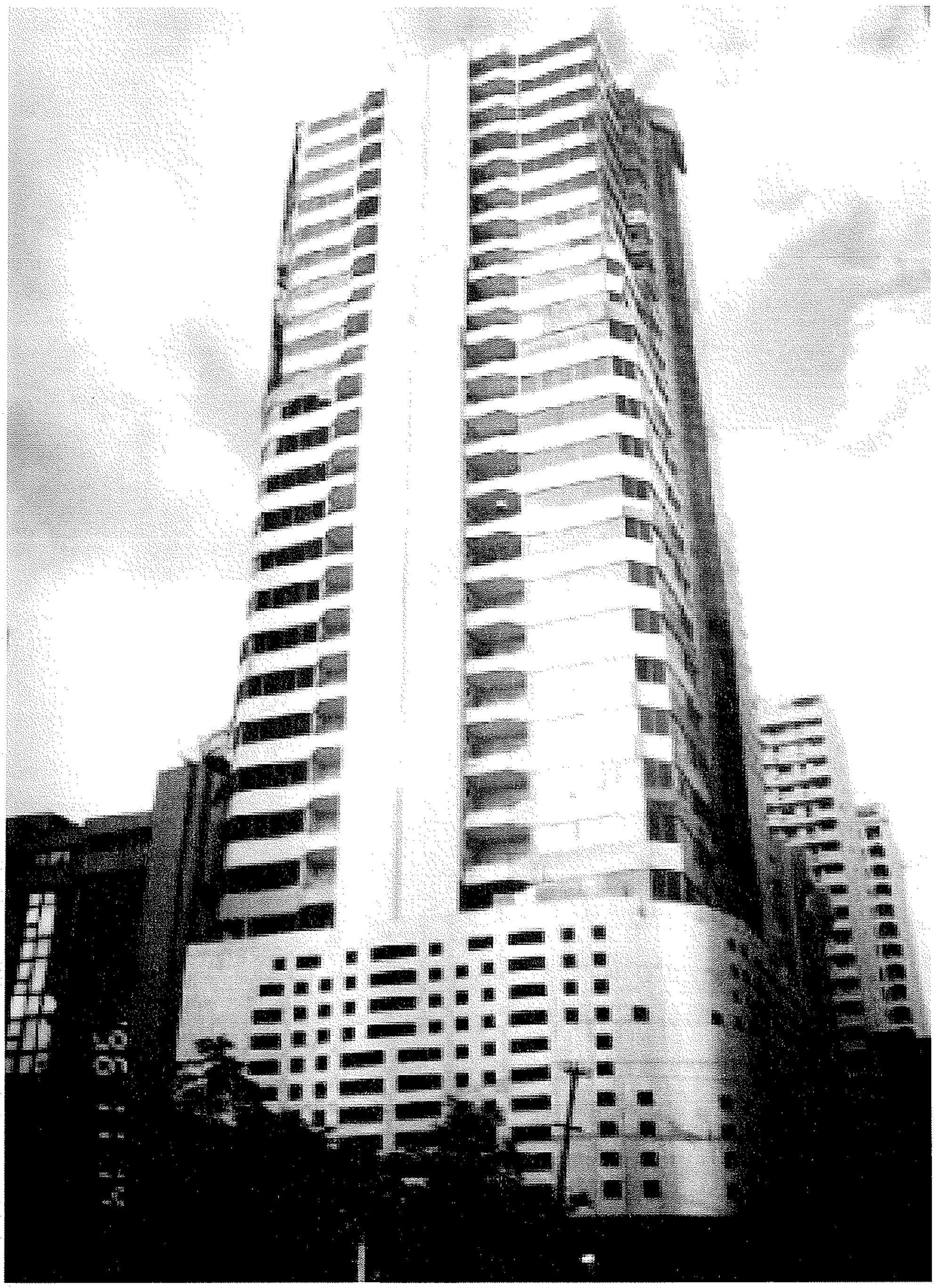


Figure 4. 30-story "Chatham Building"

removed, which could have prevented this failure from occurring. However, engineers normally have to rely on the overdesign/factor of safety of the braces and anchors and "observational approaches" for most underground construction. Savings on bracings and peace of mind on the design of the bracing system can easily be realized with a proper analysis tool in place.

As the previous case history has shown, the need for a soil-structure interaction computer model with excavation is needed. The PI has already developed such a program called DIG-DIRT. Its current capabilities include the element birth/death option to model the placement/removal of new/old materials, fluid flow option, consolidation with free-surface seepage, nonlinear soil behavior, and geometric nonlinearity options [10,11]. However, DIG-DIRT currently has limited input/output capabilities. Therefore, an interface capable of bringing the features of DIG-DIRT to the construction and design industry is essential.

The main software used in this project is AutoCAD [1,2,5]. Specifically, the AutoLISP programming language within AutoCAD is the main tool used to develop the interface. AutoCAD has numerous macro and custom features already developed through the years to supplement the creation of the interface [3,6]. DIG-DIRT and AutoCAD have now been linked in the computers of the Advanced Computing Environment (ACE) Lab at Stanford University. Both DIG-DIRT and AutoCAD now runs on a Windows-based Pentium Personal Computer. It is also available to run more complex problems on the Unix based Silicon Graphics Inc. (SGI's) in the Center for Integrated Facilities Engineering (CIFE) building at Stanford University. The current AutoCAD model allows the user to input the problem, run DIG-DIRT, and view the results (displacement, stresses, strains) all from within AutoCAD. A simulated excavation will be shown to highlight graphically the current capabilities of DIG-DIRT and the AutoCAD interface. For future work, topics include: (1) improving the capabilities of the interface with respect to its input and output capabilities, (2) incorporating field monitoring and performance feedback, and (3) testing DIG-DIRT with the interface in classroom and research settings.

2. Purpose/ Objectives

The purpose of this project is to develop an interface that allows the engineer to use the powerful features of DIG-DIRT. This transfer of technology gives the geotechnical engineer a tool to design the proper bracing system and to plan the sequence of construction. With the graphical interface available in real time, the geotechnical engineer will be able to see the deformations in the soil, and compute stress or strain concentrations and any areas that could cause potential failure. By viewing these results in a timely fashion, the engineer will be able to prevent failure through modification of the design and sequence of construction. The engineer can redesign the bracings or add the bracings sooner to prevent the possible failure that he/she "foresaw" while running the program in real time.

3. Impact/Significance

The main impact of this project lies in the transfer of technology. The main tool used is the AutoCAD AutoLISP language. This language has been in use since the early versions of AutoCAD. The value added from this project lies in the easier usage of DIG-DIRT. People who are unfamiliar with DIG-DIRT will now be able to take advantage of its features. Specifically, geotechnical and construction engineers will be able to take both programs, AutoCAD and DIG-DIRT, and run a simulated excavation sequence without having to know the inner workings of either program. This will then help prevent geotechnical failures such as shown previously. Another benefit from the project is having a visualization tool for design and analysis. Finally, it will also benefit the educational sector in classroom teaching and as a research tool. Students and researchers alike will be able to "see" the results and process information more efficiently. Students in a foundation engineering class will be able to view the deformations going on during an excavation instead of just viewing numbers. Researchers, on the other hand, will be able to show more effectively the failures or effects due to strain localization.

4. Current State of DIG-DIRT

DIG-DIRT is a nonlinear FE code written in FORTRAN. It has a number of advanced computational features that allow simulation of actual construction process. However, it lacks a powerful integration and user interface tool that facilitate easy input and output of data. Currently, DIG-DIRT reads in the geometry, soil properties, and other problem-related data from an input text file created by the user. It then analyses the problem and outputs the deformation, stresses, strains and other data in output text files (see Fig. 5).

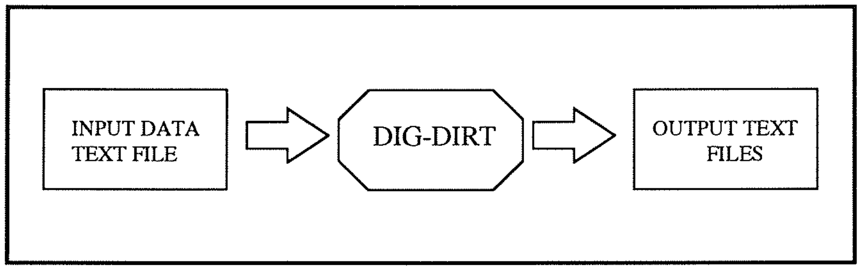


Figure 5. Flow chart of current state of input/output to DIG-DIRT.

4.1. Input

The current version of DIG-DIRT includes a help manual, which the user can use to create the input text file. A sample input file is shown in the appendix. A page of the help manual used to create the first two lines of the input text file is also included in the appendix. While it is not too difficult for a user familiar with DIG-DIRT to reproduce an input file, creating it is not convenient. Problems and inconveniences encountered with creating an input file include:

- Input file created by text editor *outside* of AutoCAD.
- Restrictive formats.
- Sequential input of information.
- No graphical way to check input mesh.

4.2. Output

The outputs from DIG-DIRT are also text files. Unlike the input files, there is no manual that guides the user. The user would have to be familiar with the code itself to know what are the output files generated and what information is contained in them. A partial sample of an output file on nodal deformed coordinates is shown in the appendix. Additional problems/inconveniences include:

- Deciphering text-based information is a slow process.
- No graphical results to view.
- Other software needed to view output files.

5. Methodology/Tools

This section deals with the bulk of the work in interfacing DIG-DIRT to make it more user friendly. The problems just highlighted has created the need to produce an interface that will create the input file automatically. Also, the interface must be able to read the output files and present it in a graphical setting. The flow chart for the input and output of information to and from DIG-DIRT (Fig. 6) shows that the user has no direct contact with DIG-DIRT and can treat it as a "black box" analysis tool. The user will then just have to master the interface.

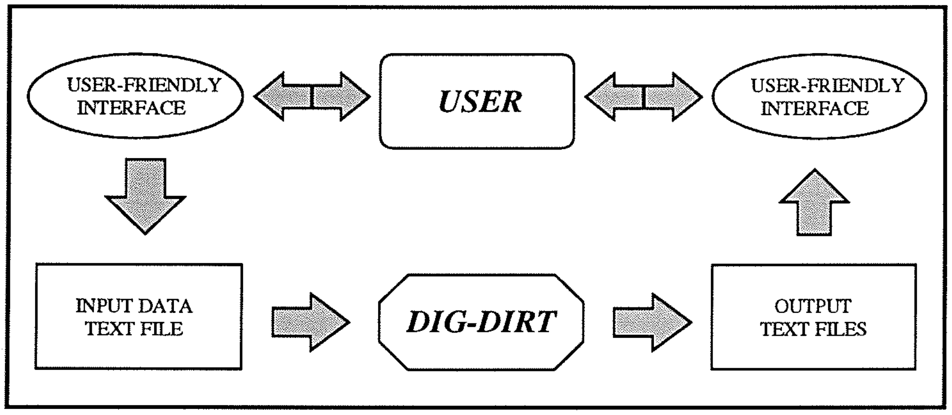


Figure 6. Flow Chart of input/output to DIG-DIRT with interface.

The software used to accomplish this user-friendly interface is AutoCAD. Within AutoCAD, the programming language used is AutoLISP. To take full advantage of AutoCAD, macro and custom features already developed and included in AutoCAD will be used to supplement AutoLISP. A more detailed flow of the AutoCAD interface and a summary of its current state and all the available features will be dealt with in the following sections.

5.1. AutoCAD

AutoCAD has many advantages and disadvantages. Both pros and cons will be discussed in the following sections. Also, other potential softwares that could have been used will be discussed. The discussion using other softwares is by no means complete because there are plenty of other softwares around that could have been used. However, a comparison with a few softwares is included for completeness.

5.1.1. Advantages

An obvious advantage in using AutoCAD is its graphics capabilities and its wide use in the design and construction industry. It has many built-in functions to draw and manipulate graphic objects [4]. Because they are already included, a developer or user can simply use the built-in functions without developing them from scratch. Examples of these functions include the line, shade, and 3d-mesh commands which were used for this project. Since most users need additional functions for their personal use, it also supports several different compiled programming languages. This allows a developer or user to create custom powerful commands and scripts. AutoCAD can "read" the compiled languages of the programming languages AutoLISP, C, and C++. After these programming languages are compiled, they become AutoLISP, ADS, and ARX files, respectively. Once loaded into AutoCAD, the user can immediately use the custom command that he/she developed.

Another reason for using AutoCAD is the support from CIFE. CIFE is an organization which gets its funding from outside companies in industry. One of its members is AutoDESK which produces the AutoCAD software. CIFE awarded a 1 year grant to fund the production of a graphical user interface (GUI) to support DIG-DIRT. The title of the grant awarded was "Integrating 4D CAD model with field monitoring program for underground construction." Because of the grant, a copy of AutoCAD version 13 for the PC was available for use and installed in the ACE laboratory. Also, access to use the more powerful SGI workstations to handle more complex problems was granted.

5.1.2. Drawbacks

There are several problems with using AutoCAD. While it is true that AutoCAD has many built-in graphics function, most are designed for the architect or draftsman in mind. Since AutoCAD was not initially designed to be used as an interface, the template to do the interface had to be done from scratch. Also, input and output is not as well-developed like in other softwares. Because of this, a FORTRAN program had to be developed to supplement the AutoCAD files. Finally, the main computations still have to be done with another program such as FORTRAN. While this is acceptable, it is still a drawback because we have to use a FORTRAN compiler when enhancing DIG-DIRT or the interface itself.

5.1.3. Other Programs

There are two other software programs which come into mind to do the interfacing. These include MATLAB and FORTRAN itself [7-9]. Both are good choices and "might" do the job adequately. Both programs have the computational power to do the calculations involved and were created with the engineer/mathematician/scientist in mind. The problem with MATLAB is that one needs the professional version to have the matrix capability and graphics environment to do the interfacing. Since this was not available, MATLAB was not considered viable. FORTRAN, on the other hand, is probably the most ideal tool since it can do the input, analysis, and output on its own without having to rely on another program. Recall, that DIG-DIRT is written in FORTRAN. Specifically, the code is written using FORTRAN 77 standards. The new Microsoft FORTRAN 90 compiler being used to run DIG-DIRT also has advance graphics capabilities to do the job. However, since the FORTRAN 90 compiler was just recently obtained, most of these features have not been looked into. Therefore, the viability of using FORTRAN 90 could not be ensured. Also, since the use of AutoCAD has already reached an advanced stage, it will continue to be used for the current project. In the future, the use of FORTRAN 90 will be pursued because of its inherent "in-house" advantages.

5.2. AutoLISP

The main programming language used is AutoLISP. AutoLISP, which is a scaled down version of Common LISP, is easy to use and takes full advantage of all the graphical commands in AutoCAD. As mentioned in a previous section, AutoCAD supports three different compiled languages: AutoLISP, AutoCAD Development System (ADS), and AutoCAD Runtime Extension (ARX). ADS and ARX (based on C and C++ respectively) are more robust and have more features. They can access facilities directly like the host operating system and generally can run more efficiently in terms of speed and memory usage. However, they are also harder to develop and maintain. Since the ADS and ARX programs are compiled on a C compiler, maintenance could be a problem. There are different C compilers which support and recognize different libraries (collection of built in commands and functions). Once a change is made in a code, recompilation of an ADS or ARX program has to be performed. This entails the use of a compatible C compiler or it won't recognize some of the functions. Also, ADS programs compiled for use in AutoCAD version 12 might not be able to run on the current version 13. AutoLISP, on the other hand does not have these problems since AutoCAD has an AutoLISP compiler built in. Therefore, it can just read an AutoLISP program, compile and then run it. Of course, the advantages of ADS and ARX can also be added to the interface at any time without affecting the AutoLISP environment already in place. The additional ADS and ARX code will simply supplement AutoLISP.

The following sections describe the AutoLISP programs created and their individual functions. All of the AutoLISP files have a .lsp file extension and have to be loaded into AutoCAD by the load command. One of the drawbacks to using AutoLISP is its poor formatting capabilities to handle input and output. Therefore, a fortran code called convert.f is also included in the discussion.

5.2.1. Main (.lsp) file

The main (.lsp) file is the file that loads up all the other AutoLISP files created for this project. It also automatically executes the command to create and define the different layers the graphical outputs will be in (see layer (.lsp) file in the next section). Finally it loads up the menu file that lets the user choose what specific action to take. This file actually doesn't do any of the calculations but serves as the controlling file for all the other AutoLISP and AutoCAD support and customized files. Figure 7 shows that the user loads the main (.lsp) file immediately after starting AutoCAD. Once main.lsp is loaded into AutoCAD, the user would be able to choose from a menu the functions he/she wants to use.

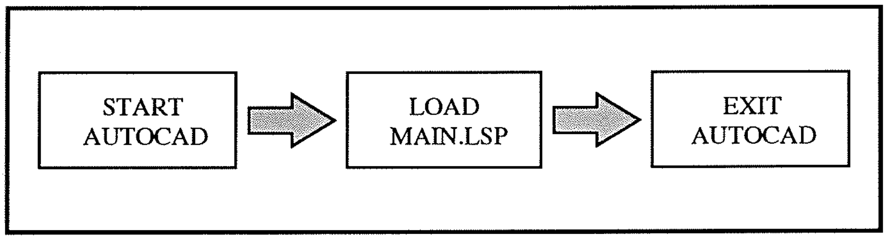


Figure 7. Flow chart of loading Main.LSP file .

The user does not need to load up any other files since everything needed was already loaded by the main (.lsp) file. To load the main.lsp file, the user simply selects "application" from the standard "tools" menu. A dialog box will appear requesting what AutoLISP file to load. The user needs to know where the main.lsp file is located (which drive/directory it is in). After selecting the file, the user clicks on the load button and the file will then be loaded. The user is in the "DIG-DIRT environment" once main (.lsp) is loaded. The user can go to the AutoCAD environment by choosing "exit" from the DIG-DIRT menu at any time during the excavation simulation. The user can then go back to the DIG-DIRT environment by simply loading up main (.lsp) again.

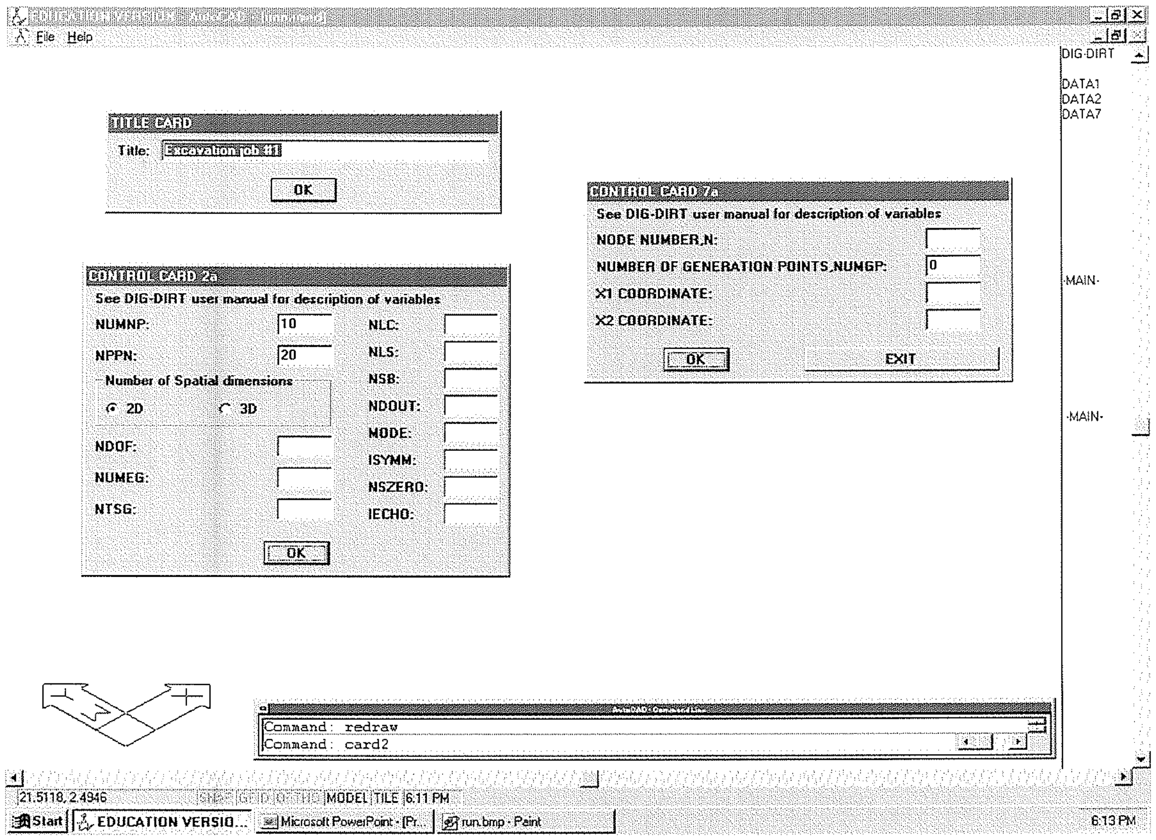
5.2.2. Layer (.lsp) file

This AutoLISP file creates the different layers and attributes the layers will possess. The attributes include the name, color, and linetype of a particular layer. An important feature that AutoCAD offers is the ability to draw and view objects in different layers. An example of this use is when we draw the original underformed mesh in a layer which is different from the current deformed mesh after a particular loading is applied. We can then view either mesh or both mesh at the same time by simply turning the layers on or off.

5.2.3. Dialog (.lsp) file

This file manages the dialog boxes used to get input information needed to run DIG-DIRT. The user inputs the problem by entering the information on dialog boxes (see Fig. 8). All the information entered in the dialog boxes are stored in different variables defined in the dialog.lsp file. The dialog.lsp file actually contains the commands to load up each of the custom-made dialog boxes (created in Dialog.dcl files talked about later). After the user enters all the information for that particular dialog box and enters "ok", the AutoLISP command within the dialog.lsp file closes the dialog boxes and stores the values in the respective variables. If a particular dialog box is re-opened, the values entered on the previous input will be shown and the user can make changes.

Overall, there are currently 17 "control cards" that the user has to go through and input information. The control cards are described in more detail in the DIG-DIRT user manual. In short, a control card is basically a collection of similar data grouped together. There are over 17 dialog boxes since some control cards have multiple dialog boxes.



TITLE CARD

Title: Excavation job #1

OK

CONTROL CARD 2a

See DIG-DIRT user manual for description of variables

NUMNP:	10	NLC:	
NPPN:	20	NLS:	
		NSB:	
		NDOUT:	
NDOF:		MODE:	
NUMEG:		ISYMM:	
NTSG:		NSZERO:	
		IECHO:	

Number of Spatial dimensions: [✓ 2D] 3D

OK

CONTROL CARD 7a

See DIG-DIRT user manual for description of variables

NODE NUMBER,N:	
NUMBER OF GENERATION POINTS,NUMGP:	0
X1 COORDINATE:	
X2 COORDINATE:	

OK EXIT

```
Command: redraw
Command: card2
```

```
21.5118, 2.4946          MODEL TILE 6:11 PM
```

Figure 8. Dialog boxes to get problem information from user.

5.2.4. Write (.lsp) and Convert (.f) file

After all the dialog boxes are filled out, the user then selects "write" from the menu and this AutoLISP file will write out the information on an output file called "data1". A part of "data1" can be seen in the appendix. As you can see, the formatting capabilities of AutoLISP is poor since it can only write out one value per line. Because of this, a FORTRAN program called convert.f needs to be implemented to convert "data1" into "data" which is in the format that is readable by DIG-DIRT. The following flow chart (Fig. 9) shows the initial input process to create the data file to be read by DIG-DIRT. Words in quotes are the actual words that represent the choices the user can select from the menu. With this system in place for the input phase, the problem of making formatting errors are eliminated. Also, logical errors will also be minimized since the user is guided in making the input file.

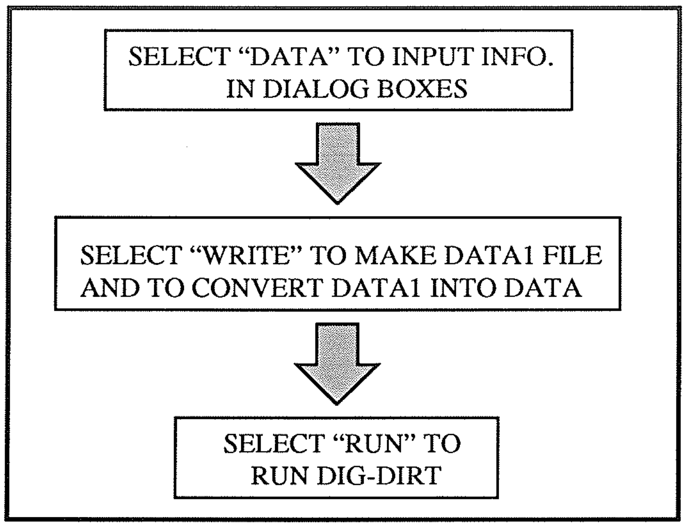

SELECT "DATA" TO INPUT INFO. IN DIALOG BOXES
SELECT "WRITE" TO MAKE DATA1 FILE AND TO CONVERT DATA1 INTO DATA
SELECT "RUN" TO RUN DIG-DIRT

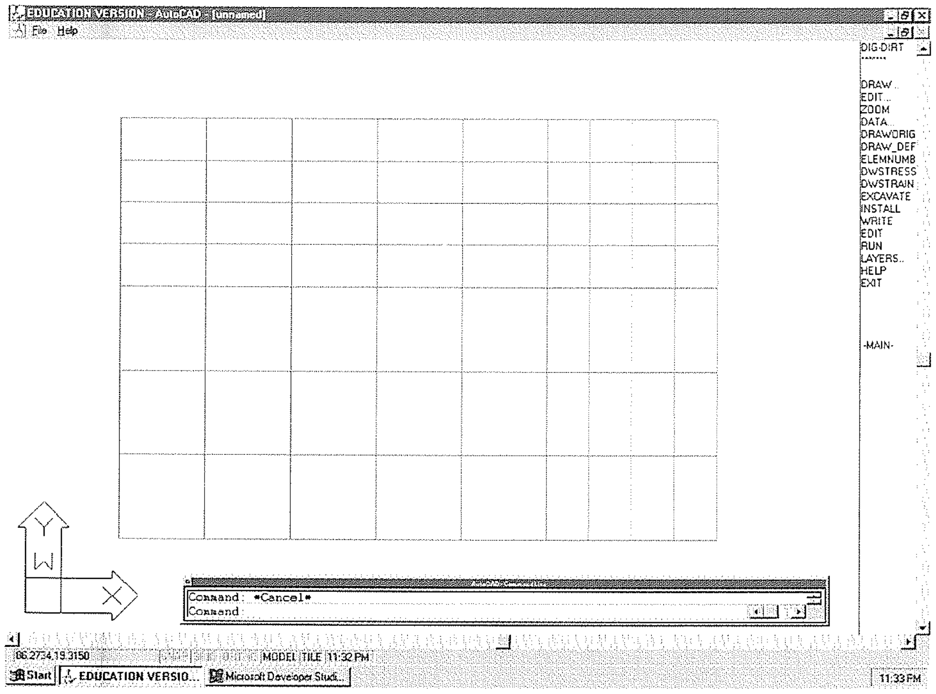
Figure 9. Flow chart for input phase of interface.

5.2.5. Getinfo(.lsp) file

The getinfo (.lsp) file is executed automatically and immediately after running DIG-DIRT. It reads in all the information from the output files of DIG-DIRT. There are three files that are specially outputted for the AutoCAD interface. These are "outdata," "out.ele," and "out.str." This process is convenient and efficient since all the output data needed by the interface is loaded into the AutoCAD environment at one time. Once this is accomplished, the user can now select from the menu which output to view.

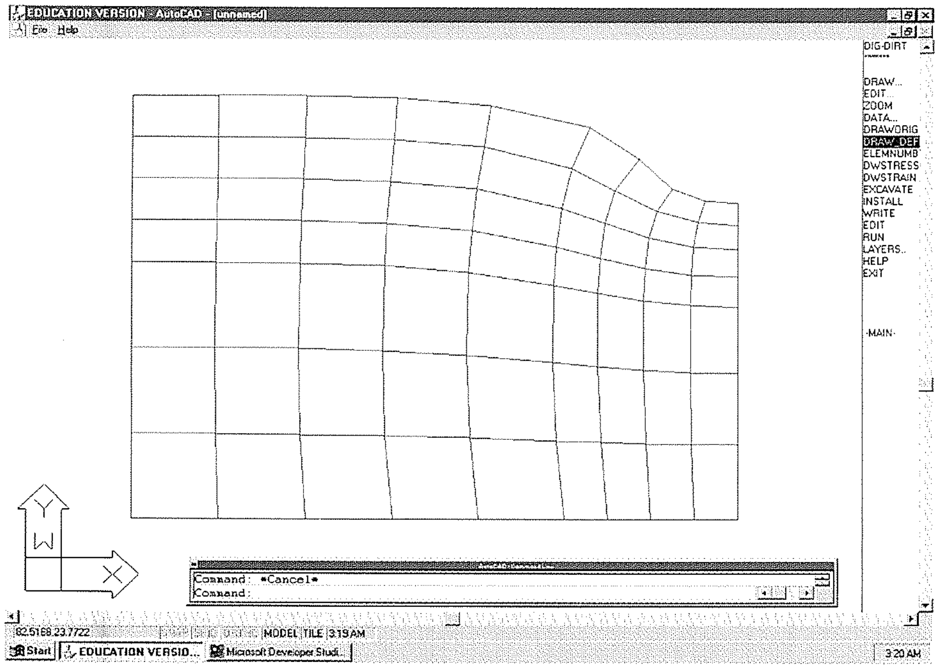
5.2.6. Draworig/Drawdef (.lsp) file

This file draws the undeformed (original) and deformed (current) mesh. A sample of this output is shown in the following figures (Figs. 10 & 11). The undeformed and deformed meshes are plotted on different layers to allow for differentiation. Here, we simply use the well-known AutoCAD command "line" to draw the individual elements for both meshes. During the excavation process, the deformed mesh will constantly be changing at different time steps. An added feature of the AutoCAD interface is that the user can specify the time step in which to view the deformed mesh. The user will be able to view other results (stresses/strains) at different time steps as well.



```
Command: *Cancel*
Command:
```

Figure 10. Undeformed (original) mesh.

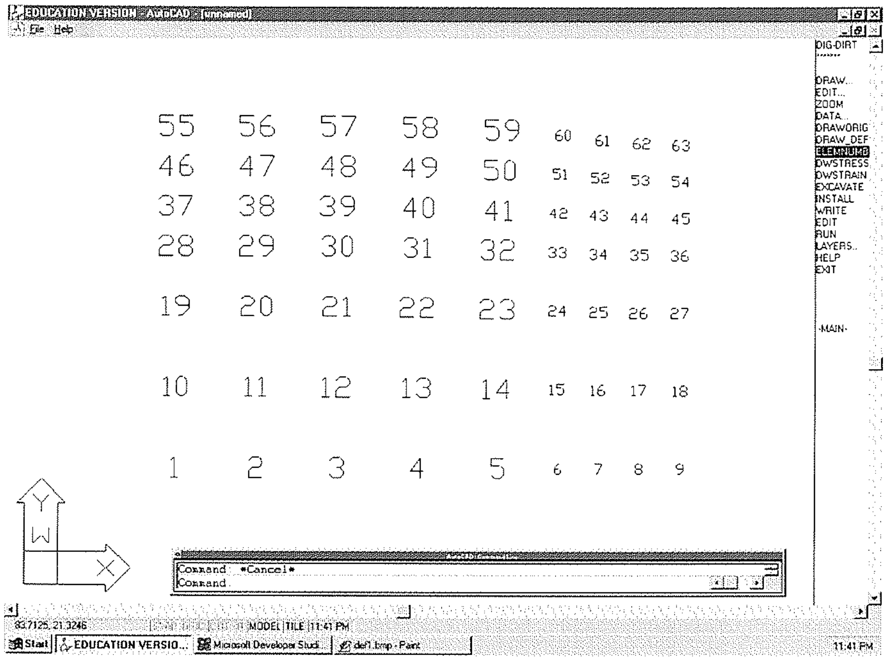


```
Command: *Cancel*
Command:
```

Figure 11. Deformed (current) mesh.

5.2.7. Elemnumb (.lsp) file

This AutoLISP file also plots the element numbers (see Fig. 12). The size of the element numbers depend on the size of the element itself since the height of the element is scaled to the length of the element sides. Also, the position of the element numbers are dependent on the position of the elements. Therefore, in a given mesh, the heights of the different element numbers will be different and their position will follow the deformed mesh. Also, the element numbers are plotted on a different layer from the undeformed/deformed meshes.



```
Command: *Cancel*
Command:
```

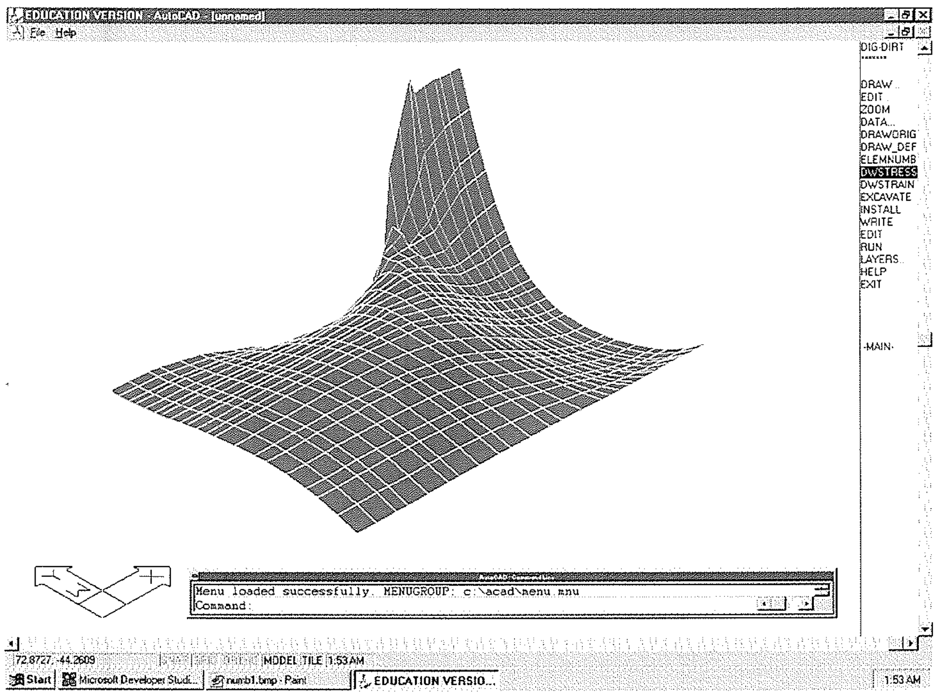
Figure 12. Plot of element numbering.

5.2.8. Drstress/Drstrain (.lsp) file

The Drstress (.lsp) file takes an output file created by DIG-DIRT and uses the information to plot the stresses. Currently, it plots the sigma11 stresses at the gauss points of each element. For background, gauss points are points on the elements where numerical integration takes place and where the stress/strain values are stored and located. Values of stress and strain at other points are then extracted from the values at the gauss points. The output file created by DIG-DIRT is called "out.str". A linear interpolation of the stresses is done by the built in AutoCAD command "3dmesh". If the user wants to get the exact value of the stress at a particular gauss point, he/she would simply click on the point and the attributes button on the standard AutoCAD toolbar. This view is further enhanced by taking an isometric point of view that will allow the user to visualize the stresses more clearly. For visibility, the stress plots are viewed in "sw isometric" view and the "shade" command was use to further enhance the plot (Fig. 13).

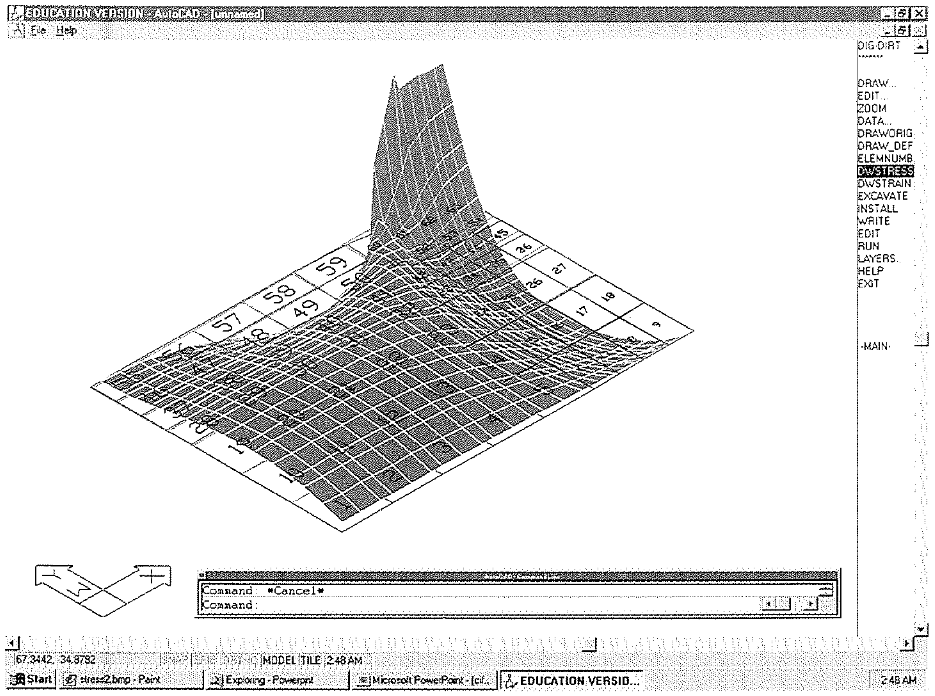
The strains are handled in a similar way. However, the octahedral shear strains are plotted. For background, the octahedral shear strains are the shear/tangential (as oppose to normal) component of the traction vector that is acting on a plane where the principal shear strains are acting.

Finally, by turning on all the layers, we can see the combine output (Fig. 14). The undeformed/deformed shape, element numbering, and a plot of stress/strain can then be seen together.



```
Menu loaded successfully. MENUGROUP: c:\acad\menu.mnu
Command:
```

Figure 13. Plot of stress at gauss points using "3dmesh" and "shade"command.



```
Command: *Cancel*
Command:
```

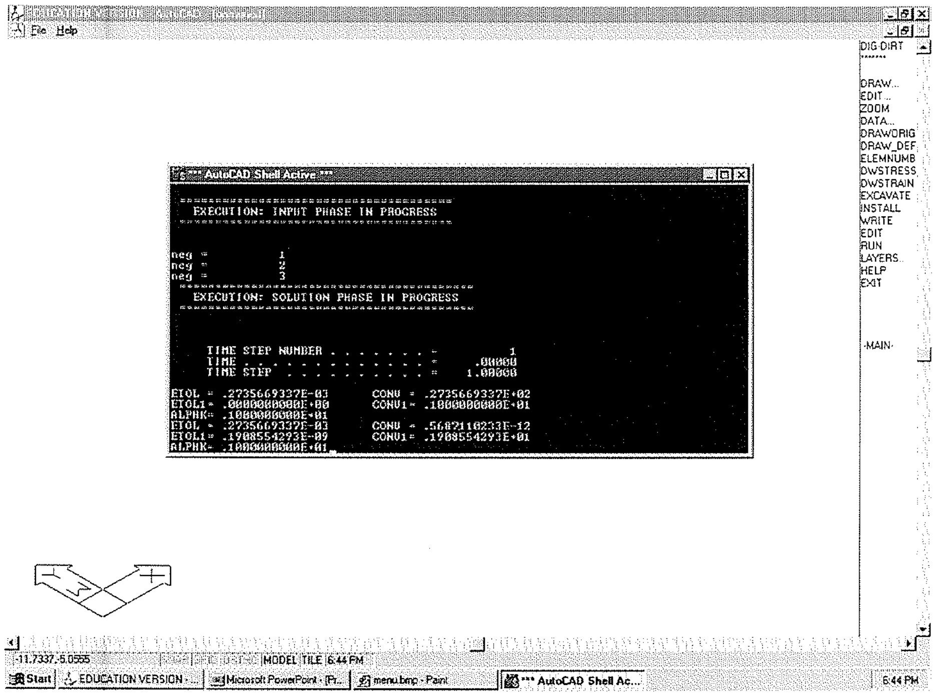
Figure 14. Combined plot of all output.

5.3. Macro and Custom Features of AutoCAD

Aside from the AutoLISP files created, other built in custom features of AutoCAD are used. These include the acad.pgp files, menu files and dialog.dcl files. These files supplement the AutoLISP programming language and will be described in the following sections.

5.3.1. Acad (.pgp) file

This file allows AutoCAD to recognize and execute DOS or UNIX commands from within AutoCAD. It also allows the creation of aliases for usual AutoCAD commands. A sample acad.pgp file is shown in the appendix. A drawback for this file is that a developer cannot create macros (custom series of commands) within this file. Therefore, all custom-made commands must be created using the AutoLISP command "defun" (short for define function). Once a user enters a command on the command line, AutoCAD executes the command if it recognizes it. If not, it looks at the acad.pgp file if it is an alias or system command. If it is still not defined in the acad.pgp file, it will issue an error command specifying it doesn't recognize it. The acad.pgp file is the one file we do not load into AutoCAD. Instead, AutoCAD will search for the first acad.pgp file in its files search path. Therefore, the custom acad.pgp file is located in the (c:r13\win) directory. Also, since it is not actually loaded into AutoCAD, the user/developer must use the "reinit" (reinitialize) command whenever a change in this file is made. The main use of this file is it allows the user to run DIG-DIRT from within AutoCAD (see Fig. 15). When the user selects run from the DIG-DIRT menu, the user is actually accessing the executable file of DIG-DIRT.



*** AutoCAD Shell Active ***

```
 ==========================================
   EXECUTION: INPUT PHASE IN PROGRESS
 ==========================================

neg =          1
neg =          2
neg =          3
 ============================================
   EXECUTION: SOLUTION PHASE IN PROGRESS
 ============================================


     TIME STEP NUMBER . . . . . . . . =          1
     TIME . . . . . . . . . . . . . . =      .00000
     TIME STEP  . . . . . . . . . . . =     1.00000

ETOL  = .2735669337E-03       CONV  = .2735669337E+02
ETOL1 = .0000000000E+00       CONV1 = .1000000000E+01
ALPHK = .1000000000E+01
ETOL  = .2735669337E-03       CONV  = .5682110233E-12
ETOL1 = .1908554293E-09       CONV1 = .1908554293E+01
ALPHK = .1000000000E+01
```

Figure 15. Running DIG-DIRT from within AutoCAD using the acad.pgp file.

5.3.2. Menu (.mnu) file

After the main.lsp file is loaded, a custom made file called menu (.mnu) file is loaded automatically. Once this screen menu is loaded, AutoCAD's default standard menus all disappear and is replaced by the menu developed in the menu.mnu file. Note that all screen menus might have to be "turned on" first. To do this, select "optionspreferences" and then the "system" tab from the standard AutoCAD toolbar. Then, check the "screen menu" button. The menu file creates a blank drawing screen with only the screen menu on the right side. The full screen will look something like figure 16.

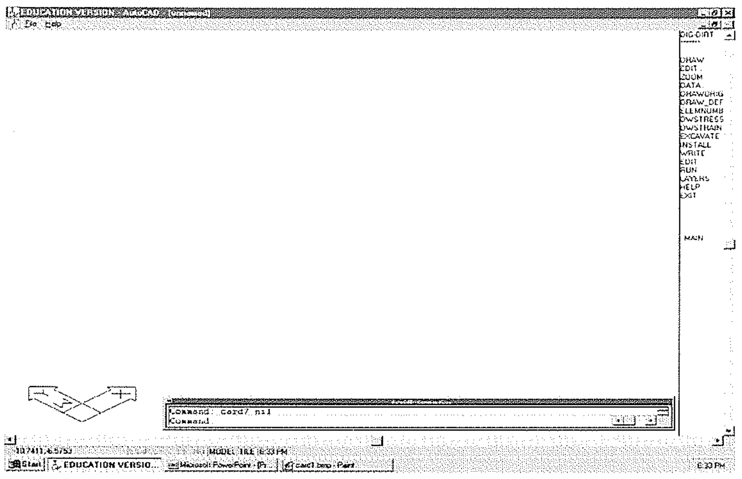


Figure 16. AutoCAD screen with loaded DIG-DIRT menu.

The options with three periods following the word will bring up another screen showing more options. The initial options above show some regular AutoCAD commands that can be included in the menu. When the user wants to return to the regular AutoCAD window, he/she can just select the exit command and the regular AutoCAD screen and pull-down menus will be loaded and the "DIG-DIRT" menu will disappear. The "DIG-DIRT" menu can easily be brought back by loading up main.lsp or using the AutoCAD command "menu" and selecting the menu.mnu file from the correct directory.

5.3.3. <u>Dialog(.dcl) files</u>

These files contain the design for all the dialog boxes in the interface. They are loaded by the dialog.lsp files. The default values for any item in the dialog boxes are also contained in the dialog (.dcl) files. Part of the dialog.dcl file is shown in the appendix. Each "card" in the current DIG-DIRT input manual is represented by a dialog box. The (.dcl) file, like the menu (.mnu) file, is again different from an AutoLISP (.lsp) file and has its own set of syntax.

5.4. Flow Chart of AutoCAD Interface

The GUI was designed for maximum flexibility. However, certain steps must still be followed for it to run correctly. Therefore, a flow chart is presented to describe the step-by-step procedures in using the AutoCAD interface (see Fig. 17). After loading and starting AutoCAD, the user first loads main.lsp. As mentioned before, this will load all the AutoLISP files and supporting files needed for the interface. This also brings up the DIG-DIRT menu. Once this is accomplished, the user is in the "DIG-DIRT" environment. At this point, the user can select from the DIG-DIRT menu what to execute. However, since the problem statement has not been specified and DIG-DIRT has not run yet, most of the options in the menu are still unavailable. The first step is therefore to create the input file into DIG-DIRT. The dialog boxes will guide the user to create the geometry, input the element material types, boundary conditions, and sequence of construction/excavation. Once the input file is created, the user then selects "run" and DIG-DIRT starts to analyze the problem.

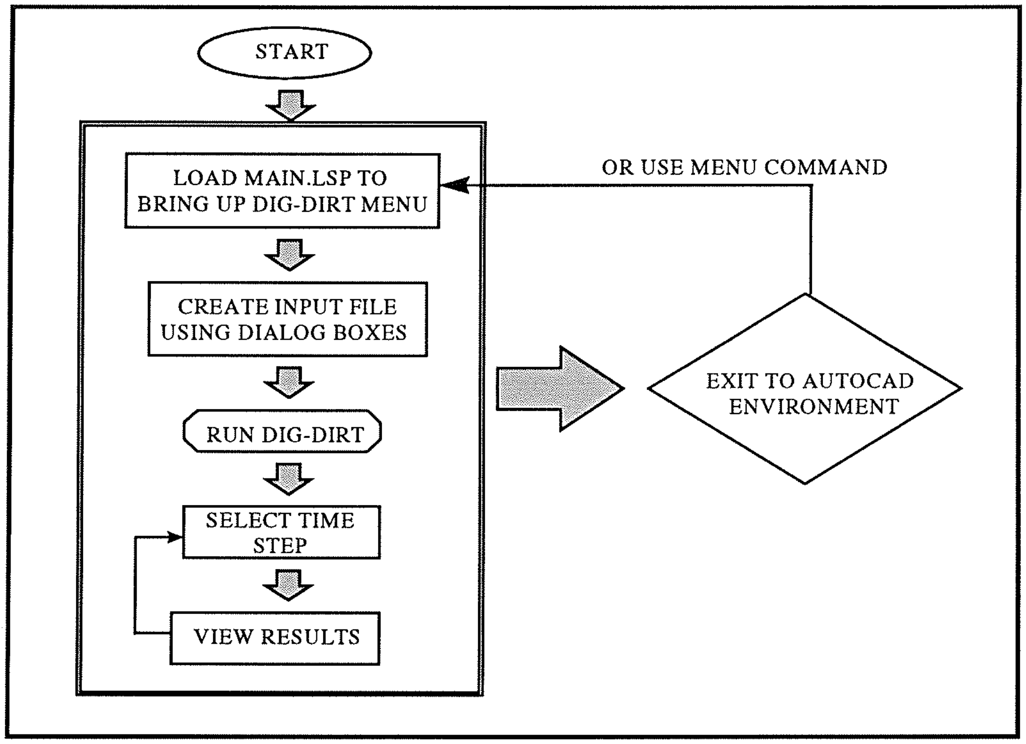

START
LOAD MAIN.LSP TO BRING UP DIG-DIRT MENU
OR USE MENU COMMAND
CREATE INPUT FILE USING DIALOG BOXES
RUN DIG-DIRT
EXIT TO AUTOCAD ENVIRONMENT
SELECT TIME STEP
VIEW RESULTS

Figure 17. Flow chart of AutoCAD interface.

When the analysis is complete, the user must now specify the time step in which to view the results. After this, the user can select to view the different results (undeformed/deformed mesh, stress/strain plot, element numbering). The user will be able to plot the results for another time step by simply selecting a different time step. Of course, the user has to delete the results of the previous time step so that it will not draw it over the current time step. Also, the user can go out of the DIG-DIRT environment and into the AutoCAD environment by simply selecting "exit" from the DIG-DIRT menu. The user can then go back to the DIG-DIRT environment by simply loading main.lsp again or using the menu command to load up the DIG-DIRT menu.

5.5. Summary of Current State of AutoCAD Interface

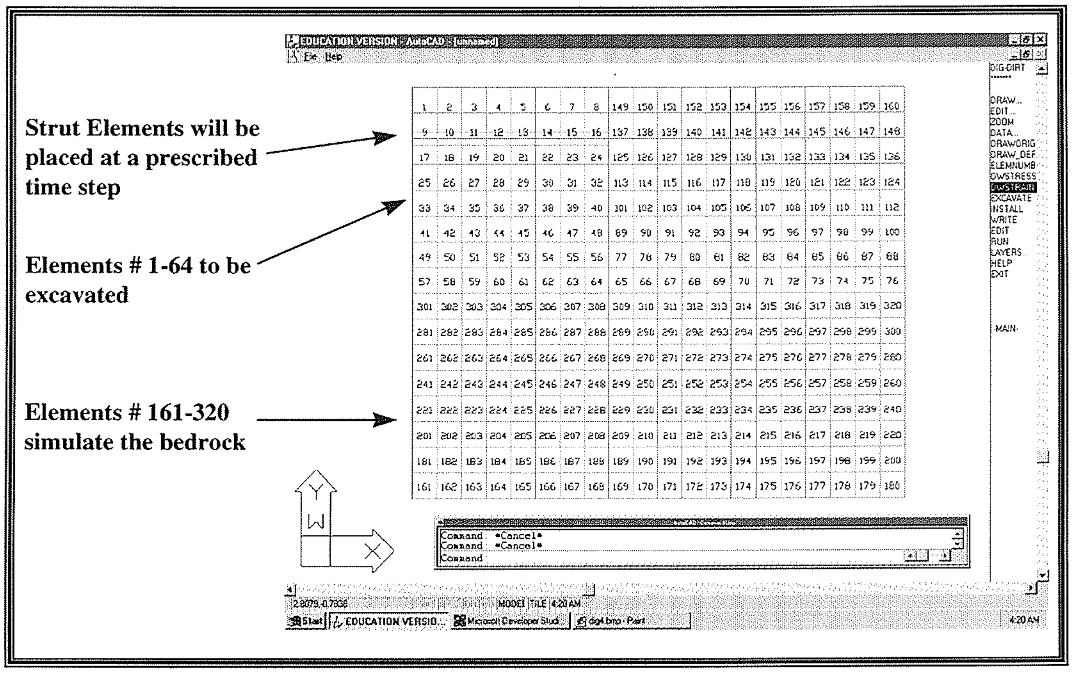
The current AutoCAD model allows the user to input the problem, run DIG-DIRT, and view the results (displacement, stresses, strains) all from within AutoCAD. The input phase consisting of the dialog boxes (dialog.lsp/.dcl files) and formatting (write.lsp/convert.f) are complete and working. The user can therefore reproduce and create the input file with no formatting errors. DIG-DIRT runs within AutoCAD via the acad.pgp file. Graphical results are in place showing the (deformed/undeformed meshes, stress/strain plots and element numbering). Also, the user can simply select a time step and view the results for that time step. The interface is able to process the information for different element groups at different time steps. This is essential especially for an excavation program where elements are born or die at different time steps.

6. Simulation of Underground Construction

To show the capabilities of the current GUI, a simulation of underground construction will be performed. First, the problem will be described and then graphical results will be presented. Since the input phase consists mainly of creating the input file for DIG-DIRT, the next sections will focus on the output phase where the GUI plays a more significant role.

6.1. Problem Description

An excavation simulation entails excavating "soil" elements and putting in "strut" elements at prescribed time steps. For the simulation, the original/undeformed mesh (Fig. 18) consists of three element groups. The first element group consists of the elements representing the soil (elements #1-160). From these, the first 64 elements will be excavated. The second element group consist of elements representing the bedrock (elements #161-320). The third element group consist of elements representing the struts which will be placed at prescribed time steps.



```
Command: *Cancel*
Command: *Cancel*
Command:
```

Figure 18. Plot of original mesh for simulated underground construction example.

The excavation sequence is as follows. There are five time steps for this simplified simulation. The first time step (time step #1) is used to allow the effect of gravity to take effect. Excavation occurs from time step #2 to time step #5 where 16 elements are excavated for each time step. Strut elements will be placed at time step #3 and will remain for the rest of the simulation. Also, note that the element numbering feature was used to keep track of the elements being excavated.

6.2. Graphical Output

A major output the user wants to view is the deformed mesh after undergoing excavation. In Fig. 19, we see the deformed meshes for time step #2 to time step #5. The magnitude of the displacement was multiplied five times for better visibility. The deformed shapes were obtained by selecting a particular time step and then plotting it through the "draw_def" option. If the user wants to view another time step, the first one would be erased and another time step would be selected. Of course, the user can also plot the results of one time step over another. As mentioned in a previous section, the interface keeps track of which elements are excavated or put in place so that the corresponding graphical results will be plotted accordingly.

Other results the user might want to see is the strain output. In Figs. 20 and 21, a plot of the strain output (magnified 100 times) is presented for time step #3 and time step #5. Notice that the views are different for each of the time steps. Here, we take advantage of the fact that the user can exit to the AutoCAD environment at any time and view the graphics from any point of view.

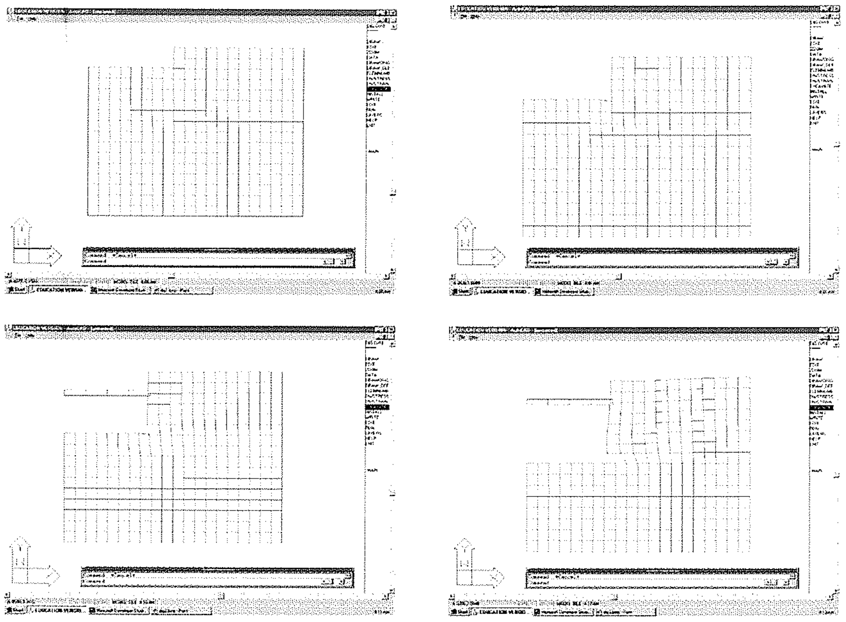
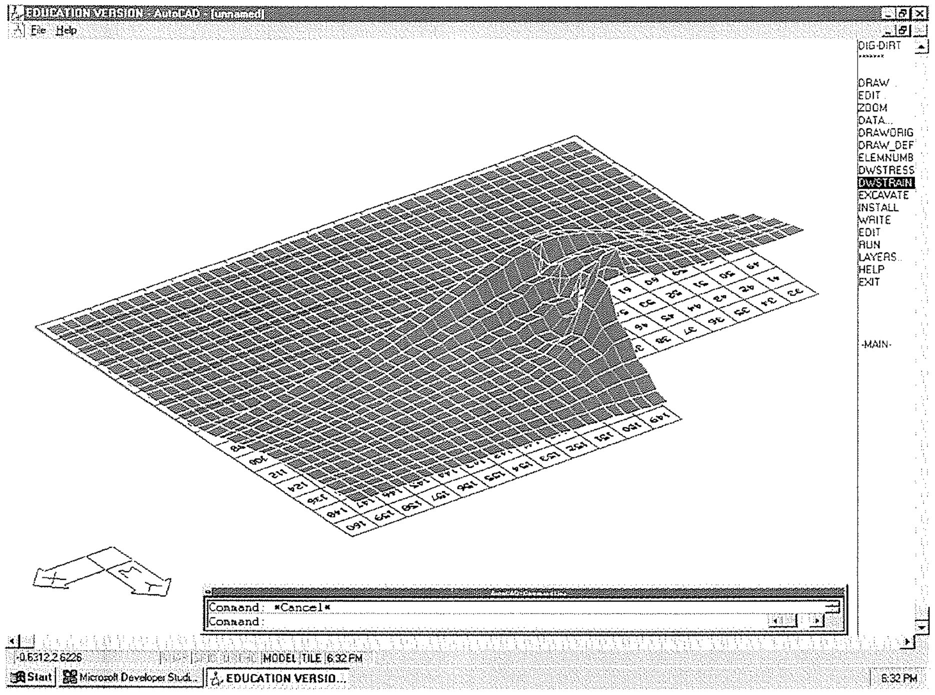
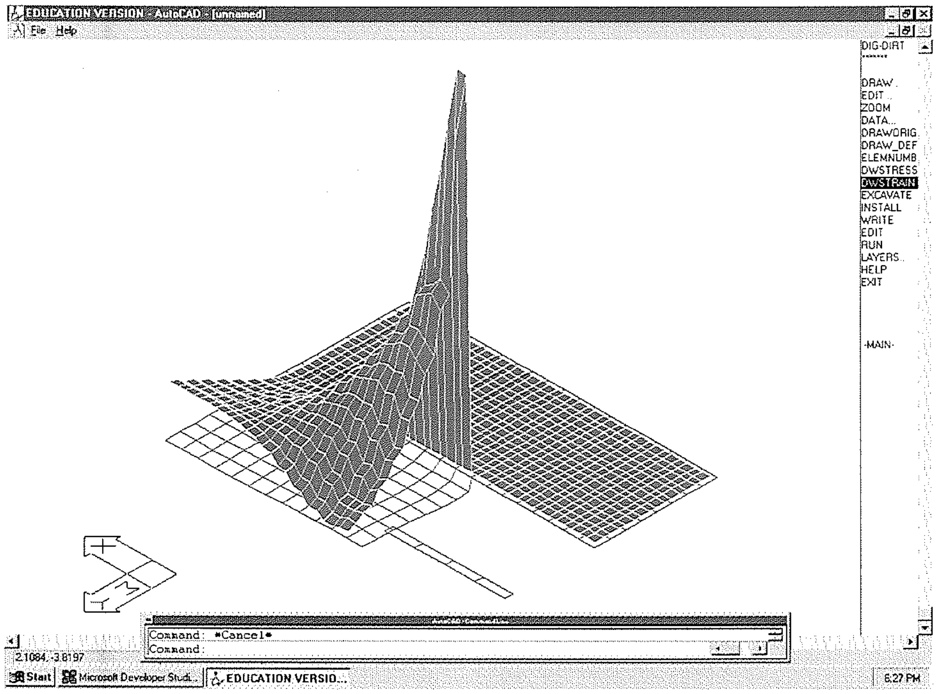


Figure 19. Plot of deformed mesh for time steps #2- #5.



```
Command: *Cancel*
Command:
```

Figure 20. Strain output at time step #3.



```
Command: *Cancel*
Command:
```

Figure 21. Strain output at time step #5.

7. Future Work

In the future, there are three main areas that can be dealt with. These are:

(1) improving the interface with regard to the way it inputs and outputs data to and from DIG-DIRT

(2) improving DIG-DIRT itself to handle field monitoring and performance feedback

(3) using the GUI with DIG-DIRT in class and research settings

7.1. Improvements to GUI

The current interface allows the user to input the problem, run DIG-DIRT, and view graphical results all from within AutoCAD. However, as with most programs, the GUI can constantly be improved. For the input phase, a more object oriented approach can be in place to get user information. For example, the geometry of the problem can be inputted by drawing the mesh instead of inputting the nodal points in dialog boxes. For the output phase, better plots of the stress and strain contours can be in place where a measure of the magnitude of the values would be in place. Also, use of object linking and embedding (OLE) can be taken advantage of to create tables and charts of some results like strut axial loads as the excavation proceeds.

7.2. Field Monitoring/Performance Feedback

The task of having a field monitoring program and performance feedback is natural since we want to test the capabilities of both the GUI and DIG-DIRT under real time, actual situations. This program (Fig. 22) would allow the engineer to make real time predictions of what will happen for the next time step. The engineer would start by using the information from the previous time step to do a numerical simulation and compare with actual results using a field monitoring program. Then, the engineer would use the information to calibrate the model for a prediction of the next time step. Large errors would mean that the model is bad. However, good results would mean that the model is capable of predicting future results. In this way, the engineer can do "what if" scenarios such as deciding to put a strut or not.

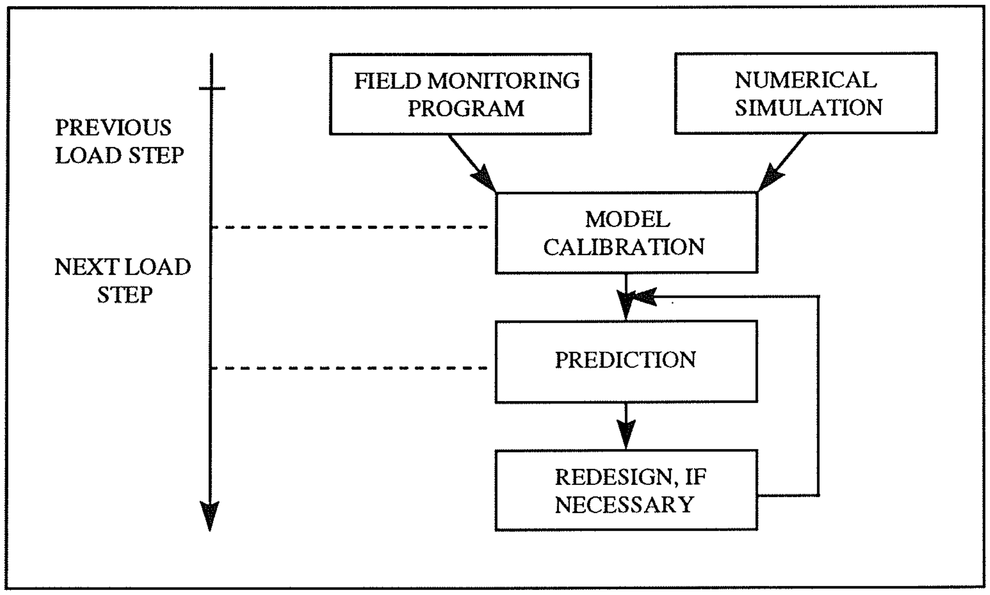


Figure 22. Incorporating field monitoring and performance feedback

7.3. Use in Classroom Teaching/Research

One important impact of the project is the value it has in the educational field. For teaching purposes, the tool realized from this project will help the student visualize the deformations and stress/strains the soil structure undergoes during underground structure construction. The GUI and DIG-DIRT will be tested next winter (1998) during the PI's foundation engineering class. Student reactions and suggestions will be obtained to assess its educational value. Also, ongoing research that will improve the capabilities of DIG-DIRT will be enhanced by having a user-friendly DIG-DIRT.

APPENDIX A
Additional Figures

Sample Input File

```
Excavation problem
  375    0    2    2    3    1    1    1    1    0    1    1    1    0
    1    0    0   15    0
    1    0       1.0
    1    9       1.0
    0    0    0    0    0

    1    4       0.0        0.0
   21    0       5.0        0.0
  357    0       5.0        4.0
  337    0       0.0        4.0
   20    1   16   21
  358    2       0.0     3.4375
  366    0       2.0     3.4375
    8    1    0    0
  367    2       0.0     3.5625
  375    0       2.0     3.5625
    8    1    0    0

    1    1    1    1
   21    0    1    1
   22   21    1    0
  337    0    1    0
   42   21    1    0
  357    0    1    0
  358    0    1    0
  367    0    1    0



       0.0        0.0
       1.0        1.0
    4  160    6    1    1    0    0    1    2    0    0
    2    2    8    0    1    1
    2    6
     1.0e3      0.45        0.0       15.5        0.0         0.0
   1.0e-3     1.0e-3        0.0
      20.0       0.0        0.0       -1.0
    1         316  317  338  337    1
    8    1    1    8    8  -21
   65         177  178  199  198    1
   12    1    1    8   12   21

    4  160    6    1    1    0    0    2    2    0    0
    0    0    0    0    1    1
    2    6
     1.0e5       0.1        0.0      1.0e5        0.0         0.0
   1.0e-3     1.0e-3        0.0
      30.0       0.0        0.0       -1.0
```

Sample Output File from DIG-DIRT

```
     0      1   .000
(         .197         3.947         2.045)
(         .197         3.803         2.045)
(         .053         3.947         2.045)
(         .053         3.803         2.045)
(         .447         3.947         2.045)
(         .447         3.803         2.045)
(         .303         3.947         2.045)
(         .303         3.803         2.045)
(         .697         3.947         2.045)
(         .697         3.803         2.045)
(         .553         3.947         2.045)
(         .553         3.803         2.045)
(         .947         3.947         2.045)
(         .947         3.803         2.045)
(         .803         3.947         2.045)
(         .803         3.803         2.045)
(        1.197         3.947         2.045)
(        1.197         3.803         2.045)
(        1.053         3.947         2.045)
(        1.053         3.803         2.045)
(        1.447         3.947         2.045)
(        1.447         3.803         2.045)
(        1.303         3.947         2.045)
(        1.303         3.803         2.045)
(        1.697         3.947         2.045)
(        1.697         3.803         2.045)
(        1.553         3.947         2.045)
(        1.553         3.803         2.045)
(        1.947         3.947         2.045)
(        1.947         3.803         2.045)
(        1.803         3.947         2.045)
(        1.803         3.803         2.045)
(         .197         3.697         6.136)
(         .197         3.553         6.136)
(         .053         3.697         6.136)
(         .053         3.553         6.136)
(         .447         3.697         6.136)
(         .447         3.553         6.136)
(         .303         3.697         6.136)
(         .303         3.553         6.136)
(         .697         3.697         6.136)
(         .697         3.553         6.136)
(         .553         3.697         6.136)
(         .553         3.553         6.136)
(         .947         3.697         6.136)
(         .947         3.553         6.136)
(         .803         3.697         6.136)
(         .803         3.553         6.136)
(        1.197         3.697         6.136)
(        1.197         3.553         6.136)
(        1.053         3.697         6.136)
(        1.053         3.553         6.136)
```

Sample "DATA1" File

```
Excavation problem
375
0
2
2
3
1
1
1
1
0
1
1
1
0
1
0
0

0
1


1



0
0
0
0
0
1
0
0

0


0
0
0
2




1
0
4

5
1
1
```

DIG-DIRT

USER'S MANUAL

1. TITLE CARD (20A4)

Columns	Variable	Description
1–80	TITLE(20)	Job title for output heading

2a. CONTROL CARD (14I5)

Columns	Variable	Description
1–5	NUMNP	Number of nodal points; if = 0, program stops
6–10	NPPN	Number of pore pressure nodes; if > 0, mixed formulation

11–15	NSD	Number of spatial dimensions;
		if = 2, 2D analysis;
		if = 3, 3D analysis; consolidation not available

16–20	NDOF	Number of degrees of freedom per node; input 2 if NPPN=0, input 3 if NPPN>0
21–25	NUMEG	Number of element groups; ≥ 1
26–30	NTSG	Number of time step groups; ≥ 1
31–35	NLC	Number of load cases; if = 0, set internally to 1
36–40	NLS	Number of load steps; ≥ 0
41–45	NSB	Number of time steps between spatial printout; ≥ 1
46–50	NDOUT	Number of displacement/pore pressure output histories; ≥ 0
51–55	MODE	Execution mode; = 0, data check only = 1, execution
56–60	ISYMM	Symmetry parameter; = 0, symmetric matrix = 1, nonsymmetric matrix
61–65	NSZERO	Step number at which displacements are to be reset to zero (NS=0 during first step)
66–70	IECHO	Input echo print mode; = 0, echo-print input data = 1, do not echo print input data

APPENDIX B
Sample AutoLISP Files

MAIN.LSP

```
;This file, main.lsp, is the main file for the AutoCadd project
;
;Load LAYER.lsp :
;Describes the information for
;the deform and current mesh layer,element number,
;stress, and current layer
 (load "c:\\acad\\layer.lsp")
 (c:layit)
;
;Load GETINFO.lsp : AutoLISP program that reads all information
;from DIG-DIRT and stores them as variables to be used by all
;AutoLISP functions
;
 (load "c:\\acad\\getinfo.lsp")
; (c:getinfo)
;
;Load other AutoLISP files:
;DRAWORIG draws the original mesh
;DRAWDEF draws the current/deformed mesh
;DRSTRESS draws the stress "mountain"
;PATCH supplements drstress
;DIALOG gives information on dialog boxes
;ELEMNUMB draws the element numbers
;WRITE writes input info into file which is then transformed
;by a FORTRAN file to format readable by DIG-DIRT
;
 (load "c:\\acad\\draworig.lsp")
 (load "c:\\acad\\drawdef.lsp")
 (load "c:\\acad\\drstress.lsp")
 (load "c:\\acad\\patch.lsp")
 (load "c:\\acad\\dialog.lsp")
 (load "c:\\acad\\elemnumb.lsp")
 (load "c:\\acad\\write.lsp")
;
;Load menu menu.mnu
;
 (command "menu" "c:\\acad\\menu.mnu")
```

GETINFO.LSP

```
;GETINFO.LSP
;AutoLISP file that gets all the info needed from output files created
;by FORTRAN code DIG-DIRT. Specifically, it reads the info from the
;output files called out.ele, out.str and outdata


(defun c:getinfo ()
  (setq f (open "c:\\r13\\win\\outdata" "r"))
  (setq g (open "c:\\r13\\win\\out.ele" "r"))
  (setq h (open "c:\\r13\\win\\out.str" "r"))
  (setq nemgr (atoi (read-line g)))
  (setq ntstep (atoi (read-line g)))
;
;load element group information into list "elemgrp"
;
   (setq test 1)
   (setq elemgrp '((0 0 0 0)))
   (while (<= test nemgr)
      (setq elemgrp (cons (READ(read-line g)) elemgrp))
      (setq test (1+ test))
   )
         (setq elemgrp (reverse elemgrp))

; read total number of elements and number of nodalpts

  (setq nnpts (atoi (read-line g)))
  (setq nelem (atoi (read-line g)))


;
;load element connectivity data into list "conn"
;
   (setq test 1)
   (setq conn '((0 0 0 0 0)))
   (while (<= test nelem)
      (setq conn (cons (READ(read-line f)) conn))
      (setq test (1+ test))
   )
         (setq conn (reverse conn))
;
;load coordinates into list "pt"
;
   (setq test 1)
   (setq pt '((0.0 0.0)))
   (while (<= test (* nnpts (1+ ntstep)))
      (setq pt (cons (READ(read-line f)) pt))
      (setq test (1+ test))
    )
         (setq pt (reverse pt))

;
;***************load stress "coordinates" into list "strpt"*********

;   gaussnumb is total # of gauss points for all times steps.
;   templist is temporary list for each element group information

   (setq gaussnumb 0)
        (setq testa 1)
   (while (<= testa nemgr)
          (setq templist (nth testa elemgrp))

;    ****** CHECK IF ELEM BIRTH/DEATH NOT USED ***********
          (if (= (nth 1 (nth testa elemgrp)) 0)
                (progn
                  (setq addon (* (nth 0 templist) (nth 4 templist)
                           (nth 4 templist) ntstep))
                  (setq gaussnumb (+ gaussnumb addon))
                )
          )
;    ****** CHECK IF ELEM BORN **************************
          (if (= (nth 1 (nth testa elemgrp)) 1)
                (progn
                  (setq addon (* (1+ (- ntstep (nth 2 templist)))
                                    (nth 3 templist)(nth 4 templist)
                                    (nth 4 templist)))

               (setq gaussnumb (+ gaussnumb addon))
                )
          )

;    ****** CHECK IF ELEM DIE **************************
;    tempb will be number of total elem died
;    temp will tempb* integration pts
;    totposs is total number of elem*(numb. of gauss pts)*ntstep
;
          (if (= (nth 1 (nth testa elemgrp)) 2)
                (progn
                  (setq count 1)
                  (setq tempb 0)
                  (setq tempa 1)
                      (while (<= tempa (1+ (- ntstep (nth 2 templist))))
                              (setq tempb (+ tempb (* count (nth 3 templist))))
                                   (setq count (1+ count))
                                   (setq tempa (1+ tempa))
                           )
                           (setq temp (* tempb (nth 4 templist)
                                    (nth 4 templist)))
                           (setq totposs (* (nth 0 templist) ntstep
                                      (nth 4 templist)(nth 4 templist)))
                           (setq addon (- totposs temp))
                 (setq gaussnumb (+ gaussnumb addon))
                   )

            )
            (setq testa (1+ testa))
           )
; After calculating gaussnumb can proceed to load strpt

   (setq test 1)
   (setq strpt '((0.0 0.0 0.0)))
   (while (<= test gaussnumb)
     (setq strpt (cons (READ(read-line h)) strpt))
     (setq test (1+ test))
   )
        (setq strpt (reverse strpt))
;
;close data files
;
  (close f)
  (close g)
  (close h)
;******************************
)
```

DRAWORIG.LSP

```
;Autolisp program to draw original mesh.
;
 (defun c:drawmesh ()
;
;Draw original mesh
;
     (command "layer" "set" "original" "")
     (setq test 1)
     (while (<= test nelem)
       (command "line" (nth (nth 1 (nth test conn)) pt)
                       (nth (nth 2 (nth test conn)) pt)
                       (nth (nth 3 (nth test conn)) pt)
                       (nth (nth 4 (nth test conn)) pt)
                       "C"
       )
       (setq test (1+ test))
     )
   (command "layer" "set" "new""""")
   (command "zoom" "all")
   (command "zoom" "vmax")

)
```

LAYER.LSP

```
;LAYER.LSP
;Autolisp file to describe the "stress," "original," "current," "elemnumb,"  and "new" layers

 (defun c:layit ()
  (command "layer""new""stress" "")
  (command "layer""new""original" "")
  (command "layer""new""current" "")
  (command "layer""new""elemnumb" "")
  (command "layer""new""new" "")
  (command "layer""color""green""stress""")
  (command "layer""color""blue""original""")
  (command "layer""color""red""current""")
  (command "layer""color""white""elemnumb" "")
  (command "layer""color""green""new""")
)
```

APPENDIX C
Sample Fortran File

CONVERT.F

```
      PROGRAM Rdconv
c
c     Program to read in file "data1", which was outputed from
c     AutoCadd, and to write file "data2", which will be used
c     by DIG-DIRT
c
c     Free formatting to read in from "data1" since reading
c     one value per line
c
c     Symbols correspond to DIG-DIRT user manual
c
c     Numbers after variables represent either direction number
c     or card numbers: Ex Variable "Inc1" in card 17 will
c     be referred to as "Inc117"

     IMPLICIT DOUBLE PRECISION(A-H,O-Z)
        CHARACTER*80 title,blank
        INTEGER I2a(14),I2b(5),I5(5),irec,nslms
        INTEGER, ALLOCATABLE:: N(:),Numgp(:),M(:),Mgen(:)
        INTEGER, ALLOCATABLE:: N9(:),Numgp9(:),M9(:),Mgen9(:)
        INTEGER, ALLOCATABLE:: N10(:),Numgp10(:),M10(:),Mgen10(:)
        INTEGER, ALLOCATABLE:: n4(:),nts(:),n8(:),ng(:)
        INTEGER, ALLOCATABLE:: id1(:),id2(:),id3(:)
        INTEGER, ALLOCATABLE:: Ninc1(:),Ninc2(:),Inc1(:),Inc2(:)
        INTEGER, ALLOCATABLE:: Ninc19(:),Ninc29(:),Inc19(:),Inc29(:)
        INTEGER, ALLOCATABLE:: Ninc110(:),Ninc210(:),Inc110(:),Inc210(:)
        INTEGER, ALLOCATABLE:: Npar1(:),Npar2(:),Npar3(:),Npar4(:)
        INTEGER, ALLOCATABLE:: Npar5(:),Npar6(:),Npar7(:),Npar8(:)
        INTEGER, ALLOCATABLE:: Npar9(:),Npar10(:),Npar11(:),Npar12(:)
        INTEGER, ALLOCATABLE:: Npar13(:),Npar14(:),Npar15(:),Npar16(:)
        INTEGER, ALLOCATABLE:: Npar17(:),matyp(:),ncrit(:)
        INTEGER, ALLOCATABLE:: N17(:),NG17(:)
        INTEGER, ALLOCATABLE:: IEN1(:),IEN2(:),IEN3(:),IEN4(:),IEN5(:)
        INTEGER, ALLOCATABLE:: IEN6(:),IEN7(:),IEN8(:),IEN9(:)
        INTEGER, ALLOCATABLE:: NEL1(:),NEL2(:),INCEL1(:),INCEL2(:)
        INTEGER, ALLOCATABLE:: INC117(:),INC217(:)
        REAL, ALLOCATABLE::  X1(:),X2(:),Temp1(:),Temp2(:)
        REAL, ALLOCATABLE::  D1(:),D2(:),D3(:),Temp19(:),Temp29(:)
        REAL, ALLOCATABLE::  F1(:),F2(:),F3(:),Temp110(:),Temp210(:)
        REAL, ALLOCATABLE:: Dt(:),Temp39(:),Temp310(:),g1(:),g2(:)
        REAL,ALLOCATABLE:: SPAR1(:),SPAR2(:),SPAR3(:),SPAR4(:),SPAR5(:)
        REAL,ALLOCATABLE:: SPAR6(:),SPAR7(:),SPAR8(:),SPAR9(:)
        REAL,ALLOCATABLE:: WT(:),GW(:),GRAV1(:),GRAV2(:)
     OPEN(10,FILE='c:\acad\data1',STATUS='UNKNOWN',ERR=334)
     OPEN(20,FILE='c:\acad\data2',STATUS='UNKNOWN')

        blank=' '
c    title card
c
        Read(10,FMT=100,ERR=333,END=333)title
        Write(20,100)title
c
c       card#2a
c
        Do i=1,14
          Read(10,*)I2a(i)
     enddo
        Write(20,fmt='(14i5)')(I2a(i),i=1,14)
c
c    card#2b
c
        Do i=1,5
          Read(10,*)I2b(i)
     enddo
        Write(20,fmt='(5i5)')(I2b(i),i=1,5)
c
c       card#3
c
        Read (10,*)irec
        Read (10,*)nslms
        Read (10,*)Beta
        Write(20,fmt='(2i5,f10.2)')irec,nslms,Beta
c
c    card#4
c
        Read (10,*)ia
        Allocate (n4(ia), nts(ia), dt(ia))
        Do i=1,I2a(6)
          Read (10,*)n4(i)
          Read (10,*)nts(i)
          Read (10,*)Dt(i)
        enddo
        Do i=1,I2a(6)
          Write(20,fmt='(2i5,f10.2)')n4(i),nts(i),dt(i)
        enddo
c
c    card#5
```

APPENDIX D
Sample AutoCAD Custom Files

ACAD.PGP

```
;ACAD.PGP FILE
;This file contains the external unix commands that can be accessed inside
;AutoCad. It can also contain aliases for AutoCad commands
;The format can be found in AutoCad Customization Guide p.24

deltri,del,0,What file is to be deleted?,4
mkdir,mkdir,0,What directory do you want to create?,4
edit,edit,0,Open what file?,4
dig,c:\\digdirt\\digdirt.exe,0,running,4
conv,c:\\acad\\rdconv.exe,0,writing,4
```

MENU.MNU

```
;Menu file to create a screen menu, taken from p.91
; of Customization guide

***SCREEN
[DIG-DIRT]
[*******]

[DATA...]$S=X $S=Data_Root
[WRITE]write conv ""
[EDIT] (startapp "notepad" "c:/acad/data2")
[RUN]dig "" getinfo
[TIMESTEP]timestep
[DRAWORIG]drawmesh
[DRAW_DEF]drawdef
[ELEMNUMB]elemnum
[DWSTRESS]drstress
[DWSTRAIN]
[EXCAVATE]TEXT 15.0,15.0 1.0 0 Not Yet Available ;
[LAYERS..]$S=X $S=Layers
[DRAW...]$S=X $S=Draw_Root
[EDIT...]$S=X $S=Edit_Root
[ERASE]erase all ;;;
[ZOOM]zoom
[HELP]help
[EXIT]menu c:/r13/win/support/acad.mnc
```

References

[1] Omura, G. (1995). *Mastering AutoCAD 13 for Windows 95, Windows 3.1, and Windows NT (second edition).* Sybex.

[2] Raker, D. & H. Rice. (1990). *Inside AutoCAD: The Complete AutoCAD Guide (sixth edition).* New Riders Publishing.

[3] Autodesk. (1995). *AutoCAD Release 13 Customization Guide.*

[4] Autodesk. (1995). *AutoCAD Release 13 Command Reference.*

[5] Autodesk. (1995). *AutoCAD Release 13 User's Guide.*

[6] Autodesk. (1995). *AutoCAD Release 13 ADS Developer's Guide.*

[7] Etter, D.M. (1993). *Structured Fortran 77 for Engineers and Scientists (fourth edition).* Benjamin Cummings Publishing Co.

[8] Mircrosoft. (1995). *Fortran Powerstation Reference (professional edition).*

[9] The Math Works Inc. (1992). *The Student Edition of Matlab.* Prentice Hall.

[10] Borja, R.I. (1990). "Analysis of incremental excavation based on critical state theory." J. Geotech. Engrg., ASCE, 116(6), 964-985.

[11] Borja, R.I. (1992). "Free boundary, fluid flow, and seepage forces in excavations." J. Geotech. Engrg., ASCE, 118(1), 125-146.