2 GETTING STARTED

This section provides the first-time user with an introduction to *UDEC*. *Getting Started* contains instructions for program installation and start-up on your computer. It also outlines the recommended procedure for applying *UDEC* to problems in geo-engineering, and includes simple examples that demonstrate each step of this procedure. When *UDEC* is executed for the first time, a dialog requesting permission to copy the data files and other user resources to your documents folder will appear. This is done to avoid operating system permission conflicts when attempting to open files in the "C:\Program Files" folder. It is recommended that you allow *UDEC* to copy example files to your "My Documents\itasca\udec700" folder. (This can also be done at any time by clicking on TOOLS/COPY APPLICATION DATA in the GUI or HELP/COPY APPLICATION DATA in the *GIIC* .)

If you are familiar with the program but only use it occasionally, you may find this section (in particular, Section 2.6) helpful in refreshing your memory about the mechanics of running *UDEC*. More complete information on problem solving is provided in Section 3. For Additional information on Running UDEC click on TOOLS in UDEC.

UDEC can be operated in *command-driven* using the GUI or graphical, *menu-driven* mode using the *GIIC* . For most of the examples in this manual, input is entered and results are viewed using the command-driven mode. We believe this is the clearest way for you to understand the operating procedures for *UDEC*. As explained previously in Section 1, the command-driven structure allows *UDEC* to be a very versatile tool for use in engineering analysis. However, this structure can present difficulties for new or occasional users. Command lines must be entered as input to *UDEC*, either interactively via the keyboard or from a remote data file, in order for the code to operate. There are over 65 main *commands* and more than 400 command modifiers (called *keywords*) recognized by *UDEC*.

The menu-driven mode is an easy-to-use alternative to the command-driven procedure. All of the commands in *UDEC* can be accessed by point-and-click operation from the graphical mode. We call this mode the "*GIIC*, " for *Graphical Interface for Itasca Codes*.

Getting Started contains the following information:

1. A step-by-step procedure to install and start up *UDEC* on your computer is given in Section 2.1. This includes the system requirements for operating *UDEC* (Section 2.1.1), the installation procedure (Section 2.1.2), a description of the components of the *UDEC* program and related files (Section 2.1.3), the memory allocation (Section 2.1.4), utility software and graphics devices (Section 2.1.5), start-up and operation procedures (Section 2.1.6), identification of version number (Section 2.1.7) and installation test (Section 2.1.8).

2. This is followed in Section 2.2 by instructions on running *UDEC*. Section 2.2.1 introduces the *GIIC*, and Section 2.2.2 provides a tutorial on running *UDEC* in menu-driven mode. Section 2.2.3 describes the procedure for running *UDEC* in the command-driven mode, and Section 2.2.4 includes a tutorial to help you become familiar with common input commands.

3. There are a few things that you will need to know before creating and running your own *UDEC* model (i.e., you need to know the *UDEC* terminology). The nomenclature used for this program is described in Section 2.3. The definition of a *UDEC* model is given in Section 2.4. You should also know the syntax for the *UDEC* input language when running in command-driven mode; an overview is provided in Section 2.5.

4. The mechanics of running a *UDEC* model are described in separate steps; in Section 2.6, each step is discussed separately and simple examples are provided.*

5. The sign conventions, systems of units and precision limits used in the program appear in Sections 2.7, 2.8 and 2.9, respectively.

6. The different types of files used and created by *UDEC* are described in Section 2.10.

---

* The data files in this section are all created in either the *FISH editor* pane of the *GIIC*, or in an external text editor. The files are stored in "ITASCA\UDEC700\Datafiles\Users_Guide\Getting_Started" with the extension ".DAT." A GUI (.udprj) project file and a *GIIC* project file (.prj) is also provided for each example. In order to run an example and compare the results to plots in this section, open a project file in the *GIIC* by clicking on the File / Open Project menu item and selecting the project file name (with extension ".PRJ"). Click on the *Project Options* icon at the top of the *Record* pane, select *Rebuild unsaved states*, and the example data file will be run and plots created. In the GUI click on the *Project* tab and select the "Master.dat" and run it. The example data file will be run, and plots created.

2.1 Installation and Start-up Procedures

2.1.1 System Requirements

To install and operate *UDEC*, your computer must meet the following minimum requirements (note that these are minimum requirements and may not represent what would be need to build large UDEC models).

Processor – A processor with a minimum clock speed of 1 GHz is recommended. The speed of calculation for a *UDEC* model is directly related to the processor speed. Therefore, the selection of a high-speed processor is a key factor for improving computational efficiency.

Hard Drive – At least 400 MB of hard disk space must be available to install *UDEC*. In addition, a minimum of 500 MB disk space should be available for model save files.

RAM – The minimum amount of RAM required to load *UDEC* with the *GIIC* is approximately 90 MB. By default, *UDEC* allocates 400 MB RAM for model save files. The *GIIC* also allocates 200 MB, by default, for plot files. The memory allocated for both model files and *GIIC* plots for a *UDEC* model can be adjusted by the user (see Section 2.1.4).

Operating System – *UDEC* is a 64-bit native Windows application. *UDEC* running on Windows 7, Windows 8, or Windows 10 is currently supported by Itasca. *UDEC* may run on earlier versions of Windows, but this is not supported by Itasca.

Operation on PC Networks – A network-license version of *UDEC* is available. The network key allows a single hardware dongle to be placed at a central location. Individual users may then run *UDEC* from any computer on the network. Network keys require a special licensing arrangement and installation. Contact Itasca for details.

2.1.2 Installation Procedure

UDEC installation operates under Windows 7, Windows 8, and Windows 10. Earlier versions of Windows or other operating systems will may run the installation, but this is not supported by Itasca.

A default installation of *UDEC* will install the program, its example files and the complete *UDEC* manual. The Adobe Acrobat Reader is necessary for viewing the manual; the Acrobat Reader is available for free from Adobe Corporation (http://www.adobe.com).

To begin installation, insert the DVD into the appropriate drive. If the autorun feature for the USB (or DVD) drive is enabled, a menu providing options will appear automatically. If this menu does not appear, at the command line (START –> RUN in Windows), type "`[drive]:\start.exe`" to access the menu. The option to install *UDEC* may be selected from this menu.

After installing the software, connect the *UDEC* hardware key to the USB port on the computer before using the code. The *UDEC* hardware key must be connected to the computer (either directly if a single-user key, or via a network if a network key) for full operation of *UDEC*.

2.1.3 Components of UDEC

UDEC Version 7.0 is provided as a 64 bit *double-precision* executable file. The start menu link "UDEC 7.0" will execute the GUI version of *UDEC* . The startup link "UDEC 7.0 GIIC" will execut the *GIIC* version of *UDEC*.

All files related to the *GIIC* for *UDEC* are stored in the "\ITASCA\UDEC700\GUI" folder *UDEC* communicates with the *GIIC* via the JAVA Runtime Environment. The user can switch from the graphics mode to the text (command-driven) mode by pressing the FILE/EXIT GIIC button in the graphics mode, and return to the graphics mode by typing

```
giic
```

from the command line in text mode.

UDEC is compiled usin an Intel compiler executin in Winsdows Visual Studio 2017. The *GIIC* is a JAVA application run using JAVA Runtime Environment, standard edition, version 1.6.0.

2.1.4 Memory Allocation

Automatic memory-allocation logic has been implemented in *UDEC* for Intel-based computers. When loaded, *UDEC* will, by default, assigne 400MB to the main array.

You can change the amount of memory used by *UDEC* by using the **model array *n*** command, where ***n*** is the memory to be made available (in MB). This command is accessed in the *GIIC* from the FILE/MEMORY SETTINGS button.

The amount of memory allocated for *GIIC* plots is set by default to 200 MB. This can be changed by using the **block giic memory *n*** command, where ***n*** is the memory to be made available for *GIIC* plots (in MB). This command can also be accessed in the *GIIC* from the FILE/MEMORY SETTINGS button.

As a guide, Table 2.1 summarizes the approximate maximum numbers of rigid or deformable blocks that can be created for different sizes of available RAM in *UDEC* 7.0.

Table 2.1 Maximum number of UDEC blocks in available RAM

Available RAM (MB)	Maximum number of rigid blocks	Maximum number of deformable blocks*
100	39,380	26,875
200	78,750	53,750
400	157,500	107,500
800	315,000	215,000

* assumes 8 translational-degrees-of freedom per block.
Maximum number of blocks will be reduced for more degrees of freedom.

2.1.5 Utility Software and Graphics Devices

Several types of utility software and graphics devices that can be of great help while operating *UDEC* are available.

Editors – When running *UDEC* from the *GIIC*, an input data file is created automatically as the model is generated in the graphical mode. This is the recommended approach to create *UDEC* models for beginners. It is possible to edit or modify this data file directly in the *GIIC*. The data file can also be saved and edited outside of the *GIIC* by an external text editor, or in the GUI editor, in order to reproduce or modify the model in later analyses. Any text editor that produces standard ASCII text files may be used. Be careful if using more "advanced" word-processing software (e.g., Word): this software typically encodes format descriptions into the standard output format; these descriptions are not recognized by *UDEC*, and will cause an error. *UDEC* input files must be in standard ASCII format.

Graphics Output – Plots are created in the *GIIC* via the FILE/PRINT PLOT menu item. The *Print plot* dialog will appear to generate a plot, as shown in Figure 2.1. A plot title, and a two-line customer title that will appear at the bottom of the plot legend, can be added to plots from this menu.

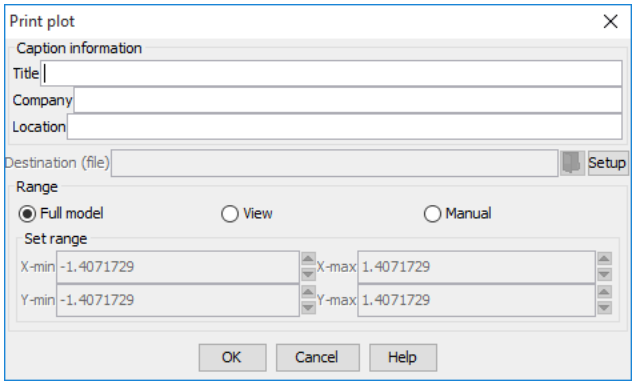

Figure 2.1 Print plot dialog

Third party graphics software can assist in the production/presentation of *UDEC* results.

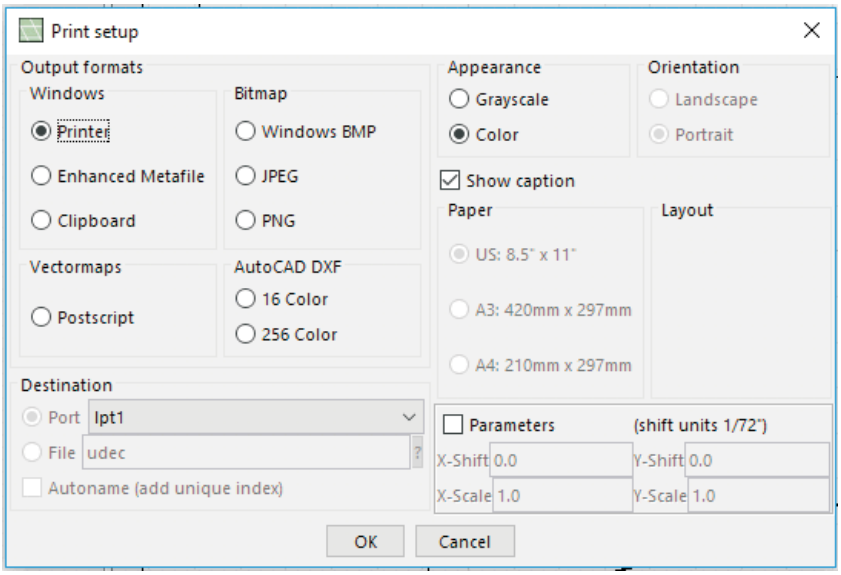

Figure 2.2 Print setup dialog

2.1.6 Start-up

The default installation procedure creates an ITASCA UDEC group under PROGRAMS on the user's START menu in Windows. The ITASCA UDEC group contains the UDEC 7.00 GIIC shortcut, which can be used to start the code.*

To load *UDEC*, simply click the UDEC 7.00 GIIC button. *UDEC* will start up in graphics mode *GIIC*. The graphics mode may take a few seconds to initialize while the JRE is being loaded to run the *GIIC*. The initialization time can be affected by other programs running in the background. If you notice a significant delay in the initialization of the graphics mode, it may be necessary to close other Windows applications. When loaded, the *UDEC* window appears, as shown in Figure 2.3.

When *UDEC* is executed for the first time, a dialog requesting permission to copy the data files and other user resources to your documents folder will appear. This is done to avoid operating system permission conflicts when attempting to open files in the "C:\Program Files" folder. It is recommended that you allow *UDEC* to copy example files to your "My Documents\itasca\udec700" folder. (This can also be done at any time by clicking on HELP / COPY APPLICATION DATA.)

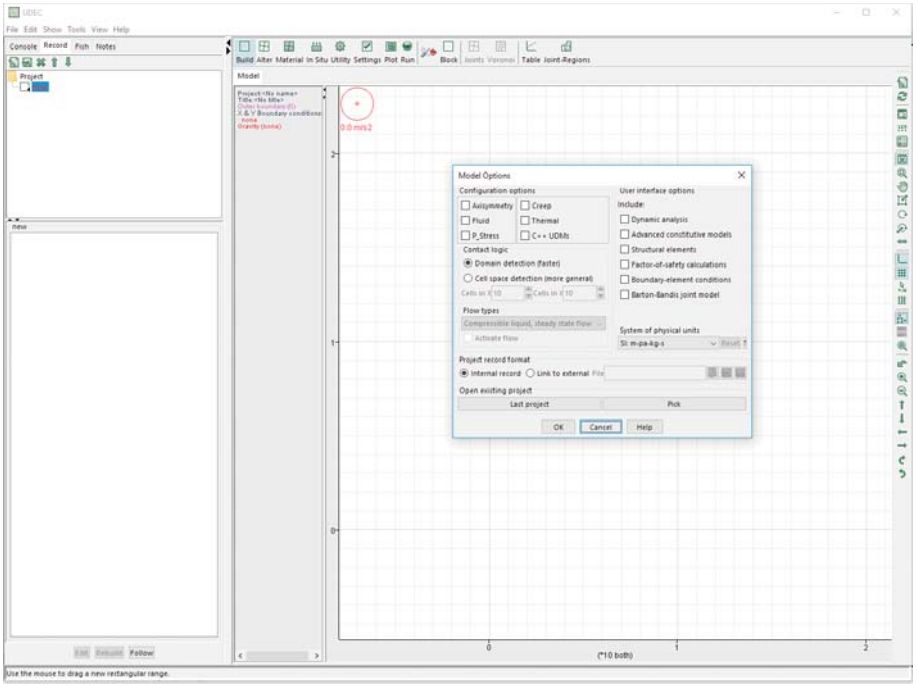

Figure 2.3 UDEC start-up window

* Be sure that the *UDEC* hardware key is attached to a USB port on your computer.

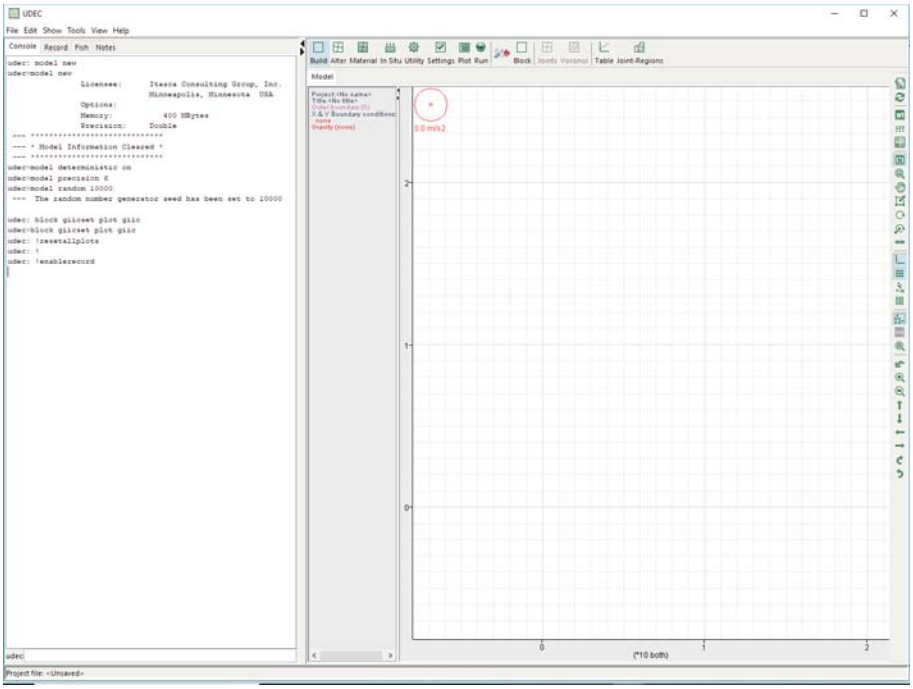

Figure 2.4 UDEC Console pane

2.1.7 Version Identification

The version number of *UDEC* follows a simple numbering system that identifies the level of updates in the program. There are three numerical identifiers in the version number – i.e.,

```
Version I.JK
```

where:

- `I` is an integer starting with 1 that identifies a major release of the code;
- `J` is an integer that is incremented whenever a modification is made that requires a major change to the code structure for a supplemental upgrade release of *UDEC*; and
- `K` is an integer that is incremented when minor modifications are officially released as an update to the current version.

In addition to the version number, *sub-version* numbers are also used to identify minor changes to *UDEC* that have been made since the official version was released. Users may access the latest sub-version of the current version of *UDEC* via the Internet. (Contact Itasca for further information.)

The version number is given in the title bar at the top of the *UDEC* window (see Figure 2.3). The *UDEC* version number (and the version numbers for the *GIIC* and JAVA Runtime Environment

associated with this version of *UDEC*) are provided in the *About UDEC* dialog, accessed from the HELP / ABOUT UDEC menu item.

The *UDEC* version number can also be obtained by typing the command

```
list version
```

at the `udec:` command-line prompt.

2.1.8 Installation Test

A simple *UDEC* project file is included so that you can verify that *UDEC* is properly installed on your computer.

To run this test, first start up *UDEC* following the procedure in Section 2.1.6. The *GIIC* window shown in Figure 2.3 should appear. Then, perform the following steps.

1. Check the OPEN OLD PROJECT button in the *Model Options* dialog. This will open an *Open Project* dialog.
2. Select the file named "TEST.PRJ" from the "My Documents\ITASCA\UDEC700 \Datafiles\Users_Guide\Getting_Started" folder.
3. Press <OPEN> in the *Open Project* dialog.
4. Double-click on the file name "test.sav," which appears in the *Record Resource* pane. See Figure 2.5.
5. Click on YES in the *Warning* dialog that appears (as shown in Figure 2.5). The test example will be run and the model will be executed for 100 calculation steps.
6. When the run is finished, click on the Y-DISPLACEMENT CONTOUR tab in the *model-view* pane, and a y-displacement contour plot will appear (as shown in Figure 2.6).
7. To exit *UDEC*, click on the FILE / QUIT menu item.

If you are not able to reproduce the results of this test, you should review the system requirements and installation steps in Sections 2.1.1 and 2.1.2. If you still have difficulty, we recommend that you contact Itasca and describe the problem you have encountered and the type of computer you are using (see Section 5.2 for error-reporting procedures).

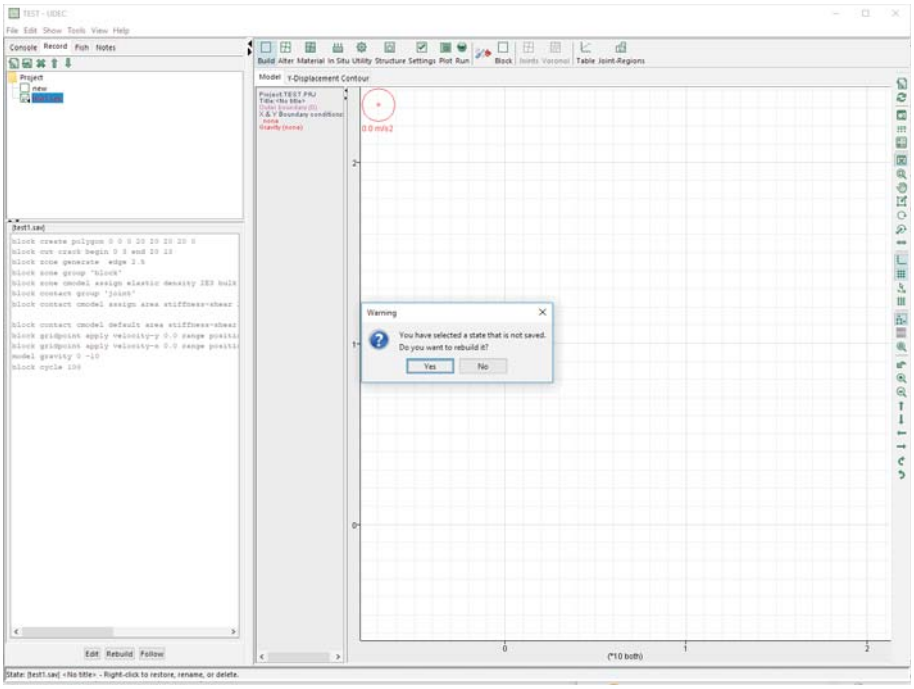

Figure 2.5 Double-click on the file named "test.sav" and click on YES *in the Warning dialog to execute the test project named "TEST.PRJ"*

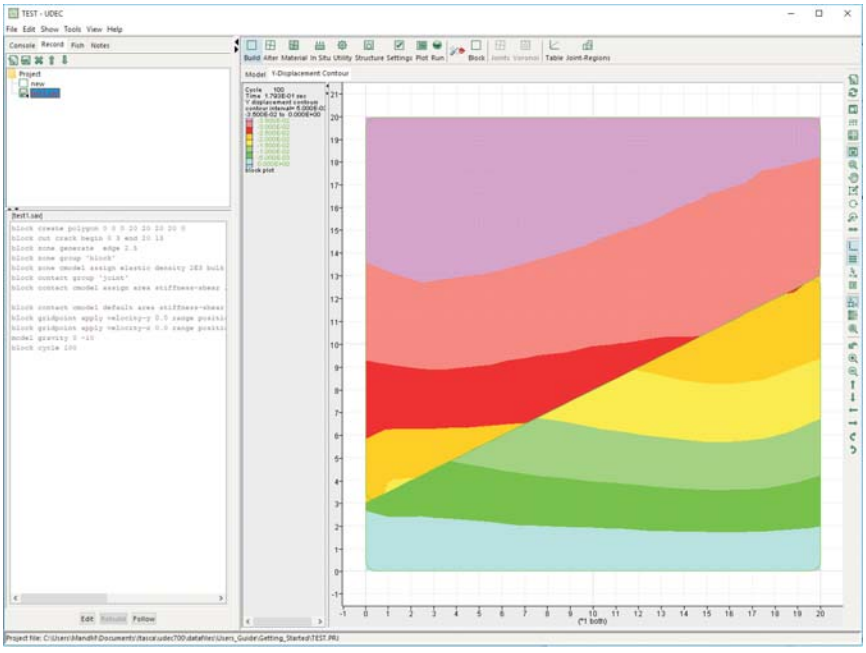

Figure 2.6 Graphics plot from "TEST.PRJ"

2.2 Running *UDEC*

UDEC can be run in menu-driven mode (*GIIC*) or command-driven mode (GUI). We recommend that you use the menu-driven mode to become familiar with the procedure for creating and solving a *UDEC* model. The installation test in Section 2.1.8 is performed in the menu-driven mode. The procedure to operate *UDEC* in the menu-driven mode is described in Section 2.2.1. A simple tutorial is provided in Section 2.2.2.

The procedure to operate *UDEC* from the command-driven mode is described in Section 2.2.3. This procedure requires direct input of *UDEC* commands; all commands are defined in the Help in *UDEC*. A simple tutorial in command-driven mode is given in Section 2.2.4.

2.2.1 Running UDEC in Menu-Driven Mode

The *Graphical Interface for Itasca Codes* (*GIIC*) is a menu-driven graphical interface developed to assist users in operating Itasca codes. The *UDEC-GIIC* is easy to use, with a point-and-click operation that accesses all commands and facilities in *UDEC*. The structure of the *GIIC* is specifically designed to emulate expected Windows features, and allows general mouse manipulation of displayed items that correspond to *UDEC* operations. You should be able to begin solving problems with *UDEC* immediately, without the need to wade through commands to select those necessary for your desired analysis.* This section provides an introduction to the *GIIC*, and includes a simple tutorial to help you get started. You will notice that a HELP menu is provided in the main menu bar for the *GIIC*. [HELP] buttons are also included with each tool in the *GIIC*, and [HELP] panes can be opened by right-clicking on model tool tabs. Consult these *Help* views for detailed information on specific *GIIC* features.

2.2.1.1 Entering the GIIC and Selecting Analysis Options

The *GIIC* starts automatically when *UDEC* is loaded by selecting the [UDEC 7.00 GIIC] button from the [ITASCA UDEC 7.0] under [PROGRAMS] on the user's [START] menu in Windows.

---

* Before conducting full engineering analyses with *UDEC*, we strongly recommend that you read the **User's Guide** and **Example Applications** volumes to gain a general understanding of the construction of a *UDEC* model, the process of assigning material properties and initial conditions to a model, and the calculation procedure.

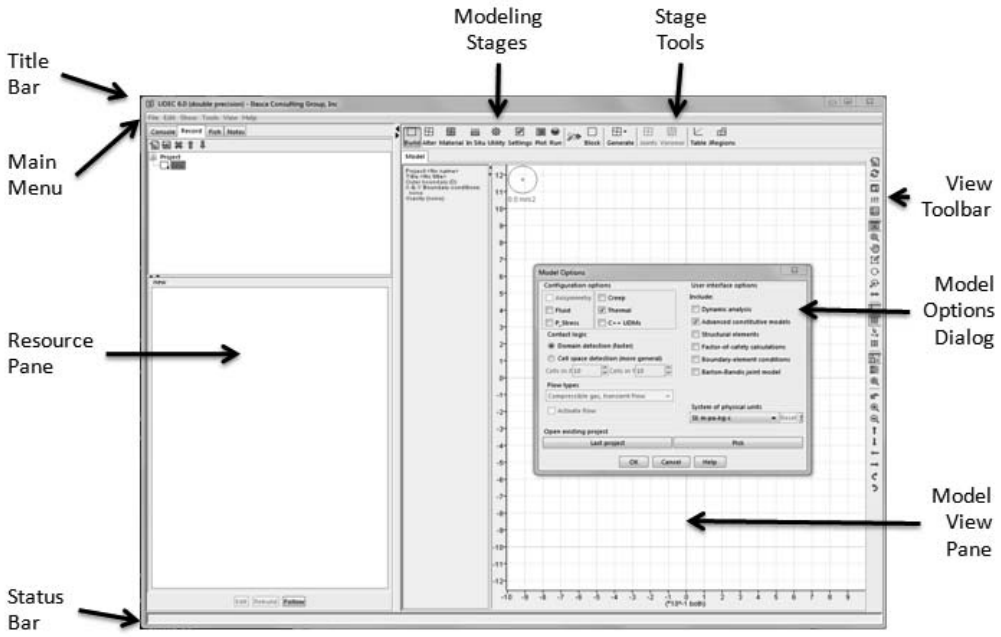

Figure 2.7 The UDEC-GIIC main window

The code name and current version number are printed in the title bar at the top of the window, and a main menu bar is positioned just below the title bar. Beneath the main menu bar are two windows: a *resources* pane and a *model-view* pane. The *resources* pane contains four tabbed panes with text-based information. A *Console* pane shows text output and allows command-line input (at the bottom of the pane). A *Record* pane shows a record of commands needed to generate the current model project state. The record is in a "project tree" format that shows changes between save files. Save states are displayed in a tree structure. The record can be exported to a data file as a set of *UDEC* commands that represent the problem being analyzed. A *Fish* pane opens the *FISH* editor and facilitates execution of *FISH* functions. Project notes are shown in the *Notes* pane. Each of these tabbed panes can be turned off individually, or all of the resource panes can be turned off, from the SHOW/RESOURCES menu item in the main menu.

The *model-view* pane shows a graphical view of the model. Additional tabbed views that display user-defined plots can be added to this window. At the top of the *model-view* pane is a tab bar containing modeling-stage buttons. When you click on a modeling-stage button, a set of tools that are associated with that stage will open. The BUILD stage tools are shown in Figure 2.7. When you click on a tool, this opens a separate window that allows access to specific tools to create and run your model.

You can use the VIEW menu item in the main menu to manipulate any *view* pane (e.g., translate or rotate the view, increase or decrease the size of the view, turn on and off the model axes). The VIEW menu is also available as a VIEW toolbar that can be turned on from the SHOW menu. The VIEW toolbar is shown to the right of the *model-view* pane in Figure 2.7.

An overview of the *GIIC* operation is provided in the HELP menu. The menu also contains a list of Frequently Asked Questions about the *GIIC*, and an index to all *GIIC Help* files. The HELP menu item in the HELP menu provides direct access to the *UDEC* command references. The PDF FILES menu item in the HELP menu provides direct access to other UDEC documentation manuals.

The text field with the `udec:` prompt located at the bottom of the *Console* pane allows you to enter *UDEC* commands directly from the *GIIC*. The *Console* pane will echo the commands you enter. You should not need to use the command line at all; it is provided as a shortcut, should you prefer to type a command rather than use the graphical interface. A status bar that displays information related to the currently active view or tool is located at the bottom of the main window.

A *Model Options* dialog box will appear every time you start the *GIIC* or begin a new model project. The dialog is shown with the main *GIIC* window in Figure 2.7, and separately in Figure 2.8. This dialog identifies which modes of analysis are available to you in your version of *UDEC*.* The *UDEC Configuration options* (axisymmetry, fluid flow, plane stress, creep, thermal and C++ UDMs) and the *Contact logic* (domain detection or cell space detection) must be selected at the beginning of a new analysis, while the *User interface options* (dynamic analysis, advanced constitutive models, structural elements, factor-of-safety calculation, boundary-element conditions and Barton-Bandis joint model) can be included at any time in the model run. This menu also has an option to link the *GIIC* project file to an external .dat file that can be shared between the *GIIC* and the GUI versions of *UDEC* .

You can select a system of units for your analysis in the *Model Options* dialog. Many parameters will then be labeled with the corresponding units, and predefined values, such as gravitational magnitude and properties within the material database, will be converted to the selected system. The selection of system of units should be done at the beginning of the analysis.

If you are a new user, or only intend to perform a simple static analysis, we recommend that you click the OK button in the *Model Options* dialog to access the basic *UDEC* features. In this case, the Coulomb slip joint model, and either rigid blocks or deformable blocks with the isotropic-elastic or Mohr-Coulomb models are active, and a static, plane-strain analysis is performed in the *GIIC*. If you wish to come back later in the analysis and, for example, add structural elements, click FILE/MODEL OPTIONS in the main menu. This will reopen the *Model Options* dialog, and you can click on the INCLUDE STRUCTURAL ELEMENTS check box to add the *Structures* tool to the *Modeling Stages* tab bar.

---

* Note that C++ user-defined models, creep material models and the Barton-Bandis joint model are separate modules that can be activated at an additional cost.

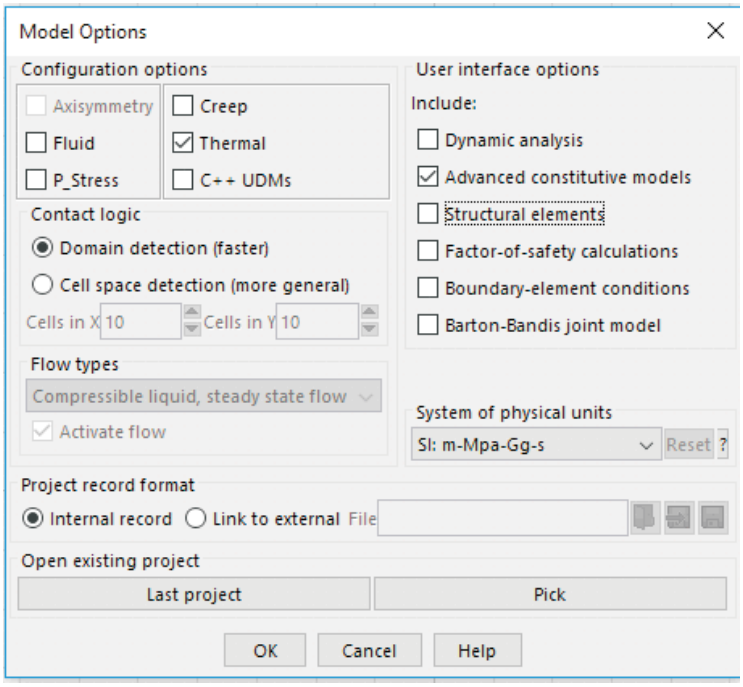

Figure 2.8 The Model Options dialog box

2.2.1.2 Changing GIIC Preferences

After you have selected which model options you wish to have operating during your analysis, you can save these preferences so that these selections are active each time you enter the *GIIC*. You can also save your preferences for the look-and-feel of the *GIIC* on start-up. You can select which *resource* pane you wish to have open, as well as the size of this pane and the *model-view* pane. Preferences for the *GIIC* appearance can be changed. Open the SHOW menu in the main menu to change the look-and-feel of the *GIIC* panes and toolbar. Once you are satisfied, click FILE / SAVE PREFERENCES in the main menu.

2.2.1.3 Modeling-Stage Tools

The modeling stages are accessed above the *model-view* pane, as shown in Figure 2.7. The stages are arranged in a logical progression for building and solving your model. The order follows the recommended procedure for problem solving. The modeling stages are described:

- The model geometry is first created via the [Build] stage. A single block is defined via the [Block] tool, and then split into separate blocks using either the [Joints] tool or the [Voronoi] tool to create the joint system in the model.
- Blocks are made deformable by installing finite-difference zones within each block using the [Zone] tool in the [Alter] stage. Excavations can be made (with either the [Delete & Add] tool or the [Cut & Fill] tool) and blocks can be hidden to facilitate model creation (with the [Hide & Seek] tool) in the [Alter] stage.
- Material models and properties are assigned to the blocks and joint structure in the model, using the tools accessed from the [Material] stage. Deformable block models and properties are assigned using the [ZoneMat] tool, and joint models and properties are assigned via the [JointMat] tool.
- Boundary and initial conditions are applied via the tools in the [In Situ] stage.
- The [Utility] stage provides tools to monitor model variables and access model results.
- The [Settings] stage allows model global conditions to be set or changed during the analysis.
- All plotting facilities in *UDEC* are accessible via the tools in the [Plot] stage.
- Calculations are performed using tools from the [Run] stage.

Note that model conditions can be changed at any point in the solution process by re-entering a modeling stage tool. For example, joint properties can be changed at any time via the [JointMat] tool in the [Material] stage, and pressure or stress boundary conditions can be changed via the [Apply] tool in the [In Situ] stage. Also, if you select structural elements in the *Model Options* dialog, a [Structure] stage will be included to access structural support for the model. If you include the factor-of-safety calculation in the *Model Options* dialog, a [SolveFos] tool will be added to the [Run] stage. If you include the optional Barton-Bandis joint model in the *Model Options* dialog, properties and settings related to this option will be made available in the modeling tools.

When you click on each of the modeling-stage tools, a separate window or dialog that provides access to operations related to that tool will appear. The [Build] stage tools are shown in Figure 2.7.

Next, a simple tutorial is given to provide an introduction to the model tools, and to help you become acquainted with the *GIIC* operation.

2.2.2 A Simple Tutorial – Use of the GIIC

In this section a simple tutorial is provided to help you get started using the *GIIC*. The tutorial demonstrates the use of several modeling tools to create and solve a simple geotechnical problem.

The example is a circular tunnel excavated at a shallow depth in jointed rock. The rock contains a continuous joint set dipping at 320° with a joint spacing of 1.0 m. A single fault dipping at 230° intersects the tunnel. The joint structure results in a triangular-shaped rock block at the crown of the tunnel. Two jointed rock types are evaluated: a rock with strong joints, and a rock with weak joints. The tunnel is excavated instantaneously, and the movement of the jointed rock around the tunnel is monitored for both rock types.

To begin, start up the *GIIC* by following the procedure given in Section 2.2.1.1. (If you have loaded *UDEC* by double-clicking on the UDEC 7.00 GIIC icon in the Itasca UDEC 7.0 group, the *GIIC* will start up automatically.)

A simple, static, plane-strain analysis is performed in the SI system of units. Block constants are set automatically for this example. Normally, you would use the default domain-logic contact detection mode. In this case, if you wish to monitor the movement of any rock blocks that may detach and fall from the roof of the tunnel, you should use the cell-space detection mode to track the movement and potential contacts of falling blocks. You can click on the CELL SPACE DETECTION button in the *Model Options* dialog to switch to this detection mode. (See Figure 2.9.) You can now press OK to save these settings.

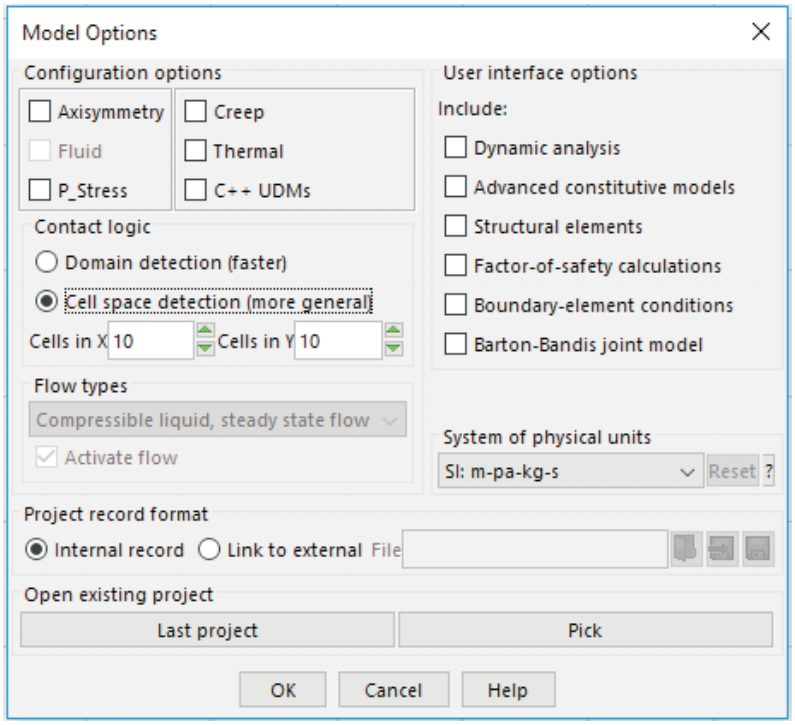

Figure 2.9 The Model Options dialog with cell space detection logic selected

When beginning a modeling project, the *Project File* dialog will appear so that you may set up a project file for this exercise. The dialog is shown in Figure 2.10. You are asked to assign a project title and file name for this project. You should click on the disk icon to the right of *Filename:* in this dialog to select a folder in which to save the project file. You can save the project as "TUNNEL.PRJ." (Note that the ".PRJ" extension is appended automatically.) The location of the project file and the project file name appear in the *Project File* dialog, as shown in Figure 2.10. The project file contains the project record and allows access to all of the model save (".SAV") files that you will create for the different stages of this analysis. You can stop working on the project at any stage, save it and reopen it at a later time simply by opening the project file (from the FILE/OPEN PROJECT menu item); the entire project and associated model save files will be accessible in the *GIIC*.

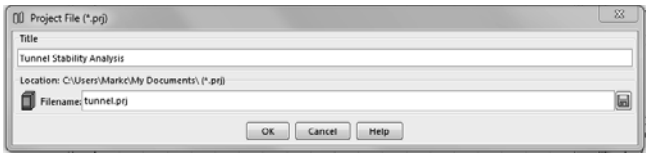

Figure 2.10 Project File dialog

You can now begin the model creation. To set up the initial block, you should click on the BLOCK tool from the BUILD modeling stage. This tool creates the initial model geometry via the **block create** command. A *New block* dialog, as shown in Figure 2.11, opens first so you can enter the outer dimensions of the starting block for this model. The block constants, corner rounding length and minimum edge length are also specified in this dialog. You can press OK in the *New block* dialog after setting the block dimensions to 10 by 10 and selecting the automatic settings for rounding length and edge length, as shown in Figure 2.11. A plot of the block will now be shown in the BLOCK tool. This example is using SI units, and therefore the model domain is 10 m by 10 m. See Figure 2.12. Note that the boundary can be changed, and additional corners and edges can be added by using this tool. You can accept the present boundary geometry by pressing EXECUTE. A **block create** command is created and executed in *UDEC*. The *GIIC* returns to the *model-view* pane.

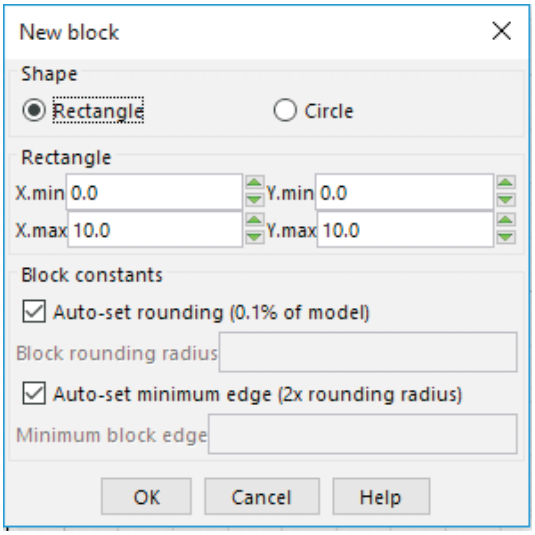

Figure 2.11 New block dialog

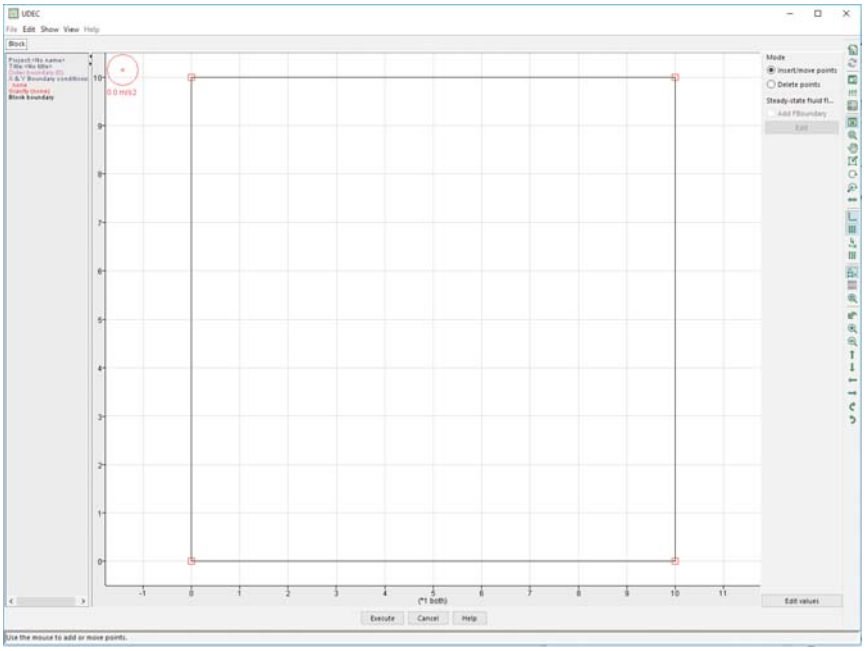

Figure 2.12 Virtual block in BLOCK *tool*

At this point, joints can be added to the model by clicking on the JOINTS tool in the BUILD stage. (If you want to limit the jointing to specific areas of the model, you should first click on the JREGIONS tool to define separate joint regions.) In the JOINTS tool, you click on the JOINT-CONTINUOUS radio button. You can click on the INPUT SHAPE button to input the geometry via the *JSet (continuous)* dialog. Figure 2.13 shows the dialog with the values specified for the continuous joint set. Click OK and then click GENERATE SHAPE to accept the configuration. A virtual joint set will be drawn in the JOINTS tool, as shown in Figure 2.14.

A single fault can be created by either using the *JSet (continuous)* dialog with the spacing set to a large value, or by selecting the LINE radio button and entering the endpoints of the fault line ($x1$ = 0.0, $y1$ = 1.5, $x2$ = 8.39, $y2$ = 11.5).

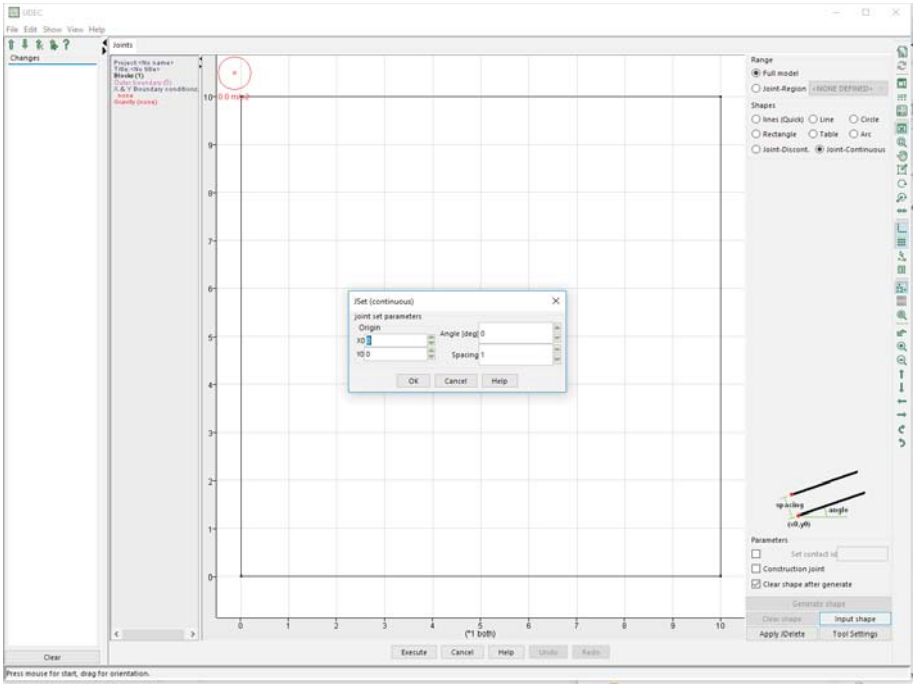

Figure 2.13 JSet (continuous) dialog used to enter the joint set geometry in the JOINTS tool

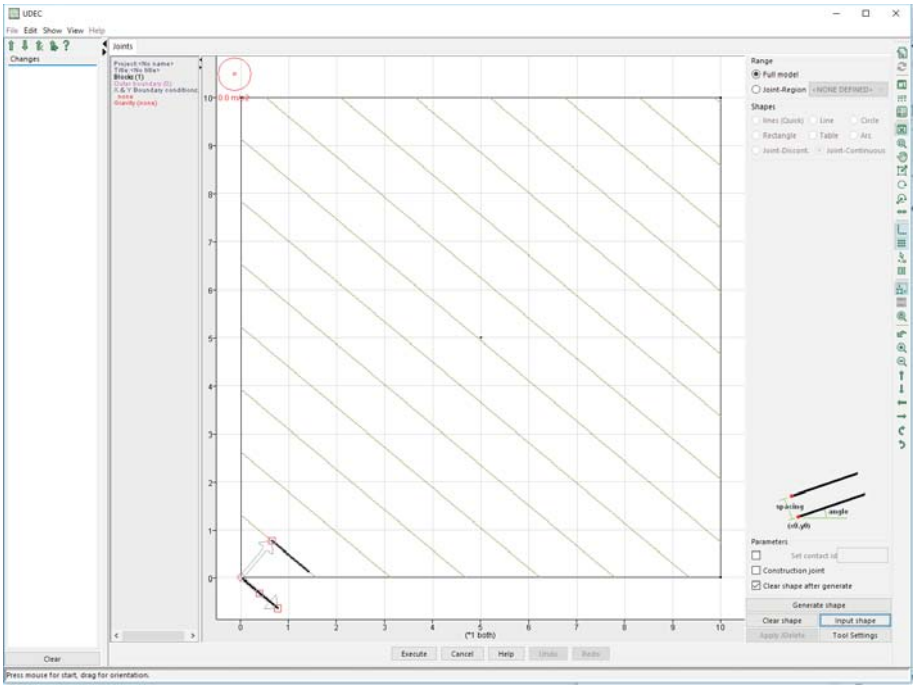

Figure 2.14 Continuous joint set created from the JSet (continuous) dialog

You can create the circular tunnel by selecting the CIRCLE radio button. You should move the mouse to a position on the block corresponding to the tunnel center, and press and hold the left mouse button while dragging the mouse. A circle tool with two red boxes will appear: one at the centroid, and one along the circle periphery (see Figure 2.15). You can move the circle and adjust its radius by pressing and holding the left mouse button while the mouse is positioned within each box. Alternatively, you can select values for the centroid coordinates and the circle radius with dialogs that open when you right-click the mouse while it is positioned within each box. The circle in Figure 2.15 is centered at $x = 5.0$, $y = 5.0$, and has a radius of 2.0 m.

When you press GENERATE SHAPE, the circle is added to the block. Note that this is a "virtual" block: any alterations you make within this block can be undone or changed. You simply press one of the arrow keys above the *Changes* pane to remove (or add) a command corresponding to the shape created in the virtual block.

Once you are satisfied with the alteration, you should press EXECUTE. This sends the command(s) to *UDEC*. The *UDEC* commands are processed, and the model with multiple blocks is displayed, as shown in Figure 2.16. The *UDEC* commands created thus far are shown in the *Record* pane in this figure.

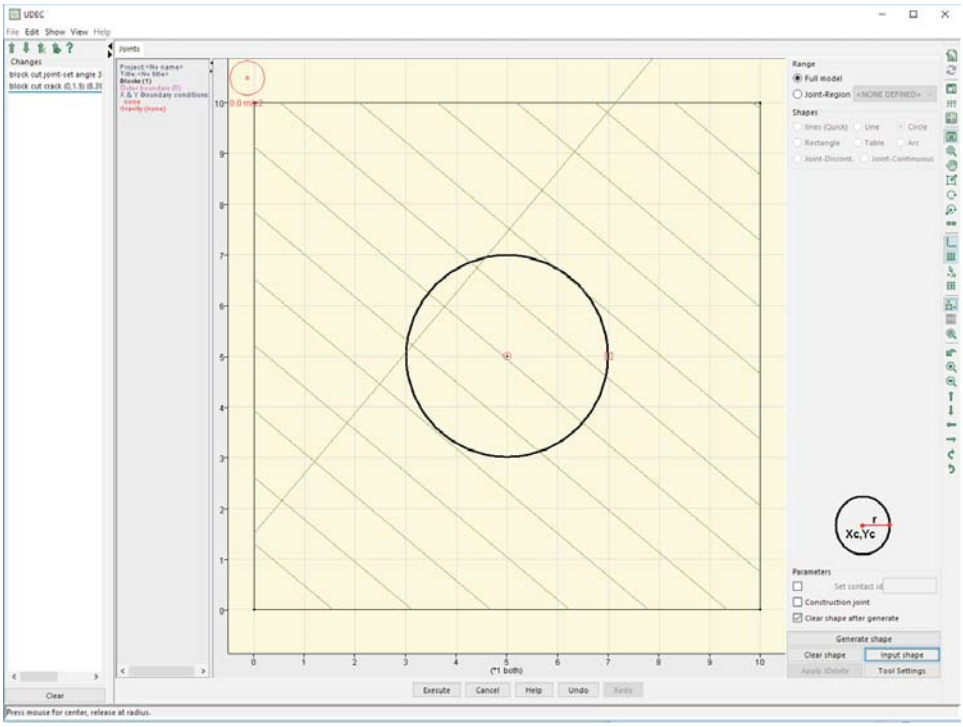

Figure 2.15 Circular tunnel created in the JOINTS tool

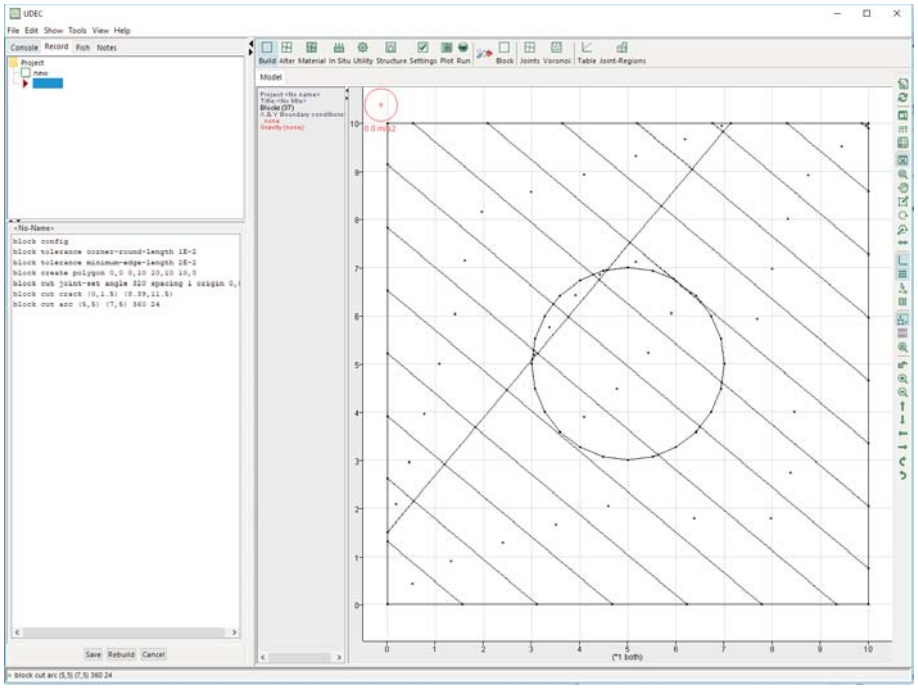

Figure 2.16 UDEC jointed rock model with circular tunnel in model-view pane

Next you should press the ZONE tool in the ALTER stage in order to make the blocks deformable. In this tool, you can select the zoning that will be applied to the blocks. You can select a maximum zone edge length of 0.5, and press SET ALL. Lines representing the approximate size of the zones that will be generated will be drawn on the model. Then press the EXECUTE button. This will cause the generation of the commands to zone the model. Now you should see a graphic of the zoned model in the *model-view* pane:

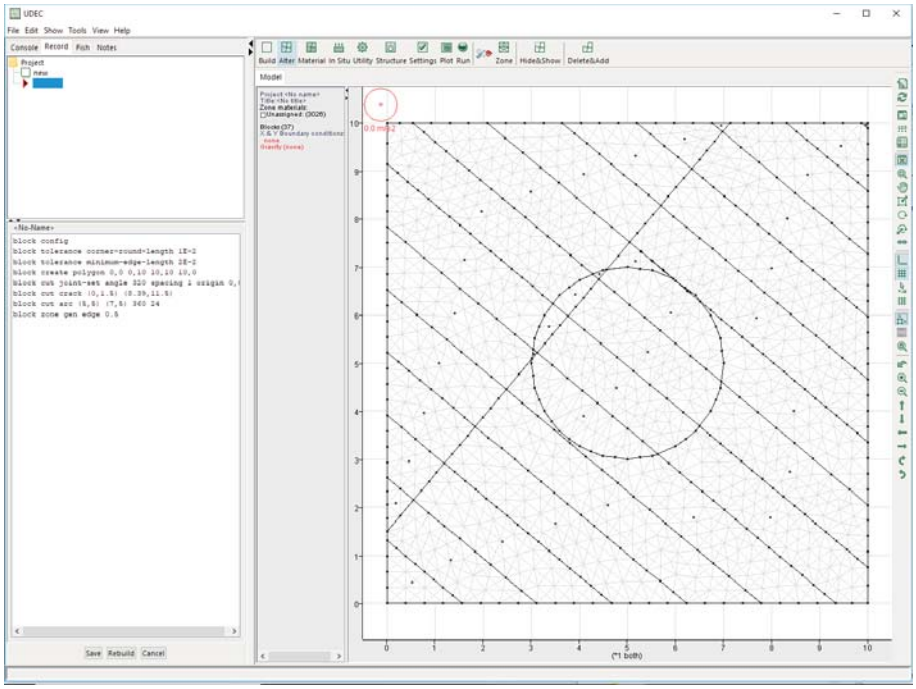

Figure 2.17 Zoning of blocks in UDEC model

You can use the MATERIAL stage to create and assign materials and their properties to the deformable blocks and joints within the model. Deformable block materials are created from the ZONEMAT tool. You should press the CREATE button in this tool to create a material. A *Property editor* dialog opens, as shown in Figure 2.18. You select a constitutive model (in this case the isotropic elastic model), and then assign a classification and material name. Soils and rocks can be divided into different classifications (such as "Tunnel" rock), with separate material names within a classification (such as "strong rock" and "weak rock"). The classification and material name are used to associate a **group** range name with each material.

You can create one material for this analysis: "rock" within the classification "Tunnel." The *Property editor* dialog with the selected properties for the rock is shown in Figure 2.18:*

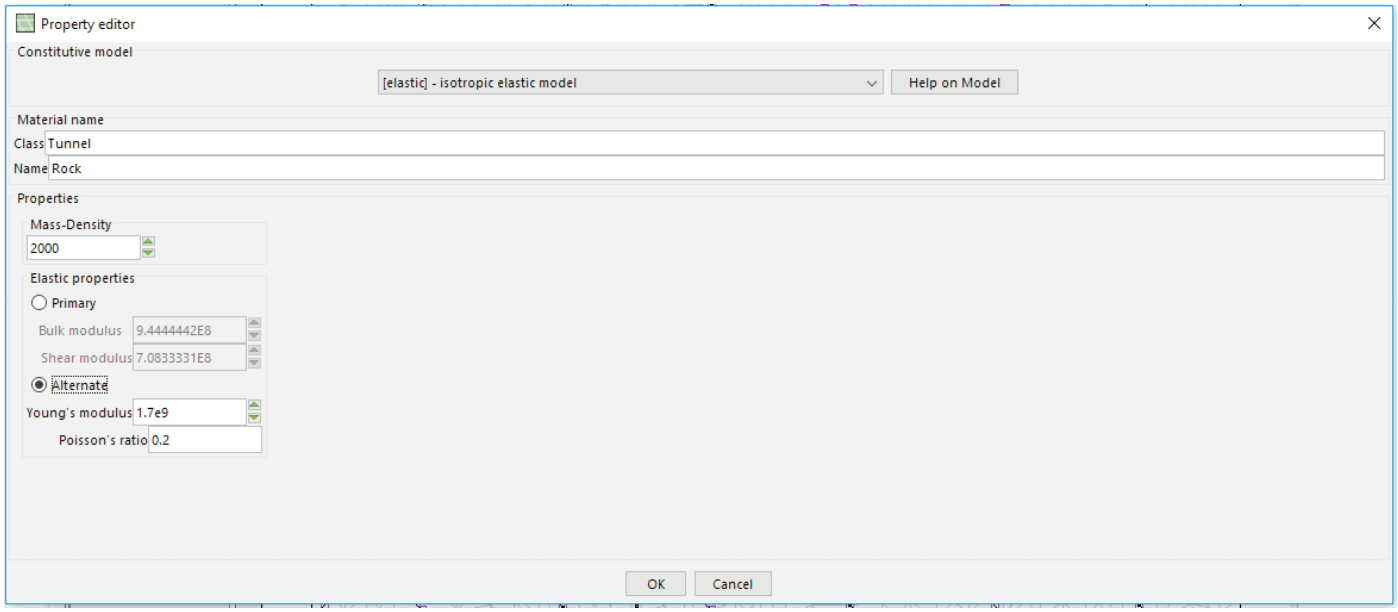

Figure 2.18 Property editor dialog to create deformable block materials

You should press OK in the *Property editor* dialog to create the material. The material is added to a material *List* shown on the right side of the ZONEMAT tool. Once all the materials required for an analysis have been created and added to the list, they can be assigned to the zones. It is possible to assign different materials to different zones in the blocks, or to different marked regions of blocks, using the *Range* tools provided in the ZONEMAT tool. In this example, you should assign the rock material to all zones. You should highlight the *Tunnel:rock* item and press the SET ALL button to assign this material to all zones in the block. Figure 2.19 shows the ZONEMAT tool with the rock material assigned. The **group** command is listed in the *Changes* pane when the material is assigned. You can now press EXECUTE to send the commands to *UDEC*.

---

* A database of common soil and rock materials and properties is also available by pressing the DATABASE button in the lower-right corner of the ZONEMAT tool. The database is divided into classification groups and material names. You can also create your own database of common materials within this database tool, which can be saved and loaded for other projects. Database materials are stored in a file with extension ".GPF."

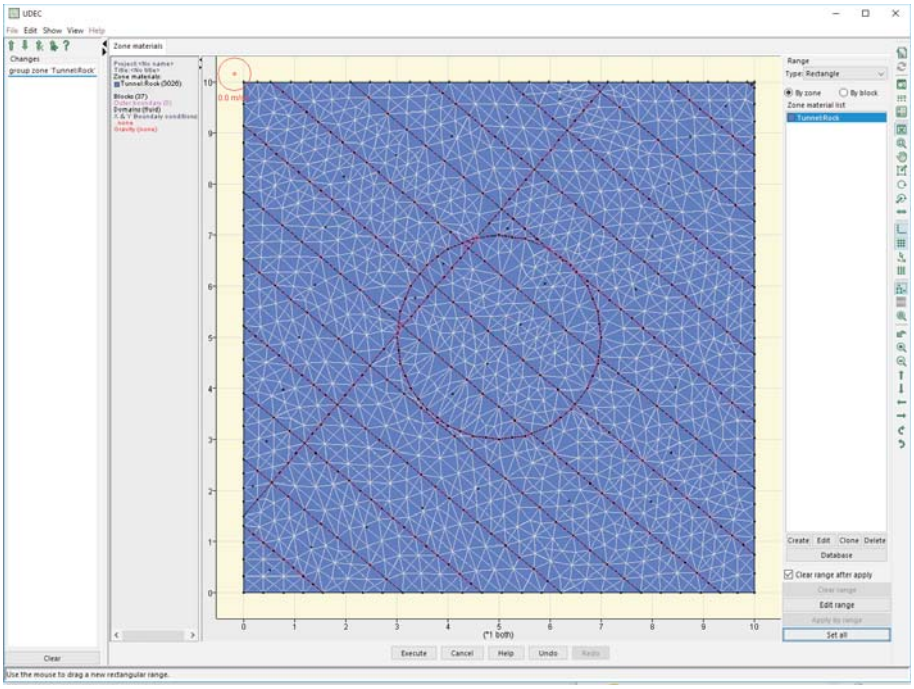

Figure 2.19 Zone material assigned in the ZONEMAT *tool*

You can select the JOINTMAT tool in the MATERIALS stage to create and assign materials and their properties to the joints within the model. Joint materials are created from a *Property editor* dialog that is opened by pressing the CREATE button in the JOINTMAT tool. Within this dialog, you can assign a classification and material name, prescribe a joint constitutive model type, and assign the material properties.*

You can press OK in the *Property editor* dialog to create the joint material. The material is added to a material *List* shown on the right side of the JOINTMAT tool. Once all the materials required for an analysis have been created and added to the list, they can be assigned to the joints. It is possible to assign different materials to different joints in the block, or to different marked regions of the block, using the *Range* tools provided in the JOINTMAT tool. In this example, you should create two different joint types: "strong joints" and "weak joints." The weak-joints material and properties are shown in Figure 2.20. The strong joints have the same shear and normal stiffnesses as the weak joints; the joint friction is 40°, and the joint cohesion and tensile strength are 100,000 Pa for the strong joints.

---

* A database of rock joint materials and properties is also available by pressing the DATABASE button in the lower-right corner of the JOINTMAT tool. The database is divided into classification groups and material names. You can also create your own database of common materials within this database tool, which can be saved and loaded for other projects. Database materials are stored in a file with extension ".GPF."

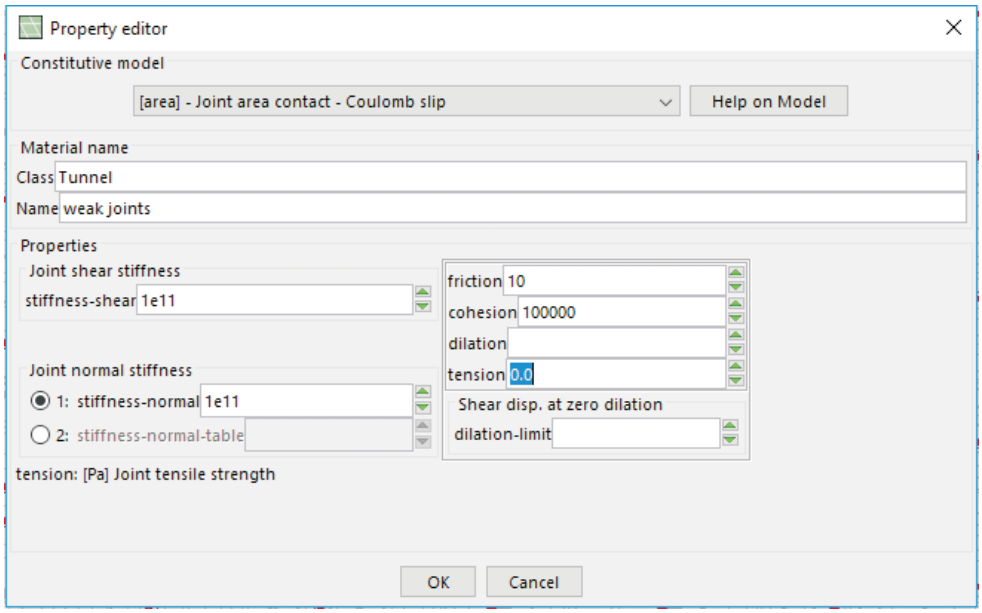

Figure 2.20 Joint Property editor dialog showing the weak joints material and properties

First, assign the strong joint material to all joints in the model. (The weak joints will be assigned later in this analysis.) The strong joint material is assigned to all the joints in the model by pressing the SET ALL button in the JOINTMAT tool, as shown in Figure 2.21. Next, press EXECUTE to send the joint material property commands to *UDEC*.

The next step is to assign the boundary conditions for our model. Select the APPLY tool from the IN SITU modeling stage. Roller boundary conditions are applied to the bottom and sides of the model. To prescribe a roller boundary on the sides, select the XVELOCITY item from the outer boundary list. Hold down the left mouse button while dragging the mouse to create a box encompassing the left boundary. Then press the APPLY BY RANGE button to open the *Set* dialog. The x-velocity is zero for the boundary fixed in the x-direction and, by specifying this value in the dialog, the velocity is fixed at the selected gridpoints. Thus, you are preventing any movement in the x-direction. Press the OK button in the dialog to set the velocities. Then select XVELOCITY from the item list again and repeat the procedure to fix the right boundary in the x-direction. Finally, select YVELOCITY from the item list and apply a zero y-velocity to the bottom boundary. Figure 2.22 shows the APPLY tool and the *Set* dialog. Press EXECUTE to send these commands to *UDEC*. Figure 2.23 shows the *model-view* pane with the boundary conditions applied.

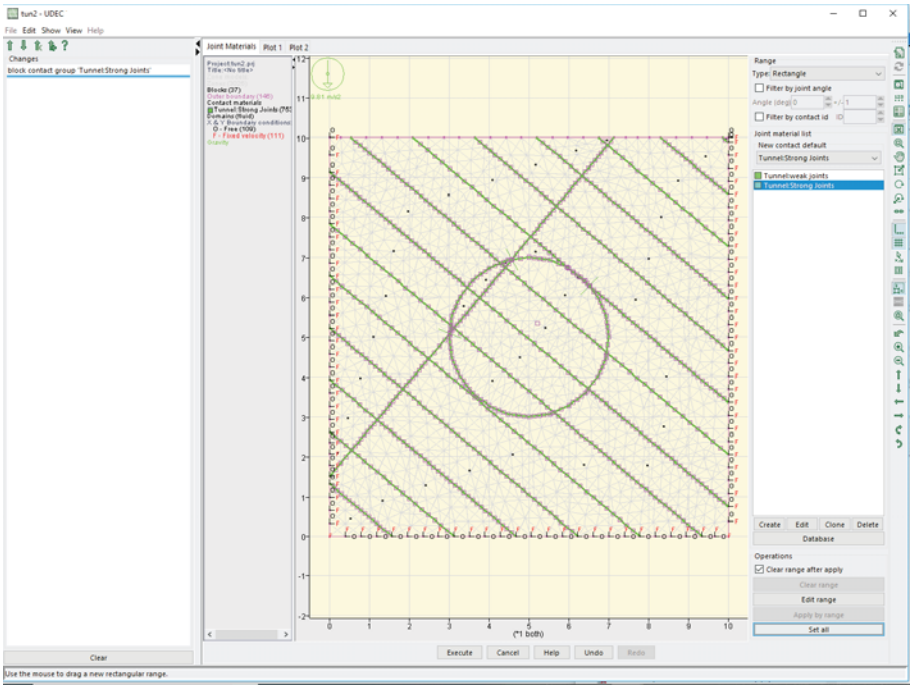

Figure 2.21 Assignment of joint properties in the JOINTMAT *tool*

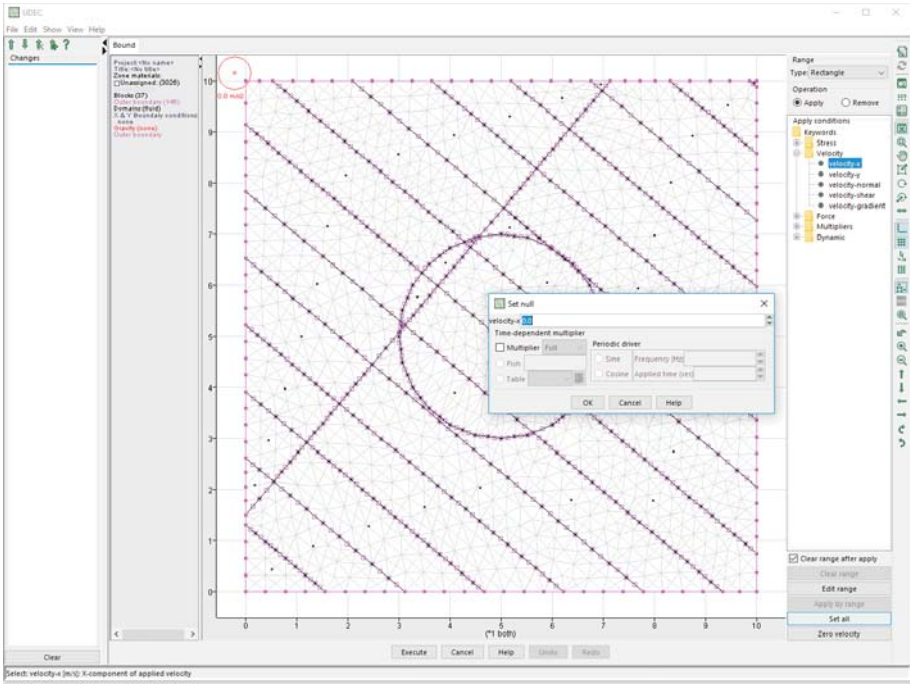

Figure 2.22 APPLY *tool*

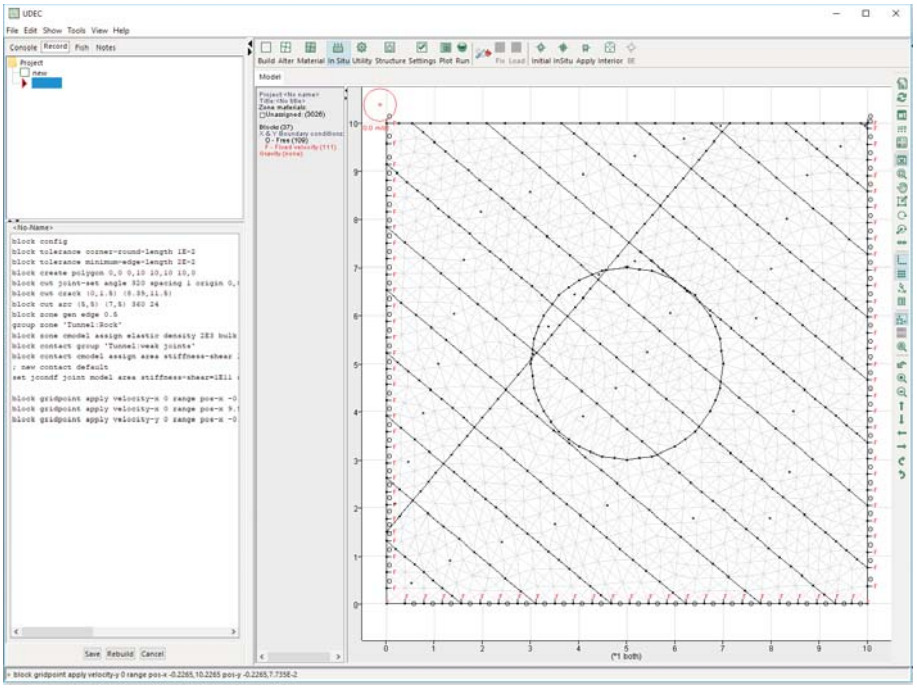

Figure 2.23 Model-view pane with boundary conditions applied

You can access different variables in this model from the UTILITY modeling stage. You wish to monitor the vertical displacement at the crown of the tunnel. To do this, open the HISTORY tool and then click on the GRIDPOINT Data Type radio button. Then select the YDISPLACE history. Then point the mouse at a gridpoint within the block at the crown of the tunnel. (Be careful not to select a gridpoint on a block that will be deleted later in the simulation.) When you click on the gridpoint, a **block gridpoint history** command is created for the *y*-displacement history at that gridpoint. Figure 2.24 shows the result of your actions in the HISTORY tool. Press EXECUTE to send the command to *UDEC*.

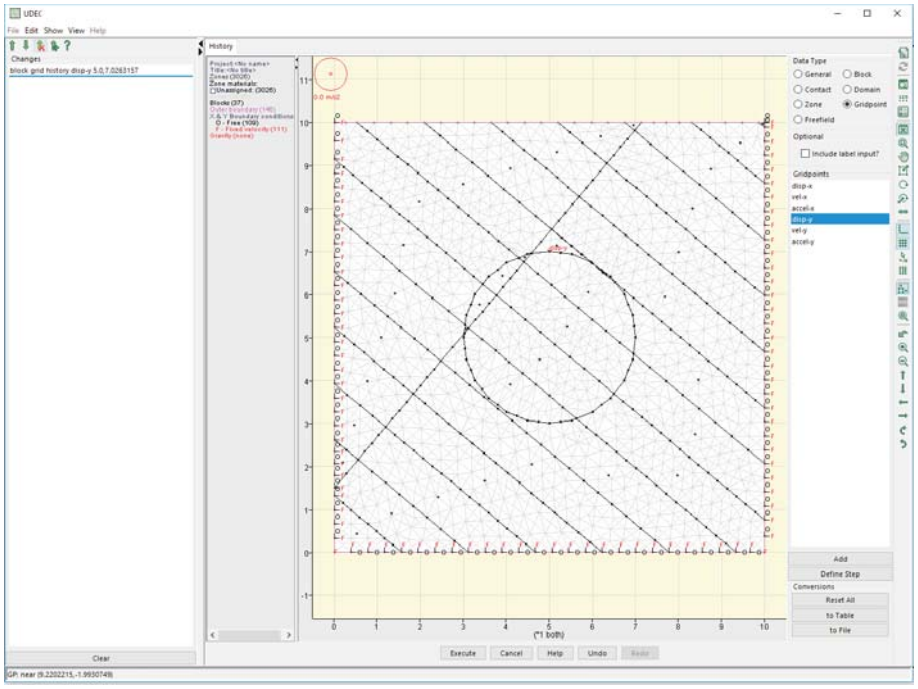

Figure 2.24 Select gridpoint y-displacement at the tunnel crown to monitor in the HISTORY *tool*

Gravitational loading is specified as a global setting in your model in the SETTINGS modeling stage. You can click on the GRAVITY tool to access the *Gravity settings* dialog. Then, by clicking on the globe icon in the dialog, the value of 9.81 m/sec^2 is listed as the magnitude of gravitational acceleration. (You can also type in a different value for the magnitude.) The dialog is shown in Figure 2.25:

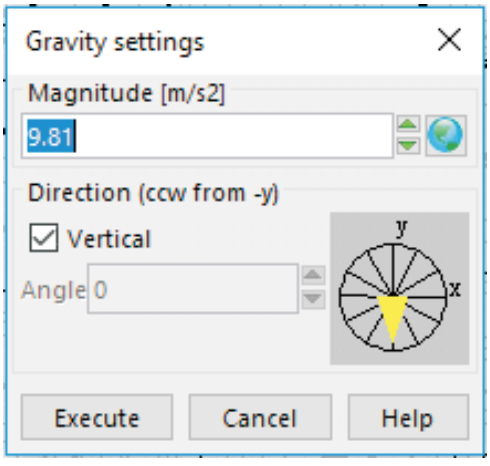

Figure 2.25 Set gravity settings in the Gravity settings dialog

You are now ready to bring the model to an initial equilibrium state. Cycle the model to a force equilibrium condition under gravity loading. Select the RUN modeling stage, and then the SOLVE tool. This opens a *Solve options* dialog as shown in Figure 2.26; press EXECUTE and invoke the **block solve** command to detect equilibrium automatically. A *Model cycling ...* dialog now appears, and the timestep number, maximum unbalanced force and equilibrium ratio are displayed. The equilibrium ratio is used to determine equilibrium. When the ratio falls below the default limiting value of 10^{-5}, the calculation stops.

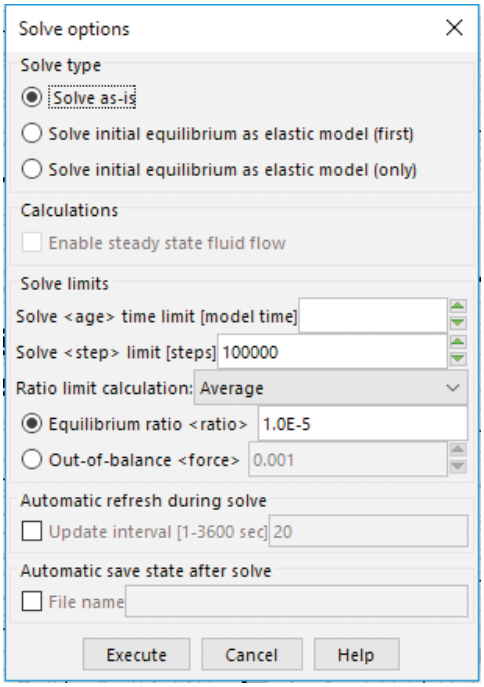

Figure 2.26 Solve options dialog from RUN stage tool

It is a good idea to save the project state at the different stages of your analysis. In this way you can easily return to a given state and make modifications without the need to run the entire simulation again. You can save the project model state at the initial-equilibrium stage by pressing the SAVE button at the bottom of the *Record* pane. This opens a dialog box (as shown in Figure 2.27) that allows the user to give a descriptive title to the saved state, select a folder in which to save the model-state file, and name the file. By default, the file has the extension ".SAV."

You choose to save the model state as "TUN1.SAV." You must save this file in the same folder as the project file "TUNNEL.PRJ," so that the project can be opened later and all associated save files accessed. The save file name is added to a list at the top of the *Record* pane. (This is the "project tree.") Each time you save the model state, a new save file name will be added to the list. You can click on any file name in the list to open that saved state.

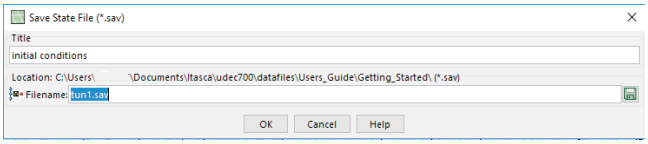

Figure 2.27 Save State File dialog

There are several ways to make sure that equilibrium has been reached. A quick check can be done by plotting the change in maximum unbalanced force during stepping. In the PLOT modeling stage, select the HISTORY button, and then the UNBALANCED FORCE plot item. A plot of unbalanced force versus calculation cycle will appear. The plot given in Figure 2.28 shows that the maximum unbalanced force is approaching zero, which indicates that an equilibrium state has been reached. If the unbalanced force history approaches a nonzero value, this is an indication that a portion of the model is failing. Another good way to check whether the model is in equilibrium or failing is to create a plot of velocity vectors in the model. A concentrated region of "large" velocity vectors (greater than approximately 10^{-5}) typically identifies the portion of the model that is failing.

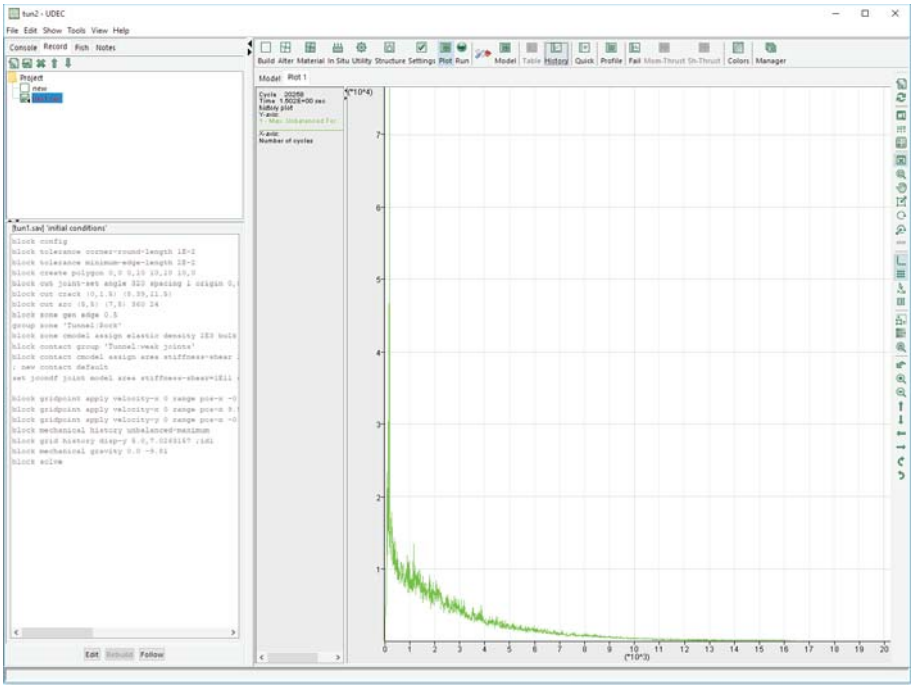

Figure 2.28 History of maximum unbalanced force

You can create plots for a wide variety of variables in a *UDEC* model. Click on the MODEL tool from the PLOT stage to open the *Plot items* dialog box. The dialog is shown in Figure 2.29. From this dialog you can access most of the general plotting facilities in *UDEC*. (Note that separate tools are provided for table, history, profile and failure plots in the PLOT stage.)

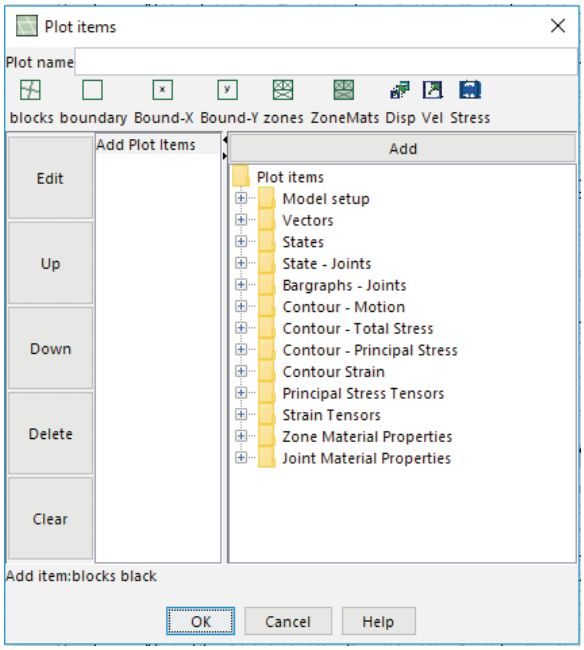

Figure 2.29 Plot items dialog

For example, if you wish to examine the gravitational stresses that develop in the model, you can create a contour plot of $\sigma - yy$-stresses. Click on the CONTOUR-TOTAL STRESS/SYY plot item from the *Plot items* tree, and add this to the *Add Plot Items* list. Then click on the MODEL SETUP/BLOCK plot item and add this to the list. You can either create a fill-contour plot or a line-contour plot. By default, a filled contour plot is created with the contour range denoted by the fill colors. The resulting fill-contour plot is shown in Figure 2.30.

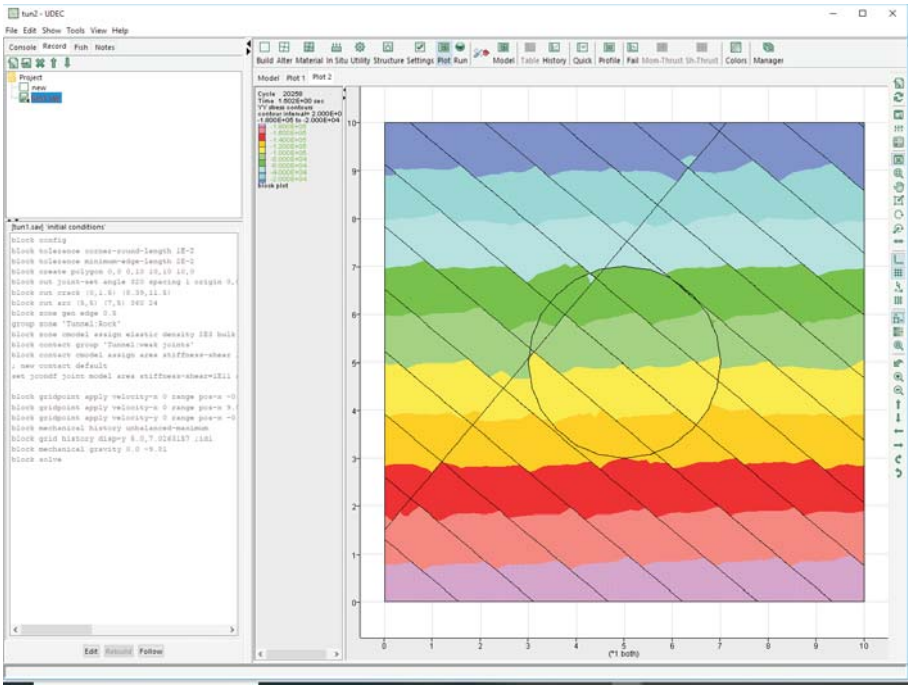

Figure 2.30 Contour plot of $\sigma - yy$-stresses resulting from gravitational loading

You can make a hardcopy plot of any *UDEC* model plot you choose. To do this, click on the FILE/PRINT PLOT menu item in the main menu. The *Print plot* dialog opens, as shown in Figure 2.31. If a printer is available, you can send the currently active plot view directly to the printer by clicking the OK button in this dialog.

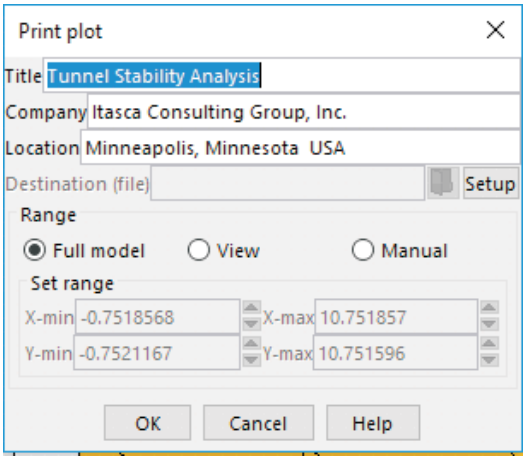

Figure 2.31 Print plot dialog

The SETUP button can be pressed in the *Print plot* dialog to change the printer device settings. Figure 2.32 shows the *Print setup* dialog that opens when SETUP is pressed.

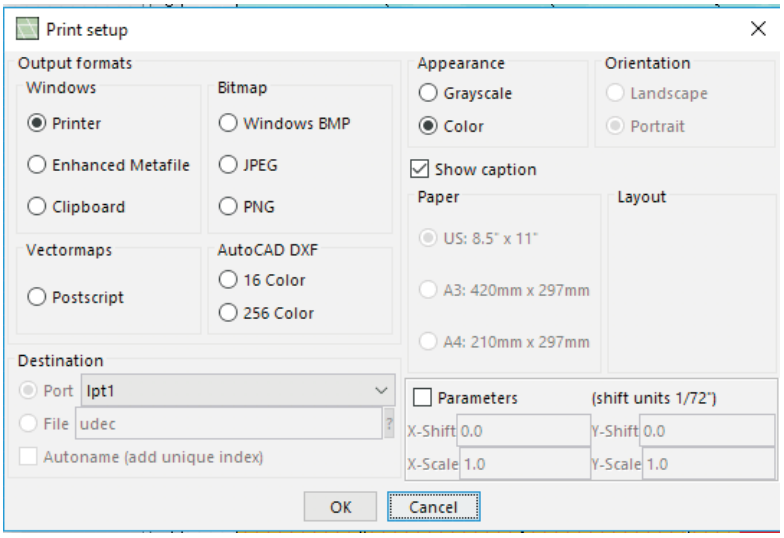

Figure 2.32 Print Setup dialog

You are now ready to excavate the tunnel. You should select the DELETE&ADD tool from the ALTER stage. In order to excavate the tunnel, click on Disk in the *Range* drop-down list and drag the mouse to encompass the centroids of the blocks to be deleted in the tunnel area. A disk shape covering the blocks to be deleted will be created, as shown in Figure 2.33. Press the APPLY BY RANGE button; these blocks are now removed from the block plot, and **block delete** commands corresponding to the deleted blocks are listed in the *Changes* pane. Press EXECUTE to send the **block delete** commands to *UDEC*.

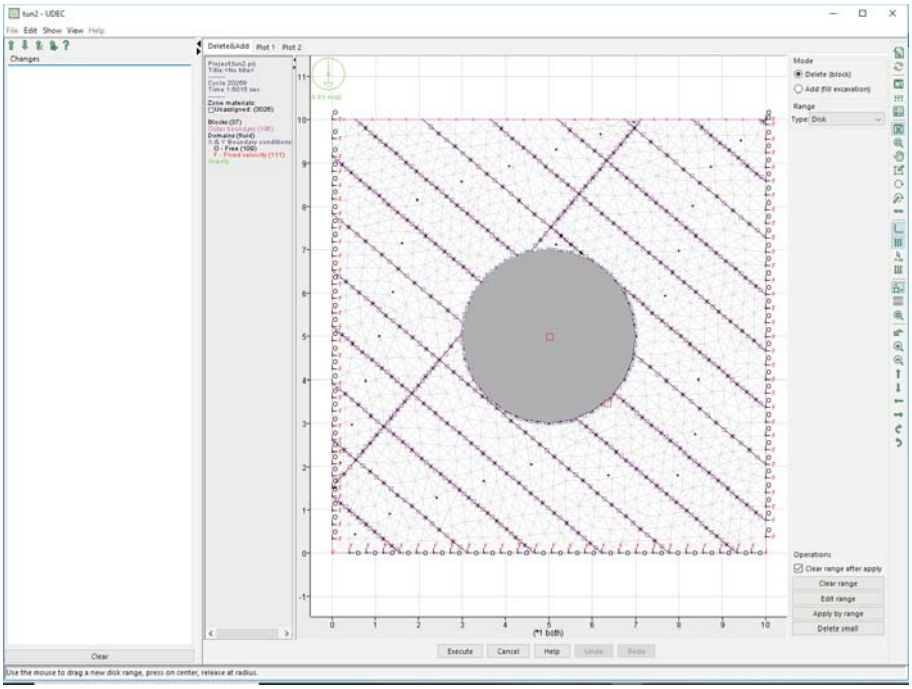

Figure 2.33 UDEC model with tunnel excavated in DELETE&ADD tool

Save the model state at this stage; "TUN2.SAV" is added to the *Record* tree.

Now evaluate the behavior of the rock with strong joints. You can perform this analysis by using the SOLVE tool, as you did to determine the initial equilibrium state. A stable solution state is calculated, and the resulting displacements around the tunnel are illustrated by the *y*-displacement contour plot shown in Figure 2.34. This plot is created by clicking on the CONTOUR-MOTION/YDISPLACE plot item from the *Plot items* tree.

Save the model state at this stage as "TUN3.SAV."

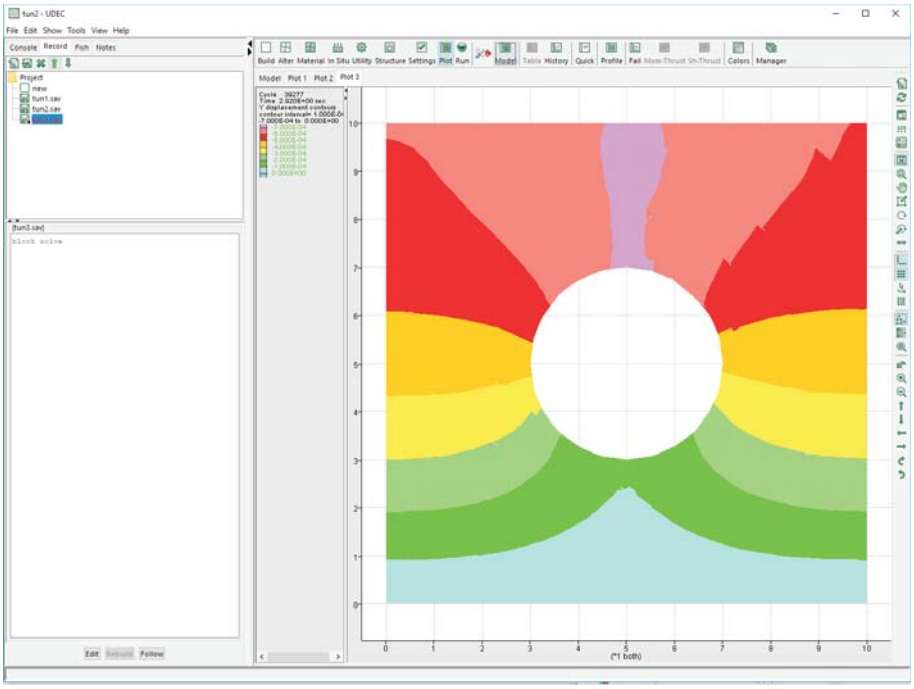

Figure 2.34 Rock with strong joints: y-displacement contours

Next, you can evaluate the jointed rock response following excavation of the tunnel in rock with weak joints. You first need to return to a previous model stage. Return to the state before the tunnel blocks are removed ("TUN1.SAV") by double-clicking on this file name in the *Record* pane. Now press the FOLLOW button at the bottom of the pane. The project tree will now create two "branches." *branch A* contains the save files related to excavation in rock with strong joints, and "branch B" is a new branch that you will use to perform the simulation in rock with weak joints. See Figure 2.35. Note that both branches start from the state "TUN1.SAV."

Return to the MATERIAL stage and open the JOINTMAT tool. Select *Tunnel: weak joints* in the material list. Then click on the SETALL button to change all of the joints to weak joints.

The initial stress state with weak joints can be different from that with strong joints. Therefore, the initial stress state should be recalculated after the joint properties are changed, by using the RUN/SOLVE tool. Figure 2.36 plots the unbalanced force history after SOLVE is applied. The new equilibrium state is saved as "TUN4.SAV."

Delete the tunnel blocks, as before for the strong joints case. "TUN5.SAV" saves the state at this stage.

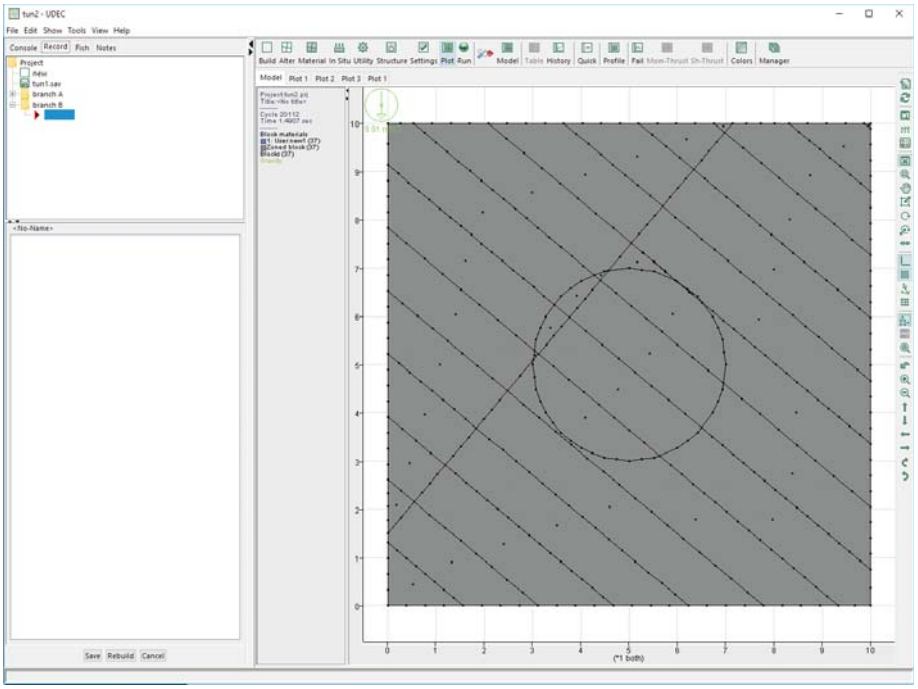

Figure 2.35 Creation of two branches in project tree when the FOLLOW *button is pressed*

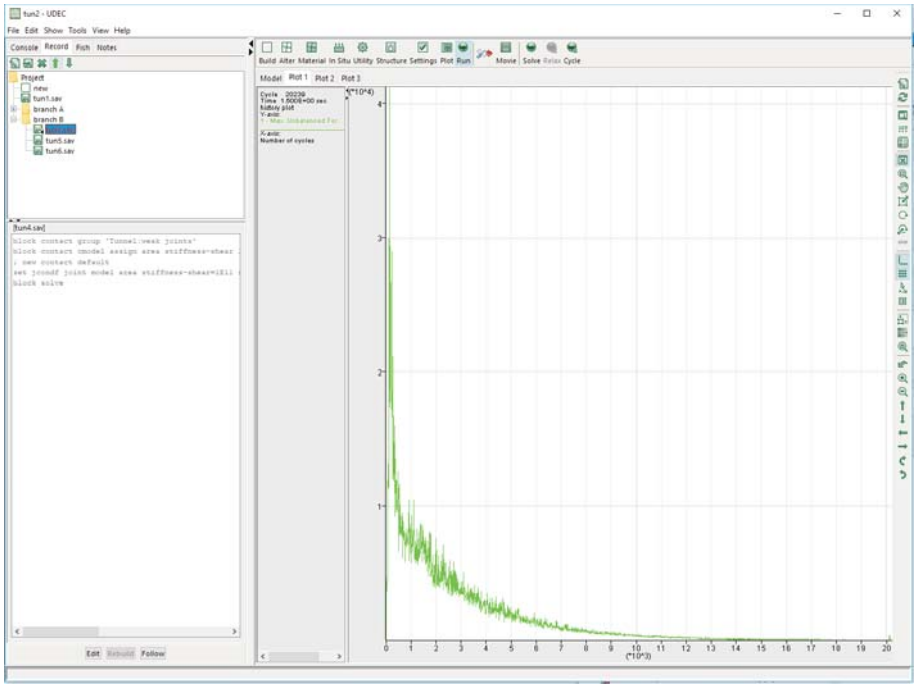

Figure 2.36 Maximum unbalanced force history

Now you should use the RUN/CYCLE tool to calculate the model response and evaluate the state with weak joints. Enter 10,000 calculation steps as the step limit in the *Cycle* dialog, as shown in Figure 2.37. In this way, you can evaluate the condition of the model during the solution process.

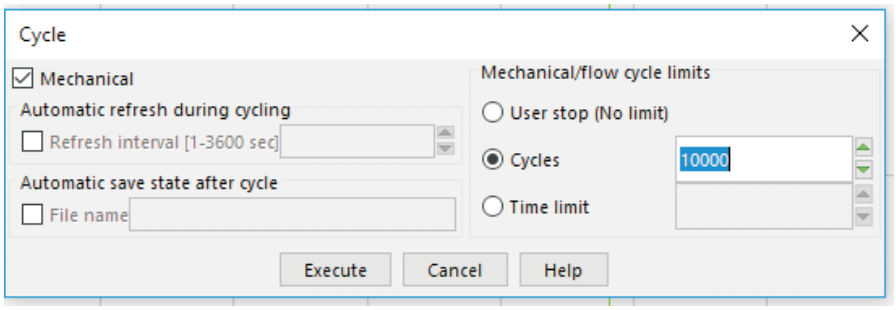

Figure 2.37 Enter cycle limit in the Cycle dialog

After 10,000 calculation steps, you notice that the triangular-shaped block in the crown of the tunnel has become detached from the surrounding rock. See Figure 2.38. The movement of this block is also evident from the y-displacement history recorded at a gridpoint on the block, as shown in Figure 2.39. This plot is created by selecting the HISTORY tool in the PLOT stage. Then, click on *Item ID* number 1 (which is the history number corresponding to the y-displacement history you selected), and press OK to create the plot.

Note that you can magnify the region of interest on the plot by simply rotating the scroll wheel on your mouse. You can also translate the model view by holding down the scroll wheel and dragging the mouse in the direction desired.

You save the state at this stage in "TUN6.SAV."

You can follow the movement of the roof block until it reaches the invert of the tunnel. Continue the simulation by selecting RUN/SOLVE. This time, change the equilibrium ratio to 10^{-6} in the *Solve options* dialog to ensure that the run does not stop prematurely when the roof block hits the tunnel invert. The run should now proceed until the block comes to rest at the invert. Figure 2.40 displays the model at this stage. If you check the AUTOMATIC box in the *Model cycling ...* dialog, the *model-view* will be updated during the calculation so that you can follow the movement of the roof block. Note that the cell-space detection logic is used in this case to ensure that the new contact will be detected when the roof block reaches the tunnel invert.

Finally, save this model stage as "TUN7.SAV" in branch B. This will also automatically update the file "TUNNEL.PRJ." You can move this project file and save files to a different folder and restore the project again, if you wish to make additional plots or perform other alterations and simulation runs.

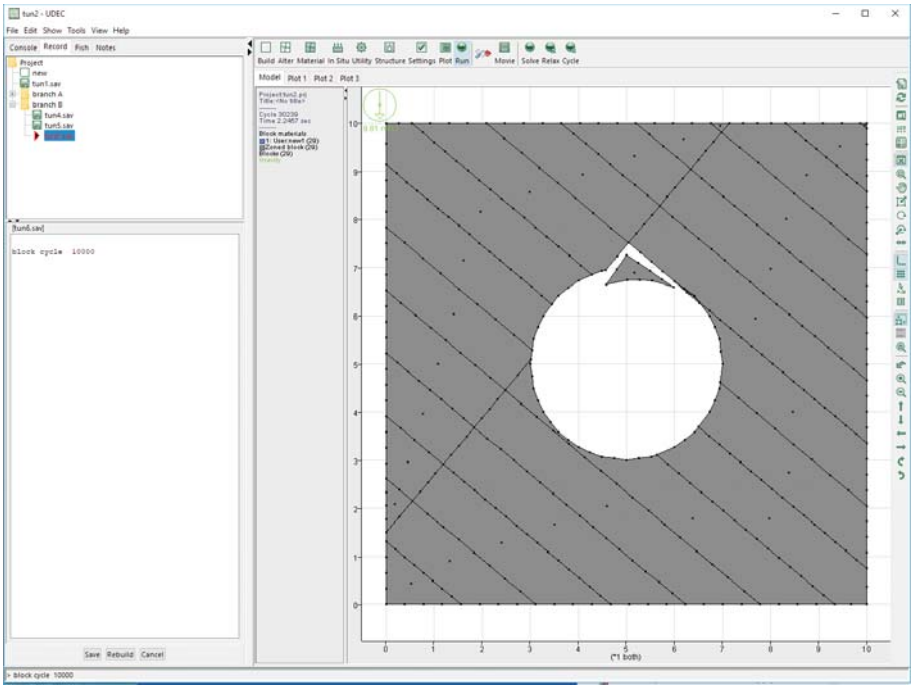

Figure 2.38 Roof block detaches after tunnel excavation in rock with weak joints

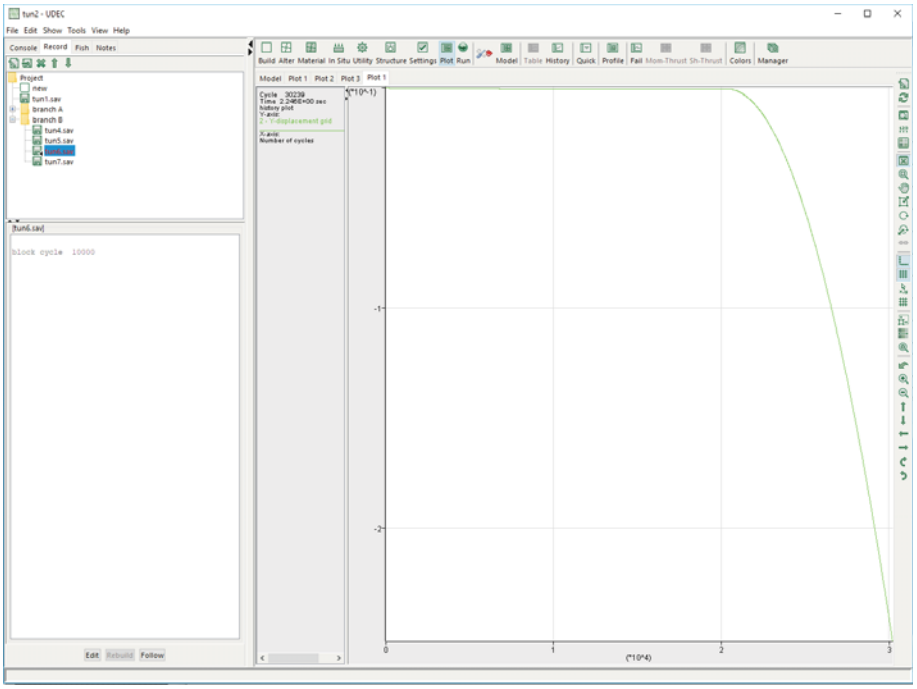

Figure 2.39 y-displacement history of gridpoint in roof block after tunnel excavation in rock with weak joints

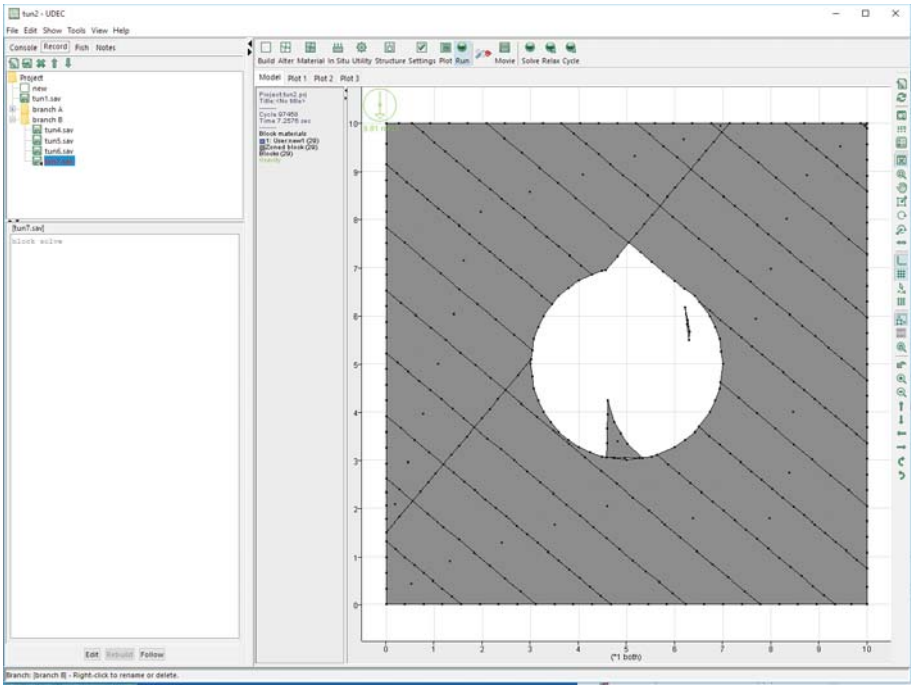

Figure 2.40 Roof block hits invert of tunnel

This completes the *GIIC* tutorial. We recommend that you now try variations of this example to become more familiar with the *UDEC-GIIC* operation. For example, begin with the "TUN5.SAV" model state, after the tunnel blocks have been deleted in the rock with weak joints, and create a new branch to add cable elements along the tunnel crown and simulate the support provided by rockbolts. You can add structural elements via the *Model Options* dialog after restoring the project state. You can now evaluate which rockbolt characteristics and properties are required to support the roof block.

2.2.3 Running UDEC in Command-Driven Mode

UDEC operates as a command-driven program. The *GIIC* provides a tool to assist users with the generation of commands from a graphical mode. Users can also enter the commands directly in the GUI version. *UDEC* can then be run in command-driven mode, either interactively or from an input data file. Launching *UDEC* from the *UDEC* 7.0 link will execute *UDEC* in the GUIas show in Figure 2.41:

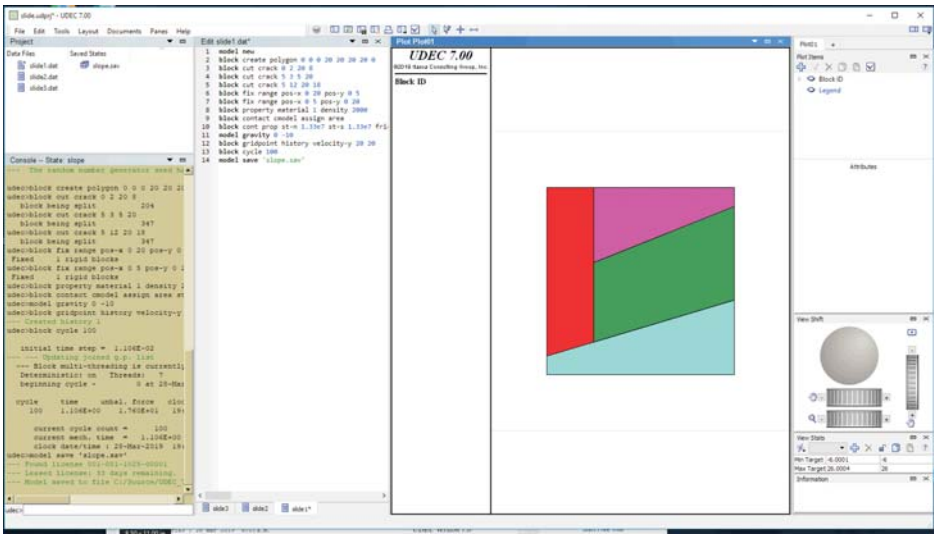

Figure 2.41 UDEC Command-line Graphical Interface

The GUI version operates through the creation of a project file which holds links to data files that contain the *UDEC* commands. It is possible to type the commands individually in the command prompt, bu this is tedious and difficult to reproduce. *UDEC* will execute each command as the <ENTER> key is pressed. If an error arises, an error message will be written to the screen.

As an alternative, an input data file may be created using a text editor (see Section 2.1.5). This file contains a set of commands, just as they would be entered in the interactive mode. Although the data file may have any name, a common identifying extension (e.g., ".DAT") will help to distinguish it from other *UDEC* files (see Section 2.10).

The data file can be read into *UDEC* by file->open into project or file->add new data file.

For more details on operating *UDEC* in GUI mode look in the *UDEC* Help->program guide->program mechanics.

2.2.4 A Simple Tutorial – Use of Common Commands

This section is provided for the new user using the command-driven mode in the GUI. A simple example is presented to help you learn some of the basic aspects of solving problems with *UDEC*.

The example is a four-block slope stability problem. We evaluate the stability for different conditions of joint friction. The project file "slide.udprj" included in the folder ITASCA\UDEC700 \Datafiles\Users_Guide\Getting_Started\Slide_tutorial, contains all of the commands we are about to enter interactively.)

We can run this problem interactively (i.e., by typing the commands from the keyboard, pressing <ENTER> at the end of each command line, and seeing the results directly). To begin, load *UDEC* 7.0.

The problem begins by specifying a single block using the **block create** command:*

```
block create polygon (0,0)  (0,20)  (20,20)  (20,0)
```

This command creates a square block with side lengths of 20 units (in this case, meters). To see the block, select a block plot from the plot item list.

A picture of the block will appear on the screen. If the round attribute is selected the corners appear slightly rounded. The rounding is used in *UDEC* to give reasonable physical behavior to blocks that just slightly overlap each other. The rounding length may be adjusted by the user.

The problem is continued by splitting the initial block into smaller blocks by typing

```
block cut crack  (0,2)  (20,8)
block cut crack  (5,3)  (5,20)
block cut crack  (5,12)  (20,18)
```

These commands split existing blocks along the line with endpoints specified by coordinates in parentheses.

If you wish to print the plot, right click on the plot and select plot->print plot and the plot will be sent to a printer. To create a plot file, right click on the plot and select export->bitmap or one of the other output formats.

Next, the lowermost and leftmost blocks are immobilized by typing

```
block fix all range position-x 0,20 position-y 0,5
block fix all range position-x 0,5  position-y 0,20
```

This command fixes the current velocity (i.e., zero) of all blocks with centroids in the range $0 < x < 20$, $0 < y < 5$ and $0 < x < 5$, $0 < y < 20$.

---

* See Help in *UDEC* for further details. Note that command words can be abbreviated and separated by any number of spaces and "=" characters

Then, required material properties are assigned to a property number for the blocks and joints by typing

```
block prop mat=1  dens=2000
block contact cmodel assign area  st-n=1.33e7  st-s=1.33e7  fric=20.0
```

For this problem, the mass density of all blocks is specified to be 2,000 units (kg/m^3, in this case). Note that the mass density, *not* the unit weight of the block material, is assigned. All joints are specified to have contact normal (**st-n**) and shear (**st-s**) stiffness equal to 1.33×10^7 (here, Pa/m) and friction angles equal to 20°. As will be seen later, different properties can be assigned to various joints and intact blocks.

Next, gravitational accelerations in x- and y-directions are specified by typing

```
block mech gravity  0  -10
```

In order to absorb vibrational energy, damping is introduced by typing

```
block mechanical damp local
```

This is the default damping condition in *UDEC*, and the **block mechanical damp local** command is actually not required here. We use it only to emphasize that this is a static analysis.

At this point, the problem is ready to be executed. As will be seen later, it is often helpful to judge behavior (i.e., equilibrium, stability and instability) by observing the motion of specified points in the rock mass. In this problem, we monitor the y-velocity of a point at the top-right corner of the model. The command used to record this motion is

```
block gridpoint history  vel-y  (20,20)
model display history 1
```

Following execution of this command, the program returns information about the selected monitoring point (20,20). The keyword **model display history 1** instructs the program to print the value (in this case, the y-velocity of point (20,20)) on the screen at specified intervals.

One hundred calculation cycles are executed by typing

```
block cycle 100
```

During execution, the current cycle count, the calculation time, the maximum out-of-balance force, the y-velocity of point (5,20) and the clock time are printed on the screen every 10 cycles. Inspection of these values indicates that equilibrium has been obtained. (The velocity and out-of-balance force approach zero.) A graphical representation of this behavior is obtained by selecting a history plot from the plot item menu and then selecting tha appropriate history.

```
model title  'A SIMPLE SLOPE STABILITY EXAMPLE: EQUILIBRIUM STAGE'
```

to view the title on the plot select the check box on the plot item tool bar and check Job Title.

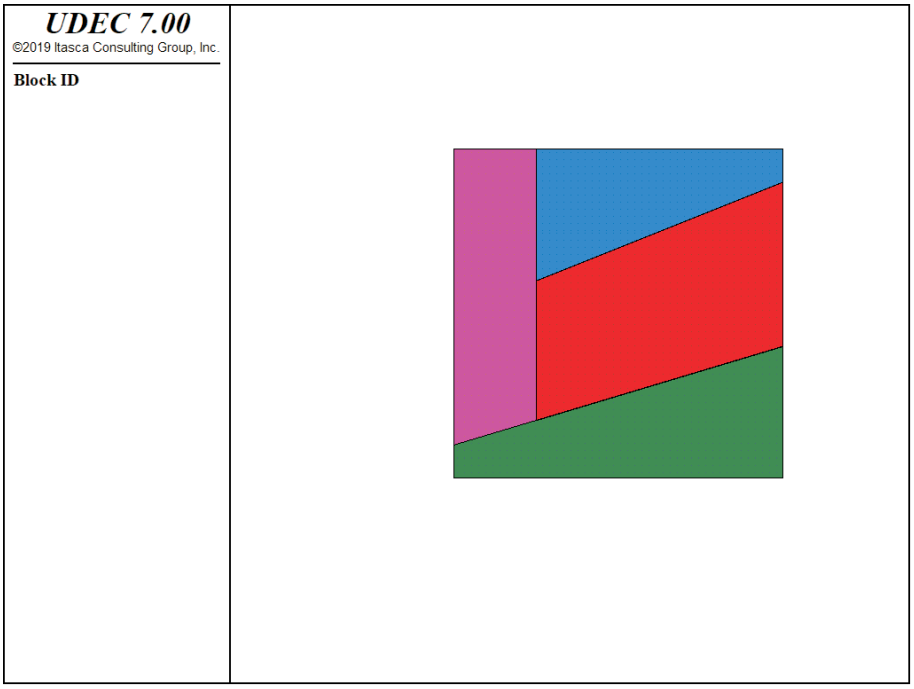

Figure 2.42 A simple slope stability example: equilibrium stage (friction angle = 20°)

It is often helpful to save this initial state so that it can be restarted at any time (for example, to perform parameter studies). To save the current state (in a file called "SLOPE.SAV"), type

```
model save  'slope.sav'
```

The behavior of the slope can be studied by removing the leftmost block by typing

```
block delete range pos-x 0,5  pos-y 0,20
```

This command deletes blocks with centroids in the range $0 < x < 5$, $0 < y < 20$. Next, the calculation process continues using the **block step** or **block cycle** command. The problem state after 1000 additional cycles (1100 cycles total) is shown in Figure 2.43.

The figure shows that only the top block is sliding. This is the expected result because the friction angle (20°) is less than the slope (22°) of the joint between the two uppermost blocks. A plot of the y-velocity history of the monitored point also indicates that the block is sliding; the velocity is increased at a constant rate.

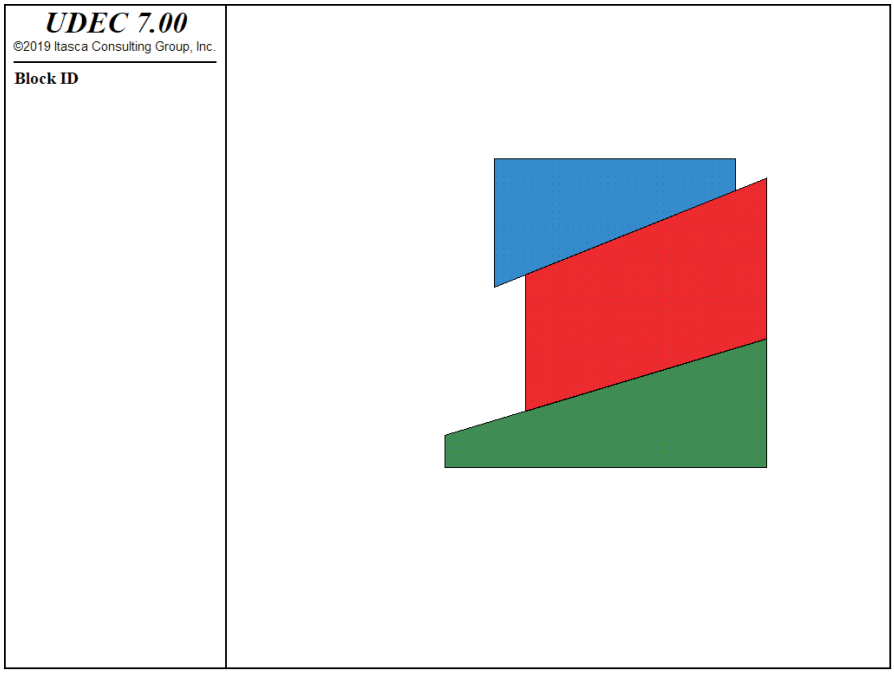

Figure 2.43 A simple slope stability example: one block sliding (friction angle = 20°)

The problem may be continued in the manner previously described, but it is interesting to examine the effect of other choices of problem parameters at this point. The initial save state may be recalled by typing

```
model restore 'slope.sav'
```

and selecting "SLOPE.SAV" from the file-open dialog. The leftmost block is removed as before, but in this case the joint friction angle is reduced to 11°. The following command sequence results in Figure 2.44.

```
block delete  range pos-x 0,5  pos-y 0,20
block contact property fric=11
block cycle  1000
```

As seen in Figure 2.44, both blocks are sliding. (The top block is sliding faster than the middle block.) This is the expected result because the friction angle (11°) is less than the joint slopes (22° for the top and 17° for the bottom).

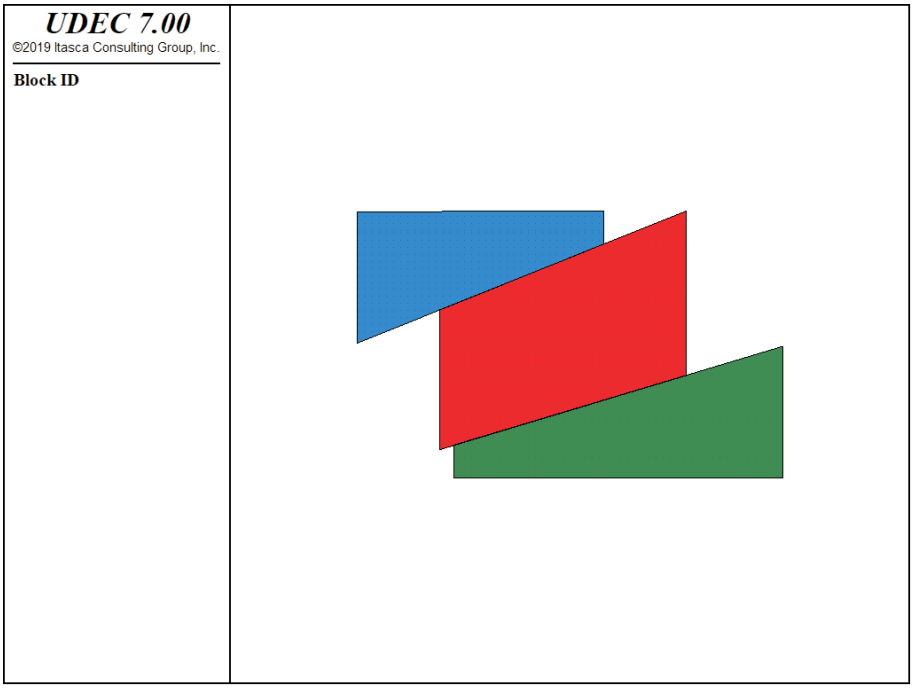

Figure 2.44 A simple slope stability example: two blocks sliding (friction angle = 11°)

This concludes the initial tutorial. In the following sections, we will present other features of *UDEC*. We recommend that you read the rest of this section for a beginner's guide to the mechanics of using *UDEC*. As you become more familiar with the code, turn to Section 3 for additional details on problem solving with *UDEC*.

2.3 Nomenclature

The nomenclature used in *UDEC* is similar, for the most part, to that used in continuum stress analysis programs. In addition, though, special terminology is used to describe the discontinuum features in a *UDEC* model. The basic definitions are given here for clarification. Figure 2.45 is provided to illustrate *UDEC* terminology.

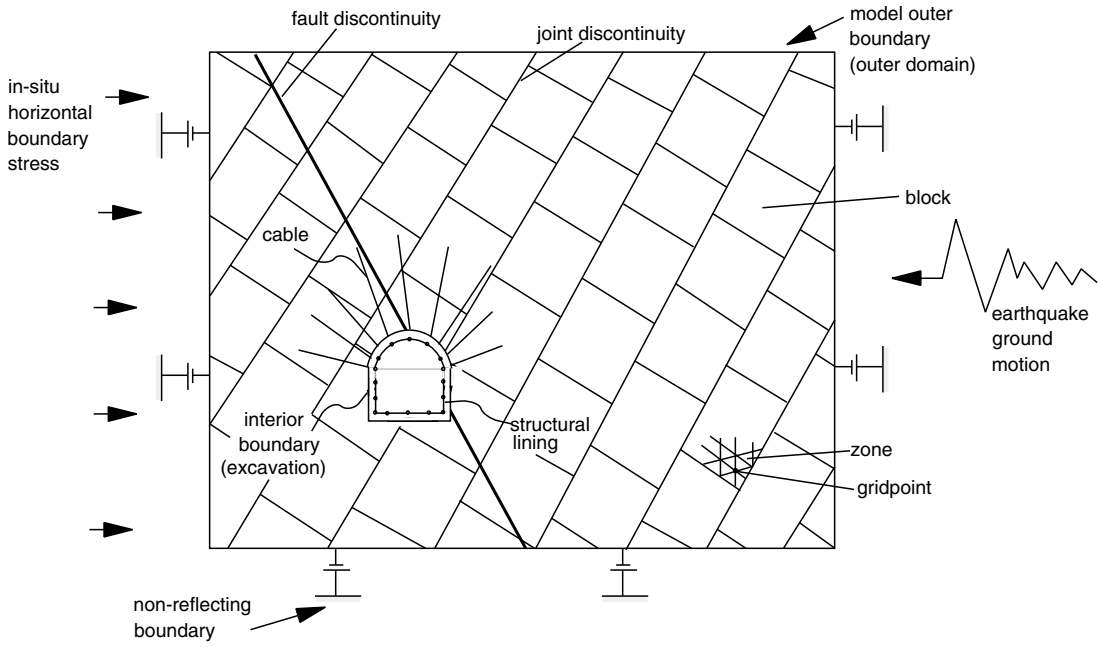

Figure 2.45 Example of a UDEC model (not to scale)

UDEC MODEL – The *UDEC* model is created by the user to simulate a physical problem. When referring to a *UDEC* model, we imply a sequence of *UDEC* commands (see Help in *UDEC* .) Section 1 in the **Command Reference**) which define the problem conditions for numerical solution.

BLOCK – The block is the fundamental geometric entity for the distinct element calculation. The *UDEC* model is created by "cutting" a single block into many smaller blocks. Each block is an independent entity that may be detached from other blocks or may interact with other blocks via contact forces.

CONTACT – Each block is connected to adjacent blocks via point contacts. A contact may be considered a boundary condition which applies external forces to each block. The term "joint" may also be used to refer to a single contact or a set of contacts.

DISCONTINUITY – A discontinuity is a geologic feature that separates a physical mass into distinct parts. Discontinuities include, for example, joints, faults and fractures and other discontinuous features in a rock mass.

To be represented in *UDEC*, a discontinuity must have a length scale which is approximately of the same order as the engineering structure being analyzed. A discontinuity in *UDEC* is defined by two or more contacts.

JOINT ID – Each time an operation cuts a block into two blocks, the contacts that are formed between the blocks are tagged with an ID number. The number may be assigned by the user. If not assigned, the ID number will be automatically incremented with each command. Contacts formed by the same command will have a common ID number.

DOMAIN – A domain is the void or space between blocks. Domains are treated as physical entities in the *UDEC* model; each domain is an enclosed region defined by two or more contacts. The "outer domain" is the region surrounding the *UDEC* model. Note that a domain exists between the adjoining faces of two overlapping blocks, even though the domain area appears to be negative (see Section 1.2.4 in **Theory and Background**). Contact detection is performed within domains. Fluid flow occurs between domains (see Section 2 in **Special Features**). Domains are not created if **block config cell** is used.

ZONE – Deformable blocks are composed of finite-difference zones. Mechanical changes (e.g., stress/strain) and thermal changes (i.e., heat transfer) are calculated within each zone. Triangular zones are used in *UDEC*.

GRIDPOINT – Gridpoints are associated with the corners of the finite-difference zones. There are always three gridpoints associated with each zone. A pair of x- and y-coordinates is defined for each gridpoint, thus specifying the exact location of the finite-difference zones. Another term for gridpoint is nodal point or node.

GROUP – A group is a gathering of objects. Blocks, zones and contacts may be members of a group. Groups are referenced by a text name. Group names may be used as input to ranges.

MODEL BOUNDARY – The model boundary is the periphery of the *UDEC* model. This boundary coincides with the outer domain of the model. Internal boundaries (i.e., holes within the model) are also model boundaries. Each internal boundary is defined by an internal domain.

BOUNDARY CONDITION – A boundary condition is the prescription of a constraint or controlled condition along a model boundary (e.g., a fixed displacement or force for mechanical problems, or adiabatic boundary for heat transfer problems).

INITIAL CONDITIONS – This is the state of all variables in the model (e.g., stresses) prior to any loading change or disturbance (e.g., excavation).

NULL BLOCK – Null blocks are blocks which represent voids (i.e., no material present) within the model. Null blocks can be made "real" later in an analysis – for example, to simulate backfilling. (Once a block is deleted from a model it cannot be restored, however it may be recreated.)

STRUCTURAL ELEMENT – Structural elements are one-dimensional elements used to represent the interaction of structures (such as tunnel liners, rockbolts, cable bolts or support props) with a rock mass. Material nonlinearity is possible with structural elements. Geometric nonlinearity occurs in large-strain mode.

STEP – Since *UDEC* is an explicit code, the solution to a problem requires a number of computational steps. During computational stepping, the information associated with the phenomenon under investigation is propagated across the blocks in the model. A certain number of steps is required to arrive at an equilibrium (or steady-flow) state for a static solution. Typical problems are solved within 2000 to 4000 steps, although large, complex problems can require tens of thousands of steps to reach a steady state. When doing a dynamic analysis, **block step** or **block cycle** refers to the actual timestep for the dynamic problem. Other terms for step are timestep and cycle.

STATIC SOLUTION – A static or quasi-static solution is reached in *UDEC* when the rate of change of kinetic energy in a model approaches a negligible value. This is accomplished by damping the equations of motion. At the static solution stage, the model will either be at a state of force equilibrium or steady flow of material if a portion (or all) of the model is unstable (i.e., fails) under the applied loading conditions. This is the default calculation mode in *UDEC**, and can also be invoked with the **block damp local** command. Static mechanical solutions can be coupled to transient fluid-flow or heat-transfer solutions.

UNBALANCED FORCE – The unbalanced force indicates when a mechanical equilibrium state (or the onset of joint slip or plastic flow) is reached for a static analysis. A model is in exact equilibrium if the net nodal force vector at each block centroid or gridpoint is zero. The maximum nodal force vector is monitored in *UDEC*, and printed to the screen when the **block step**, **block cycle** or **block solve** command is invoked. The maximum nodal force vector is also called the "unbalanced" or "out-of-balance" force. The maximum unbalanced force will never exactly reach zero for a numerical analysis. The model is considered to be in equilibrium when the maximum unbalanced force is small compared to the representative forces in the problem. If the unbalanced force approaches a constant nonzero value, this probably indicates that joint slip or block failure and plastic flow are occurring within the model.

DYNAMIC SOLUTION – Fully dynamic analysis can be performed by inhibiting the default static solution damping. For a dynamic solution, the full dynamic equations of motion (including inertial terms) are solved; the generation and dissipation of kinetic energy directly affect the solution. Dynamic solutions are required for problems involving high frequency and short duration loads (e.g., seismic or explosive loading).

---

* Some finite element (FE) literature includes the mistaken notion that a dynamic solution method cannot produce a true equilibrium state, while an FE solution is believed to perfectly satisfy the set of governing equations at equilibrium. In fact, *both* methods only satisfy the equations approximately, but the level of residual errors can be made as small as desired. In *UDEC*, the level of error is objectively quantified as the ratio of unbalanced force at a block centroid or zone gridpoint to the mean of the set of absolute forces acting at that point. This measure of error is very similar to the convergence criteria used in FE solutions. In both cases, the solution process is terminated when the error is below a desired value.

LARGE-STRAIN/SMALL-STRAIN – By default, *UDEC* operates in large-strain mode: block coordinates and gridpoint coordinates are updated at each step, according to computed displacements. *UDEC* can also be run in small-strain mode, in which case coordinates are not changed, even if computed displacements are large.

2.4 The *UDEC* Model: Subdivision of the Initial Block

The creation of a *UDEC* model begins with a single block of a size that spans the physical region being analyzed. The model features are introduced by cutting this block into smaller blocks whose boundaries represent both the geologic structure (e.g., faults, bedding planes and joint structure) and engineered structures (e.g., underground excavations and tunnels).

All blocks in the model are defined by the x- and y-coordinates of their corners and centroid. Contacts between blocks, as well as gridpoints within deformable blocks, are also defined by their coordinate position. Model generation involves cutting the model block along lines ("splits" or "cracks") whose endpoints are defined by xy-coordinates.

All entities of the *UDEC* model (i.e., blocks, corners, contacts, domains, gridpoints and zones) are uniquely identified by an address number in the main data array, allocated automatically by *UDEC*. These numbers may also be used to refer to a particular entity. The numbering system is not sequential for each entity, so the user must identify the number via a plot or printout.

For example, Figure 2.46 illustrates a *UDEC* model block of dimensions 10 units (say, meters) in the x-direction and 10 units in the y-direction. The model block is divided into two blocks separated by a horizontal discontinuity located from (x = 0, y = 5) to (x = 10, y = 5). The two blocks have block numbers 202 and 342. The blocks are connected by two contacts located at the corners of the blocks. The contact numbers are 459 and 499. An internal domain is created by the two contacts and is identified by domain number 539.

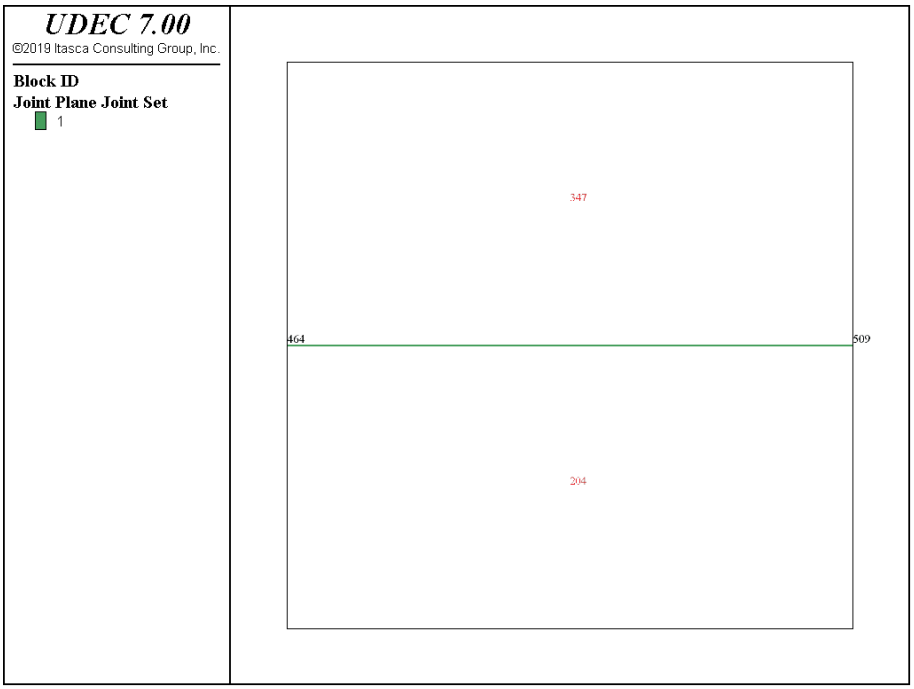

Figure 2.46 UDEC model block divided into two rigid blocks

The model shown in Figure 2.46 was created with the commands listed in Example 2.1:

Example 2.1 Dividing a block

```
model new
block tolerance corner-round-length .01
block tolerance minimum-edge-length .02
block create polygon 0 0 0 10 10 10 10 0
block cut crack 0 5 10 5
model save 'ex2_01.sav'
```

The two blocks may be made deformable by creating finite-difference zones in each block. Figure 2.47 shows the zone numbers if the upper block of Figure 2.46 is divided into 8 triangular zones and the lower block into 4 triangular zones. A new contact (with number 985) is also created between the two blocks. A corner is always created whenever a gridpoint is located on a block edge. The new contact 985 corresponds to the new corner created along the edge of the upper block.

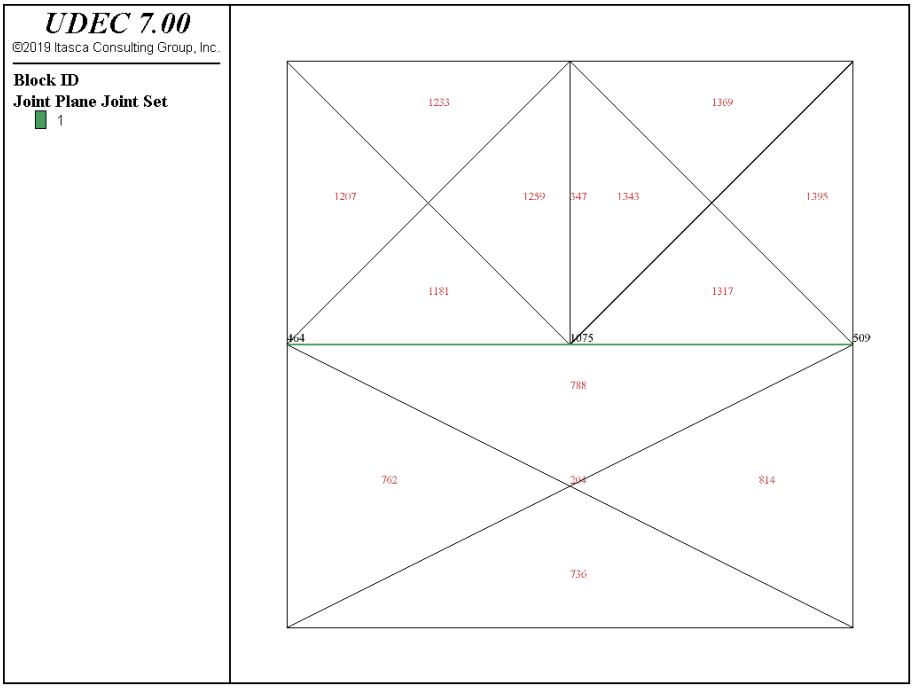

Figure 2.47 UDEC model block containing two deformable blocks

Example 2.2 Code to produce model shown in Figure 2.47

```
model restore 'ex2_01.sav'
block zone gen quad 11,6 range pos-x 0,10 pos-y 0,5
block zone gen quad 10
```

```
model save 'ex2_02.sav'
```

The index numbers also act as pointers to storage locations of all state variables in the model. Data associated with each entity in the model are stored with that entity number. For example, block forces, velocities and displacements for rigid blocks are stored with each block number. For deformable blocks, vector quantities (e.g., forces, velocities and displacements) for a block are stored with gridpoint numbers, while scalar and tensor quantities (e.g., stresses and material property numbers) are stored with zone numbers. Contact data such as contact force, velocity and flow rates are stored at contact numbers, while data such as fluid pressure and fluid volume are stored at domain numbers. *FISH* can be used to access *UDEC* data via the address numbers. See FISH in the Help in *UDEC* for lists of the variables that can be accessed.

2.5 Command Syntax

All input commands* to *UDEC* are word-oriented. The command syntax has been designed in a way that commands in all Itasca codes will behave in a similar manner. To differentiate commands that are intended only for *UDEC* , most *UDEC* commands will begin with the word BLOCK. All commands in the program are words that consist of a primary word (a noun/object) and a first keyword (a verb/action). Because of the preceeding BLOCK word, the primary word may actually consist of two words (i.e. BLOCK ZONE, BLOCK CONTACT) preceeding the verb. To this foundation may be added options and modifiers (adverb(s)), and a range (a prepositional phrase). In other words, commands syntactically mimic natural language.

The latter components (options, modifiers, and ranges) are uniformly defined for all commands and are frequently optional. These parts also usually require user-supplied values to be completed.

The syntactical pattern is expressed as:

NOUN (NOUN) - VERB - OPTION(S) - MODIFIER(S) - RANGE

Note the order is consistent and required in all commands. That said, parts may be omitted in the middle when optional (that is, the pattern NOUN - VERB - RANGE is valid in commands where OPTION and MODIFIER components are optional).

The commands are typed literally on the command line. You will note that only the first few letters are underlined. The program requires these letters, at a minimum, to be typed to recognize the command; command input is not case-sensitive. The entire word for a command or keyword may be entered if the user so desires.

Many of the keywords are followed by a series of values which provide the numeric input required by the keyword. The decimal point may be omitted from a real value, but may not appear in an integer value.

Commands, keywords and numeric values may be separated by any number of spaces or by any of the following delimiters.

( ) , =

A semicolon (;) may be used to precede comments; anything that follows a semicolon in an input line is ignored. It is useful, and strongly recommended, to include comments in data files. Not only is the input documented in this way, but the comments are echoed to the output as well, providing the opportunity for quality assurance in your analysis.

There is no maximum length of a single command. For readability commands may be split into several lines separated by an elipsis (...) or an ampersand (&).

Please note that the typographical conventions listed in Table 2.2 are used throughout this manual.

---

* The commands and their meanings are presented in the Help in *UDEC* .

Table 2.2 Typographical conventions

Type style	Used for
bold	*UDEC* commands and *FISH* statements
bold	*UDEC* keywords and *FISH* internal variables and functions
bold	user-defined *FISH* variables and functions
var	placeholders for variables
<A>	type the key between < > (here, <A>) on the keyboard
<SHIFT-A>	hold down the first key while pressing the second (here, <SHIFT> and the <A> key)

For users who are importing data files from a version of *UDEC* prior to *UDEC* 7.0, there is an automated conversion tool supplied in the *UDEC* GUI version (see edit->Command Conversion). This tool will also convert *UDEC* GIIC project files. In addition there is a list of old commands and the new equivalent in the *UDEC* GUI help under "UDEC 6.0 to 7.0 Commands Map" and "UDEC 6.0 to 7.0 FISH Mapping".

2.6 Mechanics of Using *UDEC*

UDEC is based on a command-driven format. Word commands control the operation of the program. This section provides an introduction to the basic commands a new user needs to perform simple *UDEC* calculations. If you have not done so already, run the tutorial problem in Section 2.2.4 for an example of command-driven analysis with *UDEC*.

All of the commands in *UDEC* can be accessed from the graphical interface. We recommend that you use the *GIIC* (see the introduction in Section 2.2.1) for ease of operation while learning the mechanics of using *UDEC*. You can follow the examples in this section either by entering the word commands at the `udec:` prompt in the text mode, or by point-and-click operation in the graphical mode. In the latter case, the commands will be created by the *GIIC* for you to check as you follow the example. The tutorial in Section 2.2.2 illustrates this procedure using the *GIIC*.

The example files in this section are listed in the "ITASCA\UDEC700\UsersGuide\GettingStarted" folder. The data files (with extension ".DAT") can be read into *UDEC* by using the **call** command in the command-line mode. Alternatively, the project files (with extension ".PRJ") corresponding to these data files can be called into the *GIIC* using the FILE / OPEN PROJECT menu item. Also, the project files (with extension ".UDPRJ") corresponding to these data files can be called into the GUI using the FILE / OPEN PROJECT menu item.

In order to set up a model to run a simulation with *UDEC*, three fundamental components of a problem must be specified:

(1) a distinct-element model block with cuts to create problem geometry;

(2) constitutive behavior and material properties; and

(3) boundary and initial conditions.

The model block defines the geometry of the problem. The constitutive behavior and associated material properties dictate the type of response the model will display upon disturbance (e.g., deformational response due to excavation). Boundary and initial conditions define the in-situ state (i.e., the condition before a change or disturbance in problem state is introduced).

After these conditions are defined in *UDEC*, an alteration is made (e.g., excavate material or change boundary conditions), and the resulting response of the model is calculated. The actual solution of the problem is different for an explicit-solution program like *UDEC* than it is for conventional implicit-solution programs. (See the background discussion in Section 1 in **Theory and Background**.) *UDEC* uses an explicit time-marching method to solve the algebraic equations. The solution is reached after a series of computational steps. In *UDEC*, the number of steps required to reach a solution is controlled manually by the user. The user ultimately must determine whether the number of steps is sufficient to reach the solved state. See Section 2.6.4 for ways this can be done.

The general solution procedure for an explicit static* analysis with *UDEC* is illustrated in Figure 2.48. This procedure is convenient because it represents the sequence of processes that occur in the physical environment. The basic *UDEC* commands needed to perform simple analyses with this solution procedure are described in the following pages.

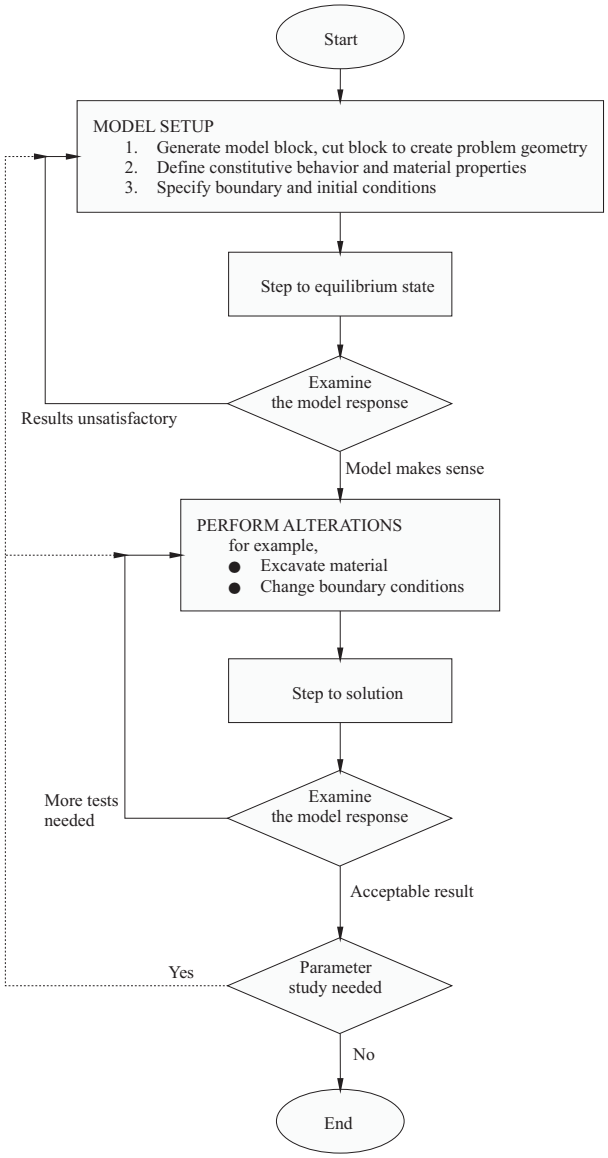

Figure 2.48 General solution procedure for static analysis

---

* Dynamic analysis with *UDEC* is discussed in Section 4 in **Special Features**.

2.6.1 Block Cutting

The *UDEC* model is created by cutting the original *UDEC* block into smaller blocks that represent boundaries of physical features in the problem. The model block is made with the command

```
block create polygon  x1,y1  x2,y2  x3,y3 ...
```

where (x_1,y_1), (x_2,y_2), (x_3,y_3) are coordinate pairs that define the corners of the block. The pairs must be entered in a *clockwise* order. The corners should be located to coincide with the boundary of the physical problem. The block can have many corners, but it is usually easiest to start with a four-corner block.

All blocks in *UDEC* have "rounded" corners. As explained in Section 1.2.4 in **Theory and Background**, this prevents blocks from hanging up at sharp corners; block hang-up causes erroneous stress concentrations to occur in the model. However, there is an upper limit to the corner rounding, and this limit is problem-dependent. For deformable blocks, the maximum rounding length should not exceed one percent of the average block edge length of blocks in the region of interest for the analysis (e.g., blocks in the vicinity of an underground excavation).* The round length is changed with the command

```
block tolerance corner-round-length d
```

where ***d*** is the rounding distance. (The default value is ***d*** = 0.5.) The rounding length is the same for all blocks in a model.

It is recommended that the rounding length be specified *before* the **block create** command. The effect of corner rounding can be seen by selecting a block plot fromt he plot item list and selecting rounding in the plot item attributes. after the **block create** command is given.

There are several commands available to cut the *UDEC* block in order to create geometries in the model. Two primary commands are used to create geologic structure (e.g., joints):

```
block cut crack
block cut joint-set
```

The **block cut crack** command creates a single straight-line fracture in the block. The crack is defined by its endpoints $(x1,y1)$ and $(x2,y2)$.

The **block cut joint-set** command invokes an automatic joint set generator; a set of cracks that is defined by characteristic parameters (i.e., dip angle, trace length, gap length, spacing and spatial location) is created.

Both **block cut crack** and **block cut joint-set** can create discontinuous fractures in the *UDEC* block (i.e., the fracture does not have to completely separate the block in two). However, *UDEC* requires continuous fractures (i.e., all fractures *must* completely intersect blocks). The locations of discontinuous fractures created by a **block cut crack** or **block cut joint-set** command are stored. These

---

* When operating in the *GIIC*, the rounding length is set by default to 0.1% of the largest block edge length.

fractures can then be used with those created by subsequent cracking commands to create continuous fractures through blocks. Cracks that do not connect to form complete blocks will be deleted at the time of execution for a rigid block model, or at the time of zone generation for a deformable block model.

The following examples illustrate block cutting with the **block cut crack** and **block cut joint-set** commands. The complete descriptions for these commands are given in Help in *UDEC* .

Joint generation is explained in more detail in Section 3.2.2.

Example 2.3 Creating a simple UDEC model

```
model new
block tolerance corner-round-length 0.1
block create polygon 0,0 0,10 10,10 10,0
block cut crack 0,5 10,5
block cut crack 2.5,10.0 5.0,7.5
block cut crack 5.0,7.5  7.5,10.0
model save 'ex2_03.sav'
```

In its simplest form, block cutting involves specifying cracks at selected locations in the block. By typing the commands, a square block of dimensions 10 units by 10 units will be created and then split into two blocks, each 5 units by 10 units. The **block cut crack** command creates a continuous horizontal crack through the model. Note that the rounding length is specified to be 0.1 unit.

A notch may be created in the top block by typing

```
block cut crack 2.5,10  5,7.5
blcok cut crack 5,7.5   7.5,10
```

These two **block cut crack** commands create discontinuous cracks in the upper block, but together they make a continuous crack (notch) in the upper block.

By selecting a block plot from the pot item meun and selecting the data index from the attributes a plot is made of the blocks, including block index numbers (see Figure 2.49).

A block may be deleted from the model via the **DELETE** command. For example, to delete the notch, type either

```
block delete range id-list 634
```

or

```
blco delete range pos-x 4.5,5.5  pos-y 8,10
```

The range $4.5 < x < 5.5$ and $8 < y < 10$ must include the centroid of the block which is to be deleted. Note that it is advisable to use a coordinate range when performing some operations on the model; the use of addresses is problem-dependent (e.g., the notch block's number will change if the lower block is split before the notch is created).

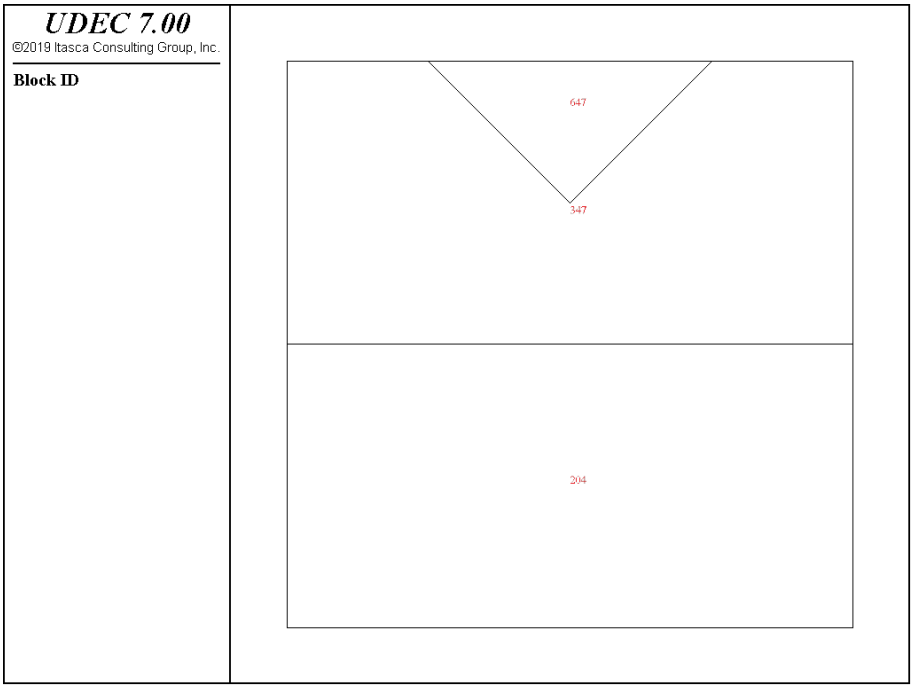

Figure 2.49 Block model with horizontal crack and notch

The parameters for the **block cut joint-set** command are entered as keywords with parameter pairs: the first number in the pair is a mean value, and the optional second is the maximum standard deviation from the mean (based on a uniform probability distribution). The first keyword is the angle of the joint track from the positive global x-axis, the second keyword is the trace length of the joint segment, the third keyword is the gap length between segments, and the fourth keyword is the spacing normal to joint tracks. There are additional optional parameters that may be given to assist in creating more complex joint patterns. The general use of the **block cut joint-set** command is discussed in Section 3.2.2. The use of **block cut joint-set** for the generation of two continuous joint sets is illustrated in Example 2.4:

Example 2.4 Generation of two continuous joint sets

```
model new
round 0.01
block create polygon 0,0 0,10 10,10 10,0
block cut joint-set angle 45,0 trace 20,0 gap 0,0 spac 2,0
block cut joint-set angle -10 spac 1.5
model save 'ex2_04.sav'
```

The first **block cut joint-set** command creates a continuous joint set oriented at an angle of 45° from the positive x-axis with a spacing between joints of 2 units. The second **block cut joint-set** command creates a continuous set oriented at $-10°$ from the x-axis with a spacing of 1.5 units. (Optional

parameters are omitted in the second **block cut joint-set** command.) The choice of rounding length may affect the orientation of the joint sets; the joint locations may be altered because blocks cannot be created with edge lengths smaller than two times the rounding length.

Select block from the plotitem menu to see the results of the **block cut joint-set** commands (see Figure 2.50). If the rounding length is increased (to, say, 0.1) before invoking **block cut joint-set**, the location of some joints in the model will be altered.

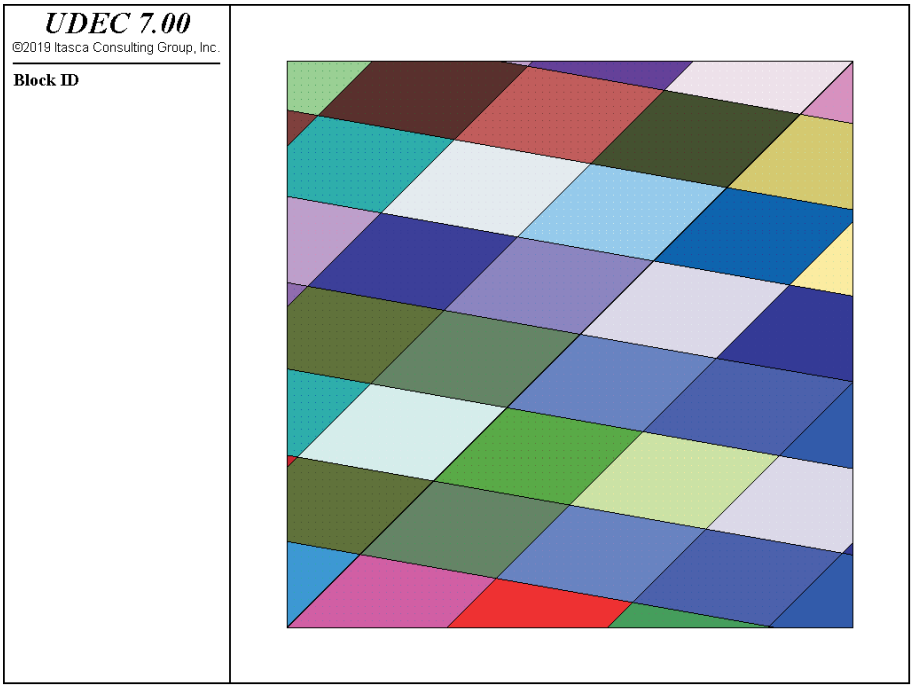

Figure 2.50 Block model with two continuous joint sets created with **block cut joint-set**

Creation of joints with the **block cut joint-set** command can involve some trial-and-error. Recommendations for implementing the joint generation stage are provided in Section 3.

When creating many blocks with greatly varying sizes, it is recommended that very small blocks be deleted from the model to increase the calculation efficiency. In Example 2.4, block sizes range from 1.752×10^{-3} to 3.679. This is found by typing

```
block list max
```

The very small blocks can be deleted by typing

```
block delete range area 0 3e-2
```

All blocks with areas smaller than 3×10^{-2} will be deleted from the model. Usually, blocks smaller than roughly 1% of the largest blocks in the area of interest can be deleted without noticeably affecting model results.

Finally, note that the **model new** command is used in the second example to permit starting a new model without leaving *UDEC*. Also, note that command words can be abbreviated (i.e., **block**, **bl** and even **b** have the same meaning). This command syntax is described in Section 2.5.

An important consideration when cutting blocks (especially when using **block cut joint-set**) is that a balance must be struck between the number of blocks in the model and solution speed. The calculation speed to reach a solution varies directly as a function of the number of blocks (and number of zones if the blocks are deformable). As a rule of thumb, models containing up to roughly 1200 rigid blocks (or 500 deformable blocks with 8 degrees of freedom per block) will typically reach a static solution state for a prescribed alteration in approximately 2000 to 4000 cycles. On a i7 microcomputer, the runtime for a 500 deformable block model is roughly 3 seconds to perform 4000 calculation steps. Check the speed of calculation on your computer for the specific model to estimate the runtime required. A runtime benchmark test is provided in Section 5.

Shapes of engineered structures must also be cut in the *UDEC* block, and these must be created before model execution begins. The three commands commonly used to create shapes are

```
block cut crack
block cut tunnel
block cut arc
```

The **block cut crack** command has already been described. The **block cut tunnel** command creates a circular cut; the circle is composed of a user-specified number of crack segments. The **block cut arc** command creates a pattern of cracks that conform to an arc at a user-specified angle. These commands can be used in combination to create various shapes in the *UDEC* block. Example 2.5 presents the commands to generate a model of a fault intersecting a circular tunnel:

Example 2.5 Fault intersecting a circular tunnel

```
model new
block tolerance corner-round-length 0.1
block create polygon -10,-10 -10,10 10,10 10,-10
block cut tunnel 0,0  2  16
block cut crack -5,10 5,-10
model save 'ex2_05.sav'
```

The resulting model is shown in Figure 2.51. The circular tunnel is created with its centroid at ($x = 0$, $y = 0$), with a radius of 2 units, and with 16 crack segments defining the circle. Because the tunnel is completely inside the block, separate blocks cannot be created by using only the **block cut tunnel** command; a crack *must* intersect the edge of the original block to create new blocks. If the user attempted to execute *UDEC* with only the **block cut tunnel** command, as defined above, the tunnel cracks would be deleted before execution. By introducing the **block cut crack** command, a continuous crack that connects the tunnel cracks to the outer edges of the original block is created, thus creating blocks defining the tunnel and fault, as seen in Figure 2.51:

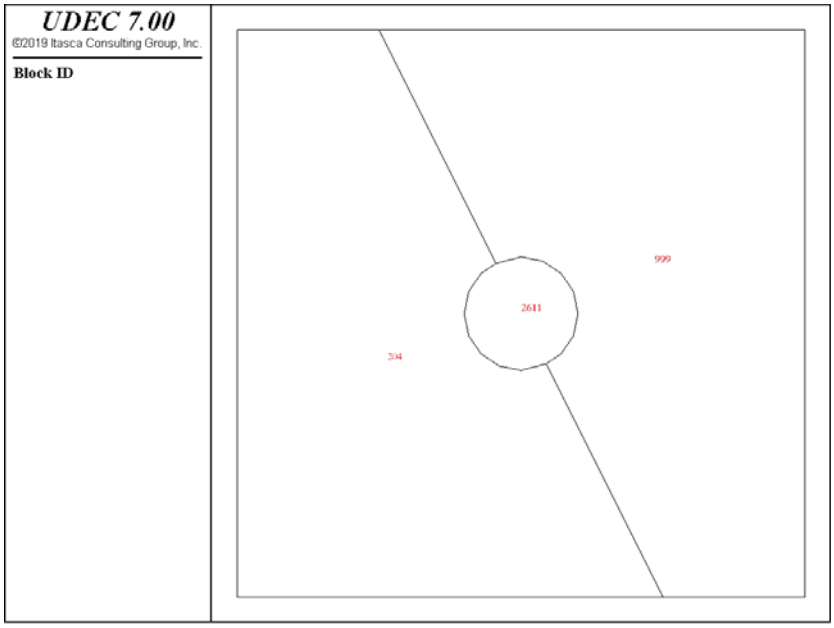

Figure 2.51 Fault intersecting circular tunnel

Note that the crack does not penetrate the tunnel periphery. If the **block cut tunnel** command is given first, any subsequent **block cut crack** or **block cut joint-set** command will not penetrate the tunnel. It is often more convenient to use **block cut tunnel** first, because excavation of the tunnel then only involves deleting one block – e.g.,

```
block delete range pos-x -1,1 pos-y -1,1
```

will cause the tunnel to be "excavated."

In Example 2.6, the fault intersects a horseshoe-shaped tunnel:

Example 2.6 Horseshoe-shaped excavation with intersecting fault

```
model new
block tolerance corner-round-length 0.1
block create polygon -10,-10 -10,15 10,15 10,-10
block cut arc 0,5 2,5 180  8
block cut crack -2,0 -2,5
block cut crack -2,0 2,0
block cut crack 2,0 2,5
block cut crack -5,15 5,-10
model save 'ex2_06.sav'
```

The tunnel shape is shown in Figure 2.52. The arc of the tunnel has a radius center at (x = 0, y = 5), a starting point at (x = 2, y = 5), an angle of 180°, and is made up of 8 crack segments. The

first three **block cut crack** commands create the sides and bottom of the tunnel. In this instance, the fault created by the last **block cut crack** command intersects the tunnel boundary. The tunnel may be excavated by deleting blocks by number or centroid location.

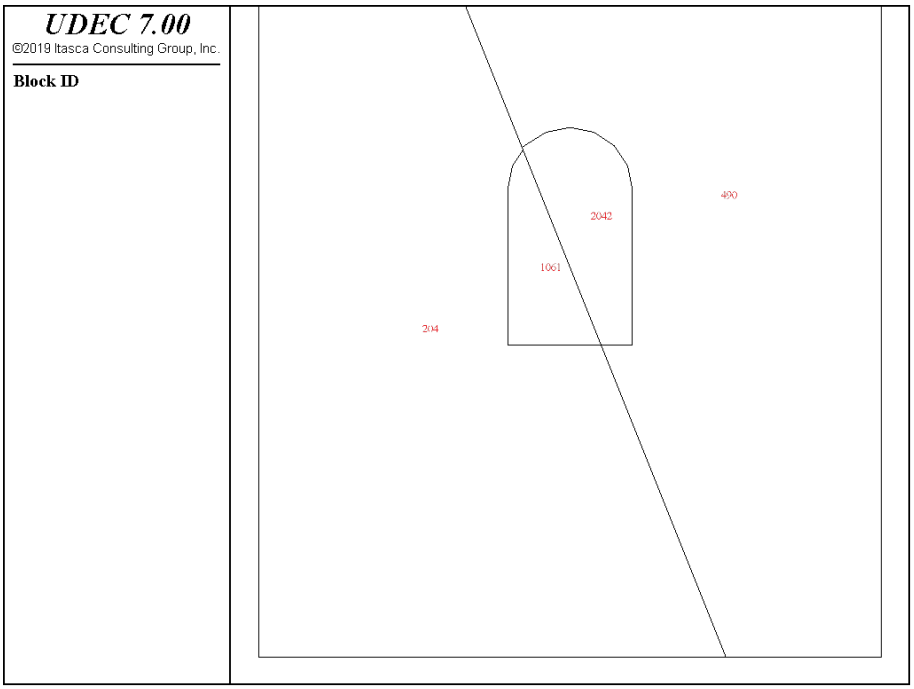

Figure 2.52 Horseshoe-shaped tunnel with intersecting fault

2.6.2 Assigning Material Models

2.6.2.1 Block Models

Once all block cutting is complete, material behavior models must be assigned for all the blocks *and* discontinuities in the model. By default, all blocks are rigid. In most analyses, blocks should be made deformable. The rigid block assumption can be applied only for cases in which stress levels are very low, or the intact material possesses high strength and low deformability. (For example, see the slope failure model in Section 2.2.4.)

Blocks are made deformable via the **block zone generate** command:

```
block zone generate edge v
```

or

```
block zone generate quad v
```

The **block zone generate** command invokes an automatic mesh generator that fills each block with triangular-shaped finite difference zones.*

The command **block zone generate edge** ***v*** will work for blocks of any arbitrary shape. The value ***v*** defines the maximum edge length of the triangular zones (i.e., the smaller the value for ***v***, the higher the density of zones in a block). Be careful, though, to not create zones that have a high aspect ratio; a practical limit on aspect ratio is approximately 1:10 for reasonable solution accuracy.

The command **block zone generate quad** ***v*** should be used if blocks are prescribed a plastic material model. This type of zoning provides a more accurate solution for plasticity problems (see Section 1.2.5 in **Theory and Background** for a description of this type of zoning). The **block zone generate quad** command may not work for all block shapes, however; if not, the **block zone generate edge** command should be used for the remaining blocks.

Once the blocks are made deformable, one or more material models and associated properties must be assigned to all zones in the blocks. This is done with the **block zone cmodel assign** command. There are several built-in material models for deformable blocks in *UDEC*; these are described in Section 1 in **Constitutive Models**. Three models are sufficient for most analyses the new user will make. These are assigned by the commands

```
block zone cmodel assign null    ; null model
block zone cmodel assign elastic ; elastic model
blcok zone cmodel assign mohr-c    ; Mohr-Coulomb model
```

block zone cmodel assign null represents material that is removed or excavated from the model. This allows the user to change the block back into an elastic or elastic-plastic material at a later stage in the model. If a block is deleted, it can be restored at a later stage if it is entirely inside the *UDEC* model (see the **block create fill** command).

block zone cmodel assign elastic assigns isotropic elastic material behavior, and **block zone cmodel assign mohr-c** assigns Mohr-Coulomb plasticity behavior.

block zone cmodel assign elastic and **block zone cmodel assign mohr-c** require that material properties be assigned via keywords following these commands. For the elastic model, the required properties are

(1) density;

(2) bulk modulus; and

(3) shear modulus.

NOTE: Bulk modulus, K, and shear modulus, G, are related to Young's modulus, E, and Poisson's ratio, ν, by

---

* There are variations of the **block zone generate** command that are available to improve the calculation accuracy for specific conditions. See the **block zone generate** command in Help in *UDEC* .

$$K = \frac{E}{3(1-2\nu)}$$
$$G = \frac{E}{2(1+\nu)}$$

or

$$E = \frac{9KG}{3K+G}$$
$$\nu = \frac{3K-2G}{2(3K+G)}$$

For the Mohr-Coulomb plasticity model, the required properties are

(1) density;

(2) bulk modulus;

(3) shear modulus;

(4) friction angle;

(5) cohesion;

(6) dilation angle; and

(7) tensile strength.

If any of these properties are not assigned, their values are set to zero by default.

For both the elastic and Mohr-Coulomb models, density, bulk modulus and shear modulus must be assigned positive values for *UDEC* to be able to execute.

As an example, properties are assigned for the Mohr-Coulomb model:

```
block zone cmodel assign mohr-c dens 2000 bu 1e9 sh 3e8 fric 25 coh 10000
```

2.6.2.2 *Joint Models*

In addition to block material models, a material model must also be assigned to all discontinuities (i.e., contacts) in the model. Joint models are assigned with the **JOINT model** command. There are four built-in constitutive models for discontinuities (summarized in Table 3.3).* The model sufficient for most analyses is the (elastic-perfectly plastic) area-contact Coulomb slip model, which is assigned to discontinuities with the command

```
block contact cmodel assign area
```

Material properties for the joint models are also assigned with the **JOINT model** command, following the model name. For the Coulomb slip model, the required properties are

(1) normal stiffness;

(2) shear stiffness;

(3) friction angle;

(4) cohesion;

(5) dilation angle; and

(6) tensile strength.

If any of these properties are not assigned, their values are set to zero by default. Normal and shear stiffnesses must be assigned positive values for *UDEC* to execute.**

For example, the following command assigns the Coulomb slip model with elastic normal and shear stiffness and friction angle properties:

```
block contact cmodel assign area st-n 1e10 st-s 1e10 fric 25
```

---

* The Barton-Bandis joint model is also available as an option.

** It is important that a material model and properties be assigned to any new contacts that may be created during a model run. This can be accomplished with the **block contact cmodel assign default** command, which allows the specification of a joint model and properties for any new contacts created during the run.

2.6.2.3 Example Block and Joint Model Assignment

Example 2.7 demonstrates the application of block and joint material models and properties. In this example, a single fault intersects a circular tunnel at an angle of −70° with respect to the x-axis. There is also a joint set dipping at −50° with a spacing of 3 m. The blocks are deformable and contain triangular zones with a maximum edge length of 2 m. Figure 2.53 shows the tunnel, joint structure and zoning within the blocks.

A group-name range is used to assign the block and joint models and properties. The zones in the model are named 'elastic block' using the **block zone group** command. The −50° joint set is named 'strong joints', and the fault is named 'weak fault' using the **block contact group** command. The range phrase **angle=130 tol 2** (measured counterclockwise from the x-axis) for the **block contact group 'strong joints'** assigns the group name only to joints with angles between −51° and −49° (measured clockwise). Likewise, the range phrase **angle=109,111** assigns the **block contact group 'weak fault'** only to joints with angles between −71° and −69°.

The 'elastic block' zones are assigned the elastic material model and properties via the **block zone cmodel assign elastic** command. The 'strong joints' are assigned **block contact cmodel assign area**. The 'weak fault' is assigned **JOINT model residual**, which is the Coulomb slip model with residual strength. All other discontinuities are named 'glued joints' and are assigned the Coulomb slip model with high cohesive and tensile strengths, which results in "glued" or "fictitious" joints that correspond to the tunnel block that will be excavated. No slip or separation should occur on these joints.

Note that any new contacts that may be created during the simulation will be assigned the same joint model and properties as the 'weak fault' via the **block contact cmodel default** command.

Example 2.7 Assigning material models and properties

```
model new
block tolerance corner-round-length 0.01
block create polygon -10 -10 -10 10 10 10 10 -10
block cut tunnel 0,0 2,16
block cut joint-set angle -70 spac 40 origin -1,-1
block cut joint-set angle -50 spac 3  origin  0,2
block zone gen edge 2.0
;
; elastic blocks
block zone group 'elastic block'
block zone cmodel assign elastic density 2.5E3 bulk 1.5E9 shear 6E8 &
 range group 'elastic block'

;
; glued joints
group joint 'glued joints'
block contact cmodel assign area st-s 2E9 st-n 2E9 cohesion 1E10 ...
  tension 1E10 range group 'glued joints'
```

```
;
; strong joints
block contact group 'strong joints' range angle 130 tolerance 2.0
block contact cmodel assign area st-s 1E9 st-n 2E9 friction 45 ...
 range group 'strong joints'
;
; weak fault
group joint 'weak fault' range angle 110.0 tol 2.0
block contact cmodel assign residual st-s 1E9 st-n 2E9 ...
  friction 10 friction-residual 0 range group 'weak fault'
; new contact default
block contact cmodel default residual st-s=1E9 st-n=2E9 ...
  friction=10 friction-residual=0
;
; excavate tunnel
block zone group 'Null:tunnel' range ...
  atblock (1,0.845) (-1.343,-0.45) (0.112,0) (-1.306,-1.118)
block zone cmodel assign null range group 'Null:tunnel'
model save 'ex2_07.sav'
```

The tunnel block is changed to **block zone cmodel assign null** to simulate excavation. The null model is assigned via the group-name range named 'Null:tunnel'.

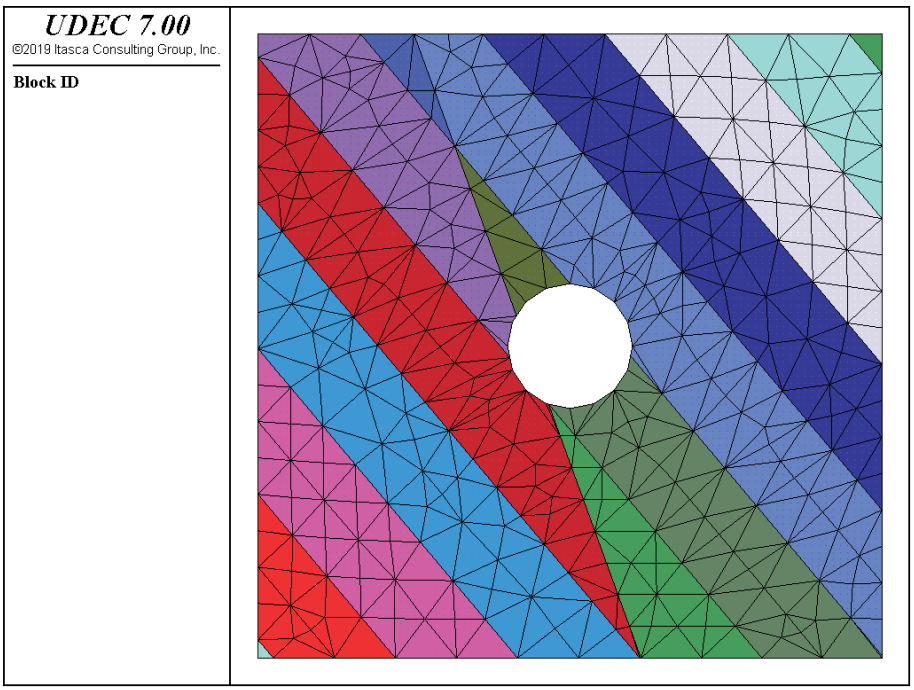

Figure 2.53 Zoning in a model of a circular tunnel with a 70° fault and a 50° joint set

2.6.3 Applying Boundary and Initial Conditions

Boundary and initial conditions must not be applied until after all block cutting is complete and the mesh for deformable blocks is generated. Mechanical boundary conditions are generally applied with the **block edge apply** command to specify stress boundary conditions and the **block gridpoint apply** command for force, and velocity (displacement) boundary conditions. Boundary forces and stresses can be applied to both rigid and deformable blocks, but boundary velocities can only be applied to deformable blocks. (See the commands **block fix**, **block free** and **block apply force** to apply boundary conditions to rigid blocks.) Table 2.3 provides a summary of the boundary condition commands and their effects. Refer to Help in *UDEC* for a complete listing of keywords for the **block edge apply** and **block gridpoint apply** commands.

Table 2.3 Boundary condition command summary

Command		Effect
block edge apply	**stress**	total stress applied to rigid or deformable blocks
	force-x	load applied in x-direction to rigid or deformable blocks
	force-y	load applied in y-direction to rigid or deformable blocks
block grid apply	**vel-x**	x-velocity applied to deformable blocks
	vel-y	y-velocity applied to deformable blocks
block fix		velocities fixed for rigid blocks
block free		velocities freed for rigid blocks
block apply	**force-x**	load applied in x-direction to rigid blocks
	force-y	load applied in y-direction to rigid blocks

The commands **block grid apply force-x** and **force-y** apply x- and y-components of force at boundary corners. The command **block edge apply stress** specifies components of the total stress tensor applied at the boundary. **block grid apply vel-x** and **vel-y** fix the x- and y-components of velocity at selected boundary gridpoints.

Note that by using the boundary commands, a condition or constraint that will not change (unless specifically changed by the user) is imposed.

Initial stress conditions can be specified for all zone stresses in deformable blocks and all normal and shear stresses along joints between rigid blocks and/or deformable blocks. The **block insitu** command is used to initialize stresses. By using this command, *initial* values are assigned to stresses; these *can change* while the computation proceeds.

Boundary and initial conditions can be applied to Example 2.7, for example, with the commands listed in Example 2.8:

Example 2.8 Applying boundary and initial conditions

```
block edge apply stress -10e6,0,0  range pos-x -11,-9
block edge apply stress -10e6,0,0  range pos-x   9,11
block edge apply stress 0,0,-5e6   range pos-y   9,11
block gridpoint apply vel-y =0.0   range pos-y -11,-9
block insitu stress -10e6,0,-5e6  stress-zz -4.8e6
```

A 10 MPa compressive stress is applied in the x-direction along the left- and right-hand sides of the model, and a 5 MPa compressive stress is applied in the y-direction along the top. (Remember that compressive stresses are negative.) The bottom boundary is fixed from movement in the y-direction. A zero-velocity boundary is particularly important when gravity is acting; this prevents the model from moving. Note that stress boundaries affect all degrees of freedom. Thus, stress boundary conditions should always be applied before velocity boundary conditions at the same boundary corners; otherwise, the prescribed velocity constraint will be lost. Also, note that x- and y-coordinate ranges are specified for each of the four commands. Be careful to ensure that the boundary affected by the command falls completely within the range. Type

```
block edge list
```

or by selecting a boundary summary plot from the plot ietem list.

to check boundary conditions.

The **block insitu** command initializes all stresses in the x-direction to -10 MPa, and in the y-direction to -5 MPa. The stress is also initialized in the out-of-plane direction, $\sigma_{zz} = -4.8$ MPa. For an elastic block analysis, the out-of-plane stress does not have to be initialized because σ_{zz} will not affect the plane-strain solution. However, for plasticity analyses, the out-of-plane stress may influence the failure state; thus, σ_{zz} should be chosen carefully.

2.6.4 Stepping to Initial Equilibrium

The *UDEC* model must be at an initial force-equilibrium state before alterations can be performed. The boundary conditions and initial conditions may be assigned such that the model is exactly at equilibrium initially. However, it is often necessary to calculate the initial equilibrium state under the given boundary and initial conditions, particularly for problems with complex geometries or multiple materials. This is done by using the **block cycle** (or **block step** or **block solve**) command. With the **block cycle** command, the user specifies a number of calculation steps to perform in order to bring the model to equilibrium. The model is in equilibrium when the net nodal force vector at each centroid of rigid blocks or gridpoint of deformable blocks is zero (see Section 1.2.5 in **Theory and Background**).

The maximum nodal force vector (called the maximum "out-of-balance" or "unbalanced" force) is monitored in *UDEC* and printed to the screen when the **block cycle** command is invoked. In this way, the user can assess when equilibrium has been reached.

For a numerical analysis, the out-of-balance force will never reach exactly zero. It is sufficient, though, to say that the model is in equilibrium when the maximum unbalanced force is small compared to the total applied forces in the problem. For example, if the maximum unbalanced force is initially 1 MN and drops to approximately 100 N, then the model can be considered at equilibrium, within 0.01% of the initial maximum unbalanced force.

This is an important aspect of numerical problem-solving with *UDEC*. *The user must decide when the model has reached equilibrium*. There are several features built into *UDEC* to assist with this decision. The history of the maximum unbalanced force is automatically recorded.

Additionally, the history of selected variables (e.g., velocity or displacement at a gridpoint) may be recorded. For example,

```
block grid hist  vel-x 5,5
block grid hist  disp-y 0,11
```

The first history records x-velocity at a gridpoint location closest to ($x = 5$, $y = 5$), while the second records y-displacement at a location closest to ($x = 0$, $y = 11$) in the model. After running several hundred (or thousand) calculation steps, a history of these records may be plotted to indicate the equilibrium condition.

By default, *UDEC* performs a static analysis by applying a mechanical damping algorithm known as local damping. This algorithm is described in Section 1.2.7 in **Theory and Background**. The data file in Example 2.9 illustrates the process to reach an initial equilibrium state.

Example 2.9 Stepping to initial equilibrium

```
model new
block tolerance corner-round-length 0.1
block tolerance minimum-edge-length 0.2
block create polygon -10,-10 -10,10 10,10 10,-10
block cut crack (-2,-2) (-2,2)
block cut crack (-2,2) (2,2)
block cut crack (2,2) (2,-2)
block cut crack (2,-2) (-2,-2)
block cut joint-set angle 70 spacing 40 origin -2,0
block cut joint-set angle 310 spacing 3 origin 1,2
block zone gen edge 2.0
block zone group 'block'
block zone cmodel assign elastic density 2.5E3 bulk 1.5E9 ...
  shear 6E8 range group 'block'
block contact group 'glued joints'
block contact group 'joint set' range angle 130 tol 2
block contact group 'fault' range angle 70 tol 2
block contact cmodel assign area stiffness-shear 2E9 ...
  stiffness-normal 2E9 cohesion 1E10 tension 1E10 &
  range group 'glued joints'
```

```
block contact cmodel assign area stiffness-shear 1E9 ...
  stiffness-normal 2E9 friction 45 range group 'joint set'
block contact cmodel assign area stiffness-shear 1E9 ...
  stiffness-normal 2E9 friction 5 range group 'fault'
; new contact default
block contact cmodel default area stiffness-shear=1E9 ...
  stiffness-normal=2E9 friction=5
block edge apply stress 0.0,0.0,-1.0E7 range pos-y   9.9,10.1
block grid apply velocity-x 0          range pos-x -10.1,-9.9
block grid apply velocity-x 0          range pos-x   9.9,10.1
block grid apply velocity-y 0          range pos-y -10.1,-9.9
block insitu stress -1.0E7,0.0,-1.0E7 stress-zz -1.0E7
block mechanical gravity=0.0 -9.81
block mech hist unbalance-maximum
block grid history displace-y 0.0,2.0
block cycle  700
model save 'fall1.sav'
```

The model shown in Figure 2.54 is for a square excavation in jointed rock. One joint set (dipping at −50°) and a single fault (dipping at 70°) are represented. The model is subjected to an in-situ hydrostatic stress field of 10 MPa, and gravity is acting in the negative y-direction. Gravity is invoked with the command

```
block mech grav  0.0  -9.81
```

in which the first value is the gravitational acceleration component in the x-direction and the second value is that in the y-direction – in this case, 9.81 m/sec^2 (acting downward).

Gravity can be omitted from a model if the stress variation due to gravity is small across the model compared to the in-situ stresses. In Example 2.9, the stress variation due to gravity is less than 0.5 MPa, and could be neglected relative to the 10 MPa in-situ stresses. Gravity is applied, though, to help identify loose blocks around the opening. If the block in the roof of the excavation becomes detached, it will fall into the opening due to gravity. This is demonstrated later in Section 2.6.5. Note that if stresses due to gravity are the same magnitude as the in-situ stresses, then a stress gradient should be applied with the **block insitu** command to speed convergence to initial equilibrium.

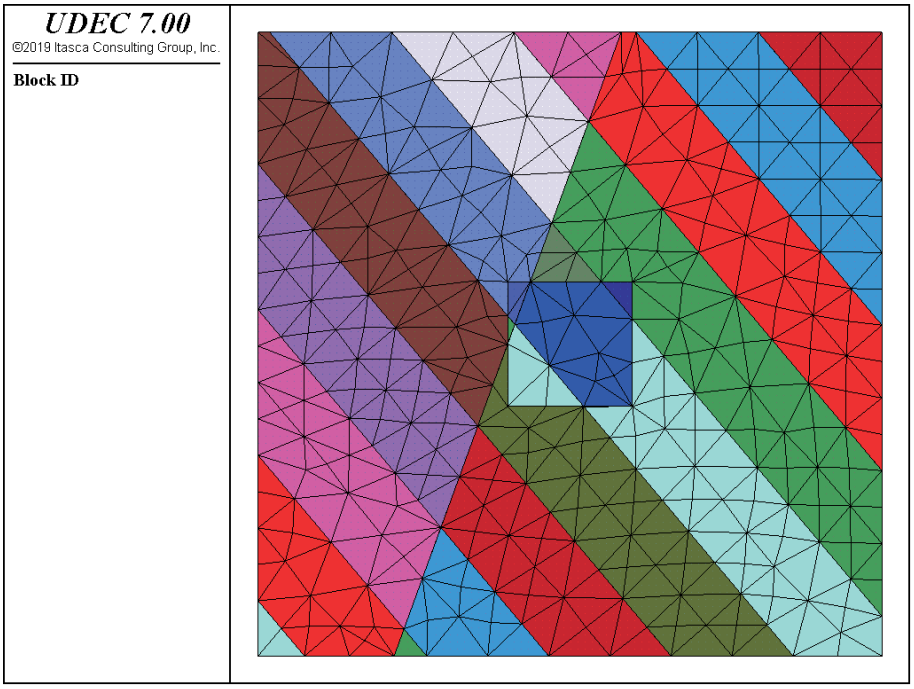

Figure 2.54 Square excavation in jointed rock

The initial unbalanced force is approximately 2.0 MN. After 700 steps, this force has dropped to around 10 N. By plotting the two histories, it can be seen that the maximum unbalanced force has approached zero, while the displacement has approached a constant magnitude of approximately 2.7×10^{-3} m. Type

Figures 2.55 and 2.56 show the unbalanced force and displacement history plots, respectively.

The force imbalance in the model is caused by the slight difference in the location of contact forces and gridpoint forces at block corners (see Section 1.2.4 in **Theory and Background**). This is related to the corner-rounding; the rounding creates very small "holes" at block corners. Some stepping is usually necessary to equilibrate this force imbalance.

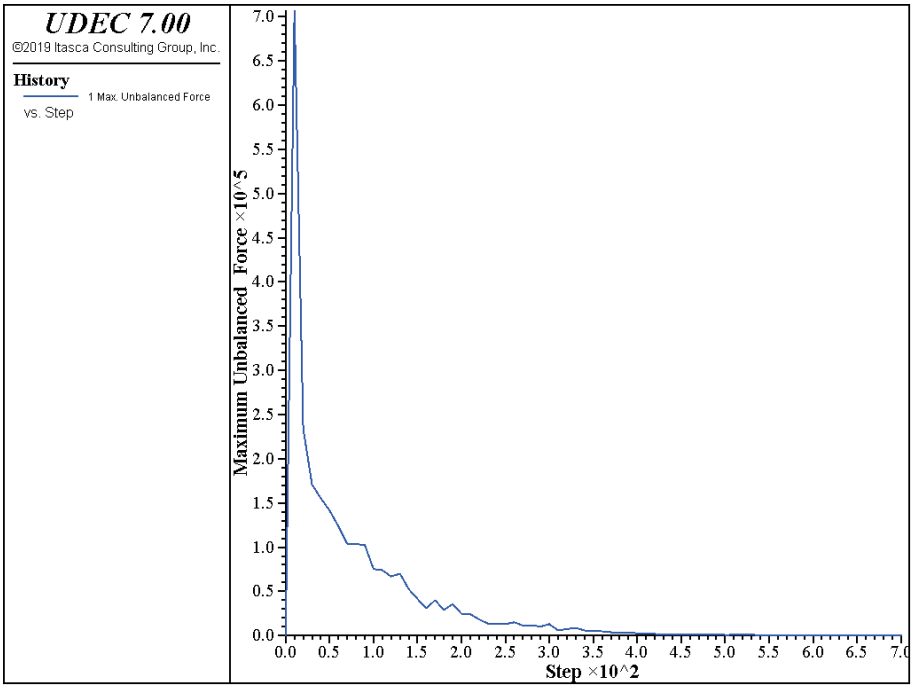

Figure 2.55 Maximum unbalanced force history

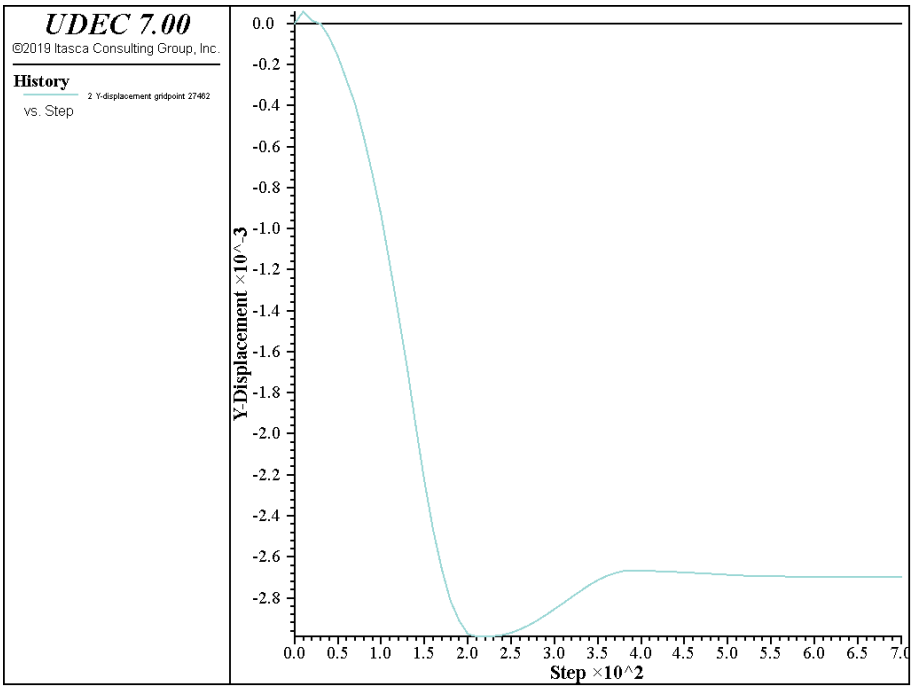

Figure 2.56 y-displacement history at (0, 2)

The **block solve** command can be used in place of **STEP** if the user wants *UDEC* to stop automatically when the maximum unbalanced force falls below a specified limit. Replace **STEP 700** with **block solve force = 100** and repeat the earlier problem. This time, *UDEC* should stop the calculation at step 740. If the plots are made again, essentially the same results as given in Figures 2.55 and 2.56 will be seen.

The force limit is set by typing

```
block solve force  =  f
```

where ***f*** is a user-specified value for the force limit.

A force ratio limit can also be specified with the **block solve** command; this limit is the default if the **block solve** command is given without any keywords. The force ratio is the ratio of the average unbalanced force magnitude to the average applied force magnitude for all rigid block centroids or deformable block gridpoints in the model. The default force ratio limit is 10^{-5}.

There is also a runtime limit, in minutes (**clock=1440**, by default), and step limit (**step=100000**, by default) for the **block solve** command.

It is very important to check the failure condition of the discontinuities at this stage. For this problem, the initial x- and y-stress components are equal; thus the joints in the model cannot slip. This can be verified by slecting the state attribute in a joint plot and adding a boundary summary plot item.

A plot will be made of the outer boundary of the model and any joints for which the Coulomb slip criterion is reached. It is possible that, for selected model conditions, joint slip will occur under the in-situ stress state. For example, change the horizontal stress component σ_{xx} to -5 MPa and rerun Example 2.9. The fault will slip along its entire length. If joint slip is indicated following the initial equilibrium calculation, the user should reevaluate the chosen parameters for in-situ stress state and for strength of the discontinuity. A discontinuity that is at slip along its entire length across the model suggests that the problem is not well-posed.

In an analysis, it is very important that the model be at equilibrium before alterations are made. Several histories should be recorded throughout a model to ensure that a large force imbalance does not exist. It does not affect the analysis adversely if more steps than are needed to reach equilibrium are taken. However, it will affect the analysis if an insufficient number of steps are taken.

A *UDEC* calculation can be interrupted at any time during stepping. In the gui this can be done by pressing the red busy circle in the toolbar. In the GUI, this may also be done by pressing the <SHIFT-ESC> key. In the *GIIC*, this can be done by pressing the STOP button in the *Model cycling* dialog.

2.6.5 *Performing Alterations*

UDEC allows model conditions to be changed at any point in the solution process. These changes may be of several forms:

- excavation of material;
- addition or deletion of boundary loads or stresses;
- fix or free velocities of boundary corners; or
- change of material model or properties for blocks or discontinuities.

Excavation is performed with either the **block delet** or **block zone cmodel assign null** command. Loads and stresses are applied with the **block edge apply force-x**, **force-y** or **stress** command. Boundary corners are fixed via the **block grid apply vel-x** or **vel-y** command. The constraint at boundaries is removed with **block grid free-x** and **free-y**. Material models and properties for deformable blocks are changed with the **block zone cmodel assign** command, and models and properties for discontinuities are changed with the **block contact cmodel assign** command.

It should be evident that several commands can be repeated to perform various model alterations. For example, continue Example 2.9 from the initial equilibrium stage using the commands in Example 2.10.

Example 2.10 Excavate a tunnel and monitor the response

```
model rest 'fall1.sav'
block delete range pos-x -2,2 pos-y -2,2
block gridpoint init displacement-x 0
block gridpoint init displacement-y 0
hist reset
block grid hist dis-y 0,2
block step 2000
model save 'fall2.sav'
```

The **block delete** command produces an alteration in stress distribution due to excavation of the square opening. The gridpoint displacements and history records are reset, so that only the change in displacement due to the excavation is monitored. It is recommended that history locations be assigned in a model *after* blocks have been deleted. This ensures that a history is not accidentally assigned to a location in a deleted block, in which case, no record would be made.

After the square opening is excavated, a high unbalanced force results, and the model is stepped to attempt to reach equilibrium again.* However, in this case, the unbalanced force is not observed to

* Do not use the **block solve** command if you anticipate that the alteration will result in failure (i.e., a force equilibrium condition cannot be achieved).

approach a very small number, but rather remains at a constant value of 0.017 MN. Furthermore, the y-displacement history shows that location (0,2) is still moving downward after 2000 additional steps. The block in the roof of the excavation has become detached from the surrounding blocks and is falling into the excavation. This is clearly seen from the plot in Figure 2.57. The unbalanced force cannot approach zero because the roof block is falling freely.

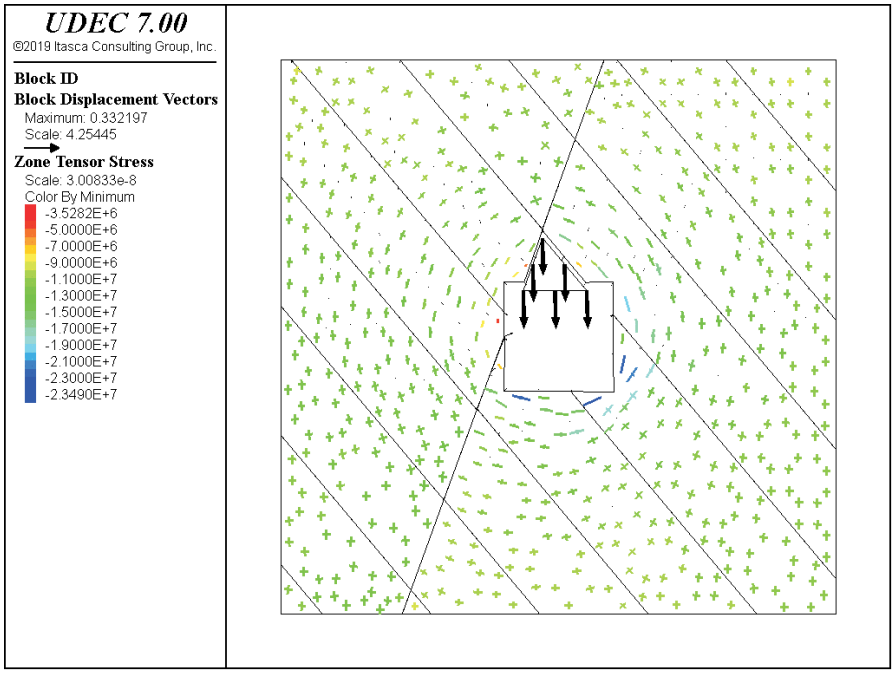

Figure 2.57 Roof block falls into opening

2.6.6 Saving/Restoring Problem State

Two other commands, **model save** and **model restore**, are helpful when performing analyses in stages. At the end of one stage (e.g., initial equilibrium), the model state can be saved by typing

```
model save 'file.sav'
```

where 'file.sav' is a user-specified file name. The extension ".SAV" identifies this file as a saved file (see Section 2.10). This file can be restored at a later time by typing

```
model rest 'file.sav'
```

and the model state at the point at which the model was saved will be restored. It is not necessary to build the model from the beginning every time a change is made; merely save the model before the change and restore it whenever a new change is to be analyzed. For example, in the previous example, the state should be saved after the initial equilibrium stage. Then, different methods can be evaluated to stabilize the falling block.

For example, we inserted

```
model save 'fall1.sav'
```

after the **block step 700** command at the end of the data file for Example 2.9. Now we try stabilizing the block with cable reinforcement using the commands in Example 2.11:

Example 2.11 Stabilize roof block with a cable bolt

```
model rest 'fall1.sav'
block delete range pos-x -2 2 pos-y -2 2
block gridpoint init displacement-x 0
block gridpoint init displacement-y 0
hist reset
block gridpoint history displacement-y 0 2
block structure cable create begin 0.0 2.0 end 1.0 4.0 segments 5 ...
  material-steel 4 material-grout 4
block struct cable property material 4 cross-sectional-area 5E-4 ...
  density 5E3 rupture-tension-strain 1E30 yield-tension 2.4E5 ...
  young 2E11 grout-stiffness 5E10 grout-strength 3E5 spacing 1
block cycle 2000
model save 'stable.sav'
```

This file can be created with the built in text editor, or within the *GIIC*, and called into *UDEC*. After the run is completed, the saved file can be restored and evaluated to study the effect of cable reinforcement. See Help in *UDEC* and Section 1 in **Special Features** for descriptions of the **block struct cable create** command and cable parameters assigned with the **block struct vcable property** command. The plot og the cables after **block step 2000** is shown in Figure 2.58. The cable location and axial force along its length are shown on the figure. A history of y-displacement shows that the block has stopped moving after 9.3×10^{-2} m of vertical displacement.

When using the *GIIC*, saving and restoring problem states is done automatically, and the *Project Tree Record* format allows the user to switch among the different saved states by point-and-click operations.

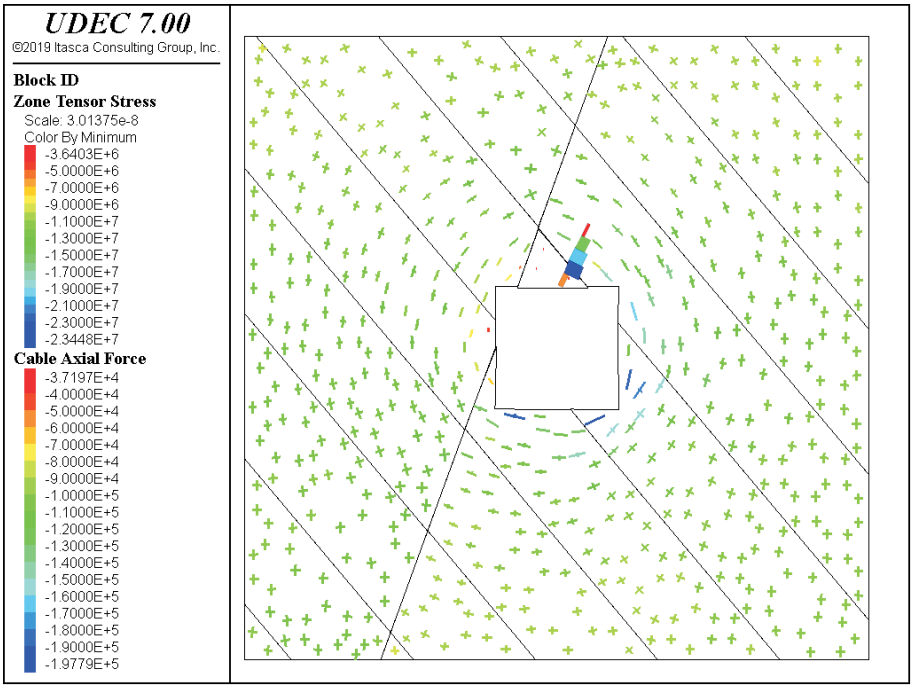

Figure 2.58 Roof block stable with one cable

2.6.7 Summary of Commands for Simple Analyses

The major command words described in this section are summarized in Table 2.4. These are all that are needed to begin performing simple analyses with *UDEC*. Start by running simple tests with these commands (e.g., direct shear tests on single joints or simple excavation stability analyses). It may be helpful to review the detailed description of these commands in The Help - *UDEC*.

Then try adding more complexity to the model. Before running very detailed simulations, though, we recommend that you read Section 3, which provides guidance on problem solving in general.

Table 2.4 Basic commands for simple analyses

Function	Command
Block Model Creation	**block tol corner-round-length** **block create**
Block Cutting	**blck cut crack** **block cut joint-set** **block cut tunnel** **block cut arc**
Material Model & Properties for Blocks and Joints	**block zone generate** **block change** **block contact change** **block contact cmodel assign** **block contact property** **block zone cmodel assign** **block zone property**
Boundary/Initial Conditions	**block edge apply** **block gridpoint apply** **block insitu**
Initial Equilibrium (with gravity)	**block mech damp local** **block mech gravity** **block cycle** **block solve**
Perform Alterations	**blck delete** **block change** **block contact propety** **block edge apply** **block gridpoint apply** **block struct cable** **block zone property**
Monitor Model Response	**block gridpoint history**
Save/Restore Problem State	**model save** **model restore**

2.7 Sign Conventions

The following sign conventions are used in *UDEC* and must be kept in mind when entering input or evaluating results.

BLOCK MOTION – Positive motion is upward and to the right.

DIRECT STRESS – Positive stresses indicate tension; negative stresses indicate compression.

SHEAR STRESS – With reference to Figure 2.59, a positive shear stress points in the positive direction of the coordinate axis of the second subscript if it acts on a surface with an outward normal in the positive direction. Conversely, if the outward normal of the surface is in the negative direction, then the positive shear stress points in the negative direction of the coordinate axis of the second subscript. The shear stresses shown in Figure 2.59 are all positive.

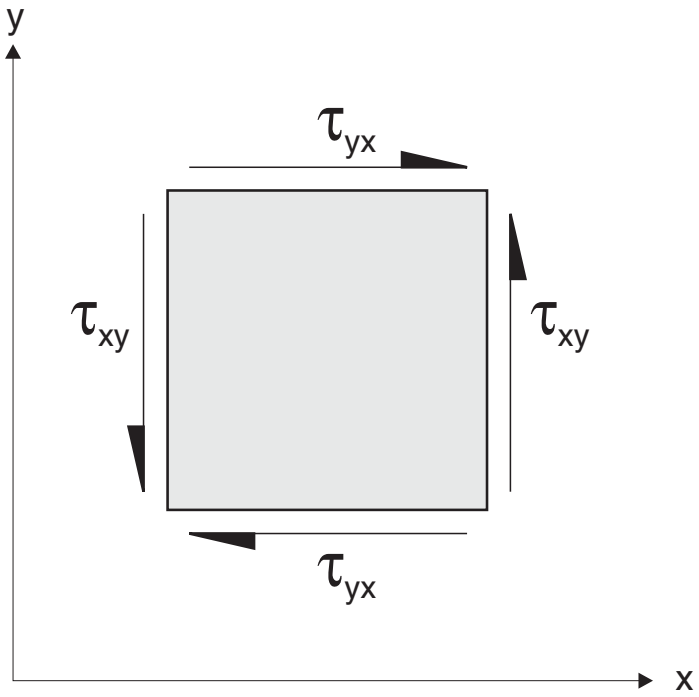

Figure 2.59 Sign convention for positive shear stress components

DIRECT STRAIN – Positive strain indicates extension; negative strain indicates compression.

SHEAR STRAIN – Shear strain follows the convention of shear stress (see above). The distortion associated with positive and negative shear strain is illustrated in Figure 2.60:

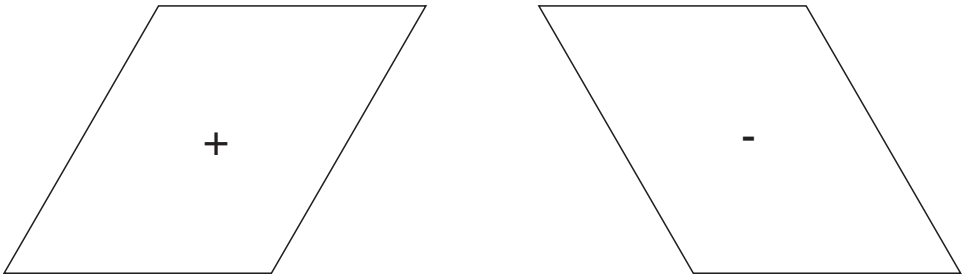

Figure 2.60 Distortion associated with positive and negative shear strain

JOINT NORMAL STRESS – Joint normal stress is positive in compression.

JOINT NORMAL OPENING – Joint opening is positive; joint closure is negative.

JOINT SHEAR STRESS – Joint shear stress is positive for the following direction of relative movement:

JOINT SHEAR DISPLACEMENT – Joint shear displacement is positive for the following direction of relative movement:

FLUID PRESSURE – A positive fluid pressure in a joint causes the domain to expand. Counter-clockwise rotations and moments are positive. The sign for a force or velocity boundary condition is determined by the direction of the force or velocity vector (i.e., + when pointing in the positive axis direction). See Section 1 in **Special Features** for the sign convention used for structural elements.

STRUCTURAL ELEMENTS – Axial forces in structural elements are positive in compression. Shear forces in structural elements follow the opposite sign convention as that given for zone shear stress (illustrated in Figure 2.59). Moments at the end of beam and rockbolt elements are positive in the counterclockwise direction.

Translational displacements at nodes are positive in the direction of the positive coordinate axes, and angular displacements are positive in the counterclockwise direction.

The shear force and shear displacement at a cable/grout interface-spring node, or a rockbolt shear coupling-spring node, are positive if the node displacement is in the direction of the specification of the cable or rockbolt.

2.8 Systems of Units

UDEC accepts any consistent set of engineering units. Examples of consistent sets of units for basic parameters are shown in Table 2.5. The user should be very careful when converting from one system of units to another. An excellent reference on the subject of units and conversion between the Imperial and SI systems can be found in the *Journal of Petroleum Technology* (December 1977). No conversions are performed in *UDEC* except for friction and dilation angles, which are entered in degrees.

Table 2.5 Systems of units – mechanical parameters

	SI				Imperial	
Length	m	m	m	cm	ft	in
Density	kg/m^3	10^3 kg/m^3	10^6 kg/m^3	10^6 g/cm^3	$slugs/ft^3$	$snails/in^3$
Force	N	kN	MN	Mdynes	lb_f	lb_f
Stress	Pa	kPa	MPa	bar	lb_f/ft^2	psi
Gravity	m/sec^2	m/sec^2	m/sec^2	cm/s^2	ft/sec^2	in/sec^2

where 1 bar = 10^6 dynes/cm^2 = 10^5 N/m^2 = 10^5 Pa;
1 atm = 1.013 bars = 14.7 psi = 2116 lb_f/ft^2 = 1.01325×10^5 Pa;
1 slug = 1 $lb_f - s^2/ft$ = 14.59 kg;
1 snail = 1 $lb_f - s^2/$ in; and
1 gravity = 9.81 m/s^2 = 981 cm/s^2 = 32.17 ft/s^2.

Systems of units for parameters associated with structural elements, fluid flow and heat transfer are given in Sections 1, 2 and 3 in **Special Features**, respectively.

2.9 Resolution Limits

When selecting a system of units, be careful to avoid calculations that approach the resolution limits of the computer hardware. For 64 bit computers, the number of significant digits is 16. If numbers exceed these limits, numerical roundoff will occur and the model results may be affected.

2.10 Files

There are ten types of files that are either used or created by *UDEC*. The files are distinguished by their extensions and are described below.

GIIC PROJECT FILES

This file is created when the user starts a new project in the *GIIC*. The file is an ASCII file containing variables that describe the state of the model and the *GIIC* at the stage when the project is saved, and includes a link to the individual *UDEC* save files (".SAV") associated with the project. The file contains all data and commands associated with the project, and is updated automatically every time a new model state is saved. The project file can also be updated at the user's request when the FILE/SAVE PROJECT menu item is pressed in the *GIIC*. The file name has the extension ".PRJ," which should not be changed.

GUI PROJECT FILES

This file is created when the user starts a new project in the GUI. The file is a binary file containing variables that describe the model files, plots, and the GUI layout at the stage when the project is saved. It includes a link to the individual *UDEC* save files (".SAV") associated with the project. The file contains all data and commands associated with the project. The project file can be updated at the user's request when the FILE/SAVE PROJECT menu item is pressed. The file name has the extension ".UDPRJ," which should not be changed.

SAVE FILES

"UDEC.SAV" – This file is created by *UDEC* at the user's request when issuing the command **model save**, either from the SAVE button in the *GIIC*, or by typing in the command at the command line. The default file name is "UDEC.SAV," which will appear in the default folder when quitting *UDEC*. The user may specify a different file name by issuing the command **model save** filename, where filename is a user-specified file name. "UDEC.SAV" is a binary file containing the values of all state variables and user-defined conditions. The primary reason for creating save files is to allow one to investigate the effect of parameter variations without having to rerun a problem completely. A save file can be restored and the analysis continued at a subsequent time (see the **model restore** command in Help in *UDEC* .)

If the save file is created in the *GIIC*, the file will also include information that describes the state of the *GIIC* at the stage the file is saved. Normally, it is good practice to create several save files during a *UDEC* run.*

DATA FILES

---

* Save files created from a factor-of-safety calculation (**block factor-of-safety**) are given a different extension, ".FSV," to distinguish them from standard save files.

In command-driven mode, the user has a choice of running *UDEC* interactively (i.e., entering *UDEC* commands while in the *UDEC* environment) or via a data file. The data file is a formatted ASCII file created by the user – it contains the set of *UDEC* commands that represents the problem being analyzed. To use data files with *UDEC* in command-driven mode, see the **call** command in Help in *UDEC* .

Data files can have any file name and any extension. It is recommended that a common extension (e.g., ".DAT" for *UDEC* input commands, and ".FIS" for *FISH* function statements) be used to distinguish these files from other types of files.

INITIALIZATION FILE

"UDEC.INI" – This is a formatted ASCII file, created by the user, that *UDEC* will automatically access upon start-up or when a **model new** command is issued. *UDEC* searches for the file "UDEC.INI" in the folder in which the code is executed and, if not found, in the folder pointed to by the ITASCA environment variable. The file may contain any valid *UDEC* command(s) (see Help in *UDEC* .)

Although this file does not need to exist (i.e., no errors will result if it is absent), it is normally used to change default options in *UDEC* to those preferred by the individual user each time a new analysis is run. Note that the "UDEC.INI" is only operational when running in command-driven mode. The file is not used when running in the *GIIC*.

GIIC MATERIALS FILES

This file is created by *UDEC* at the user's request as a library of commonly used material properties. The file is created from the *Materials List* dialog in the MATERIALS pane of the *GIIC*. This file is automatically given the extension ".GPF," and is an ASCII file containing the values of material properties that the user wishes to save for application in different projects. The file can be updated and modified from the *Materials List* dialog. A default materials file that is automatically loaded in the MATERIALS pane is provided.

LOG FILES

"UDEC.LOG" – This file is created by *UDEC* at the user's request when issuing the command **log on**. It is a formatted ASCII file. The default name of the file is "UDEC.LOG," which will appear in the default folder after quitting *UDEC*. The user may specify a different file name by issuing the command **log-file** filename, where filename is a user-supplied file name. The command may be issued interactively or be part of a data file. Subsequent to the **log on** command, all text appearing on the screen will be copied to the log file. The log file is useful in providing a record of the *UDEC* work session; it also provides a document for quality-assurance purposes. The "UDEC.LOG" is not operational in the *GIIC* because the log file is immediately available from the *Console* pane.

HISTORY FILES

"UDEC.HIS" – This file is created by *UDEC* at the user's request when issuing the command **history export *n***, where ***n*** is a history number (see the **history** command in Help in *UDEC* .)

It is a formatted ASCII file. The default name of the file is "UDEC.HIS," which will appear in the default folder after quitting *UDEC*. The user may specify a different file name by adding a user-supplied filename at the end of the **history** command. The user-supplied filename takes the place of "UDEC.HIS." A record of the history values is written to the file, which can be examined using any text editor.If the keyword **csv** is added, the file will be formatted for direct input into a speadsheet or data base.

TABLE FILES

"UDEC.TAB" – This file is created by *UDEC* at the user's request when issuing the command **table *n* export *sfile***, where ***n*** is the table number and ***sfile*** specifies the name of the output file (see the **table** command in the Help in *UDEC* .) It is a formatted ASCII file. If the keyword **csv** is added, the file will be formatted for direct input into a speadsheet or data base.

PLOT FILES

Plot files are created at the user's request by right click on the plot and select export bitmap. The output type can be changed in the menu.

PNG movies (sequential images) can also be created by either setting this output mode in the tools->options->movies.

VIRTUAL MODEL FILES

This file contains the commands to create a virtual-model object. The virtual-model object is given the extension ".GRD," and is created with the virtual model editor tools.

2.11 References

Journal of Petroleum Technology. "The SI Metric System of Units and SPE's Tentative Metric Standard," 1575-1616 (December 1977).