POLY3D: A THREE-DIMENSIONAL, POLYGONAL ELEMENT, DISPLACEMENT DISCONTINUITY BOUNDARY ELEMENT COMPUTER PROGRAM WITH APPLICATIONS TO FRACTURES, FAULTS, AND CAVITIES IN THE EARTH'S CRUST

A THESIS
SUBMITTED TO THE DEPARTMENT OF GEOLOGY
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Andrew Lyle Thomas
June 1993

Approved for the Department

_____

David D. Pollard (Advisor)

Approved for the Department

_____

Atilla Aydin

Approved for the University Committee on Graduate Studies:

_____

Dean of Graduate Studies

PREFACE

Poly3D is a C language computer program that calculates the displacements, strains and stresses induced in an elastic whole- or half-space by planar, polygonal-shaped elements of displacement discontinuity. Dislocations of varying shapes may be combined to yield complex three-dimensional surfaces well-suited for modeling fractures, faults, and cavities in the earth's crust. The algebraic expressions for the elastic fields around a polygonal element are derived by superposing the solution for an angular dislocation in an elastic half-space.

The purposes of this research are (1) to verify the workability and accuracy of the polygonal boundary element method and (2) to provide a user-friendly program that other researchers can apply to problems in crustal deformation. In its present state, Poly3D produces correct and accurate results for some problems but not for others. We believe that the remaining difficulties result from programming error(s) rather than a fundamental (or numerical) limitation of the method. The Rock Fracture Project at Stanford University will continue to develop, maintain and test the code.

Because Poly3D is ultimately intended for distribution to other researchers, this thesis is presented in the form of a user's manual with example problems. The accuracy of the output produced by Poly3D is verified by checking it against analytical solutions or other numerical solutions. Appendix A details the algebraic implementation of the polygonal element method.

ACKNOWLEDGMENTS

I could not have written Poly3D without the help of others. John Rudnicki captured my interest in the geological application of polygonal dislocations during a talk at Stanford University. Ramon Arrowsmith gave me a head start by sharing the results of his literature search on the subject. M. Jeyakumaran pointed out several typographical errors in a key reference and provided a FORTRAN code which served as an early template for Poly3D. Atilla Aydin, Paul Segall, and Mark Zoback suggested a number of potential applications, imparting focus and direction to my programming efforts.

Dave Pollard has been an earnest and patient mentor since the first day I walked into his office as an undergraduate. His open door and open mind have enriched my days at Stanford tremendously. I am also indebted to Ken Cruikshank, whose technical insight, programming advice, encouragement and coffee got me through many obstacles.

Thank you to my parents, for their love and support and to Carol and Paul Evans for their many hours of baby-sitting.

This thesis is dedicated to Julia and Evan. I could never have done it without you.

CONTENTS

1. DISTRIBUTION NOTES

1.1 Distribution

Poly3D is a C language computer program written by Andrew L. Thomas as part of an M.S. thesis in the Department of Geology at Stanford University. The program will be updated and maintained by the Rock Fracture Project at Stanford University. Copies of this manual and the source code for Poly3D can be obtained from:

Rock Fracture Project, ATTN: Poly3D
Dept. of Environmental & Geological Sciences
Stanford University
Stanford, CA 94305 USA

A minimal fee will be changed to cover postage, photocopying, and other costs. Correspondence by electronic mail should be sent to Prof. David D. Pollard at the following Internet address:

dpollard@pangea.stanford.edu

1.2 Redistribution

Because Poly3D is available at negligible cost, please do not redistribute the source code. Requests for additional copies should be directed to the Rock Fracture Project. This will minimize your risk of inheriting a copy with undocumented changes and will enable us to maintain a mailing list to notify users of updates and bug fixes.

1.3 Literature Citation

If you use Poly3D to produce results for a report or publication, please cite this thesis as follows:

Thomas, A.L., 1993, Poly3D: A Three-Dimensional, Polygonal Element, Displacement Discontinuity Boundary Element Computer Program with Applications to Fractures, Faults, and Cavities in the Earth's Crust: M.S. Thesis, Stanford University, Stanford, CA, **p.

Also, please send one reprint of each report or publication to Prof. David D. Pollard at the address given in §1.1.

2. LEGAL MATTERS

2.1 Numerical Recipes

Poly3D uses several functions adapted from the book *Numerical Recipes in C: The Art of Scientific Computing* (Press et al., 1992). They appear in the source files `nr.c`, `nr.h`, `nrutil.c`, and `nrutil.h`. While permission has been granted for their use in Poly3D, you are not authorized to copy the functions for use in any other program. *Numerical Recipes* has been published in several editions covering the most commonly used programming languages. We have found it to be an excellent reference for anyone engaged in scientific computing.

2.2 Stanford University Disclaimer

Stanford makes no representations or warranties, express or implied. By way of example, but not limitation, Stanford makes no representations or warranties of merchantability or fitness for any particular purpose, or that the use of the license software components or documentation will not infringe any patents, copyrights, trademarks, or other rights. Stanford shall not be held liable for any liability nor for any direct, indirect, or consequential damages with respect to any claim by licensee or any third party on account of or arising from use of this program.

3. BETA VERSION 0.0 RELEASE NOTES

3.1 New Features

The documentation for future versions of Poly3D will provide information in this space on new features added since the first release. If you have any suggestions for changes, please direct them to the Rock Fracture Project at the address listed in §1.1.

3.2 Bug Fixes

The challenges of implementing a three-dimensional, polygonal boundary element code are many. The present version of Poly3D contains a number of known, unsolved bugs. Additional errors will doubtless be discovered as other researchers adapt the program to their own purposes. The Rock Fracture Project will attempt to fix all program errors that come to its attention. The source code listed in Appendix A therefore represents a snapshot of a still-evolving program. For the latest version of Poly3D, contact the Rock Fracture Project at the address listed in §1.1.

Known Bugs in Poly3D

• Poly3D cannot calculate the displacement, strain, or stress at any point directly below an element vertex (§A6). For observation points directly below a vertex, these elastic field quantities are assigned a null value given by the Poly3D constant `null_value` (§11.1). Because tractions and displacements are evaluated at element centers, no element can lie with its center directly below a vertex.

• We have not yet been able to successfully solve problems in which the polygonal elements are assembled to form a closed surface. Problems which combine elements of different shapes and sizes are often unstable. We believe that these difficulties result from programming error(s) rather than a fundamental (or numerical) limitation of the method.

Box 1

4. NOTATION

4.1 Tensor and Vector Notation

In this manual, vector quantities are indicated by a character with a single underscore (e.g. $\underline{v}$). Unit vectors are indicated by a character topped by the carrot symbol (e.g. $\hat{v}$). A letter followed by a single subscript, for example v_i, denotes the i^{th} component of the vector $\underline{v}$), where i is understood to range from one to three.

Tensor quantities are indicated by a double underscore (e.g. $\underline{\underline{\sigma}}$). In this manual, we follow the standard *on-in* notation for tensor subscripts. For example, the stress component *on* a surface with outward normal parallel to $\hat{x}_i$ acting *in* the direction $\hat{x}_j$ is given by σ_{ij} (§11.1.5). Subscripts i and j are understood to range from one to three. See Fung (Fung, 1969) for a review of vector and tensor notation.

4.2 Summation Convention

Equations in this manual employ the summation convention for repeated indices within a term, where indices are understood to range from one to three. For example, the equation $t_i = n_j \sigma_{ji}$ expands to $t_i = n_1 \sigma_{1i} + n_2 \sigma_{2i} + n_3 \sigma_{3i}$.

5. INTRODUCTION

5.1 What is Poly3D?

Poly3D is a C language computer program that calculates the displacements, strains and stresses induced in an elastic whole- or half-space by planar, polygonal-shaped elements of displacement discontinuity (Fig. 1). Polygonal elements may have any number of sides, with a minimum of three. The algebraic expressions for the elastic fields around a polygonal element are derived by superposing the solution for an angular dislocation in an elastic half-space (Yoffe, 1960; Comninou and Dunders, 1975) in the manner described by Brown (1975) and Jeyakumaran et al. (1992). Appendix A presents a more complete discussion regarding the implementation of polygonal dislocations.

Geologically, a polygonal element may represent some portion, or all, of a fracture or fault surface across which the discontinuity in displacement is assumed constant. Several (or many) elements may be used to model a fracture or fault with a non-uniform opening and/or slip distribution. Elements also may be joined together to form a closed surface. Depending on the problem being considered, the enclosed volume may represent either a finite elastic body or a void in an otherwise infinite or semi-infinite elastic body.

The polygonal elements in Poly3D are well-suited for modeling complex surfaces with curving boundaries. Fault surfaces which change in both strike and dip can be meshed without creating gaps (Fig. 2a). Polygonal elements easily replicate the irregular periphery of a hydraulic fracture (Fig. 2b). A spherical void can be modeled by assembling hexagonal and pentagonal elements in the manner of a soccer ball (Fig. 2c). These kinds of problems would be difficult or impossible to attempt using rectangular elements alone. For this reason, Poly3D represents a significant improvement over existing rectangular element codes such as Dis3D (Erickson, 1986)

5.2 Displacements and Tractions Across an Element

The displacement field induced by a polygonal element is continuous except along the element itself. Thus, no element can produce a displacement discontinuity across another. On the other hand, the traction on an element surface is determined by resolving any remote stress plus the *total* stress field induced by *all* elements onto the element plane (see Appendix A). Another consequence of the mathematics of dislocations is that while the displacement field is discontinuous when passing through the element from one side to the other, certain components of the stress tensor are continuous. However, the assumption of a uniform displacement discontinuity causes a non-uniform distribution of tractions across the element surface. For example, the displacement discontinuity does not taper to zero at the edge of an element, so strains and stresses there are unbounded (Fig 1). For these reasons, Poly3D provides separate routines for calculating elastic fields in the body and for calculating displacement discontinuities and tractions on an element.

5.3 Boundary Conditions

The exact boundary conditions on a polygonal element, α, are given by the components of the uniform displacement discontinuity as measured by the Burgers vector $(b_1^\alpha, b_2^\alpha, b_3^\alpha)$ (§7.2). Approximate boundary conditions may be given by the components of traction $(t_1^\alpha, t_2^\alpha, t_3^\alpha)$ acting at the center of the element surface. Mixed boundary conditions (e.g. $b_1^\alpha, b_2^\alpha, t_3^\alpha$) are also allowed. Because the tractions acting on an element surface to produce a uniform displacement discontinuity are non-uniform, particularly near its edges, stress components are evaluated at the element center (centroid) and used to define the approximate traction boundary condition. Experience with dislocation modeling in two- and three-dimensions (Crouch and Starfield, 1983; Jeyakumaran et al., 1992) has demonstrated that the errors introduced by evaluating tractions in this manner typically are small, especially when many elements are used to model a fracture, fault, or void.

In order to calculate displacements, strains, and stresses in the elastic body, we must know the complete Burgers vector, $\underline{b}$, for each polygonal element (see Appendix A).

When only displacement discontinuity boundary conditions are given for each element, we can calculate the these elastic fields directly by superposition. However, whenever a traction boundary condition t_i^{α} is specified, we must first determine the corresponding (unknown) Burgers vector component, b_i^{α}. Each traction boundary condition leads to one equation -- giving the traction, t_i^{α}, at the center of element α induced by the remote stress plus the displacement discontinuities (b_1,b_2,b_3) on *all* elements -- and one unknown, b_i^{α}. If the total number of traction boundary conditions specified on all polygonal elements is Q, we must solve a system of Q linear equations for the Q unknown Burgers vector components. Displacement, strains, and stresses at any point in the elastic body are then calculated by superposition.

6. COMPILING POLY3D

6.1 ANSI Compatibility

The source code for Poly3D conforms to the ANSI standard for the C programming language (1989). The most important change between ANSI C and the older "K&R" definition of the language is a new syntax for declaring and defining functions (Kernighan and Ritchie, 1988). In order to maintain compatibility with older K&R compilers, the source code for Poly3D contains redundant function declarations and definitions that support either syntax.

Compiling for ANSI Compatibility

The ANSI syntax for declaring and defining functions is used when one of the symbolic names, `__STDC__` or `ANSI`, is defined at compile time. The K&R syntax is used otherwise. The documentation for your ANSI C compiler should contain information regarding how and when these symbolic names can be defined.

Box 2

6.2 Source Files and Makefiles

The source code for Poly3D is divided among several files which must be compiled and linked to create the executable program. On a UNIX system, you might compile Poly3D by typing

```
cc <sourcefiles> -o poly3d -lm
```

where `cc` is the name of your C compiler, `<sourcefiles>` is a space-separated list of the source files listed in Appendix B, `-o poly3d` names the executable file to be created, and `-lm` is an option for linking with the math object library.

On many computers, a makefile may be used to simplify the process of compiling and maintaining Poly3D. For your reference, a sample makefile for UNIX systems is included in Appendix C. A makefile appropriate for your computer and compiler may also be available in digital form on your Poly3D distribution disk/tape. Check the file, README, for more information.

6.3 The FPROMPT Option

Poly3D normally assumes that the names of input and output files are supplied on the command line (§8.1). If command line arguments are difficult or impossible to supply on your computer, you may want to compile Poly3D using the FPROMT option. Poly3D will then use the standard input (normally the keyboard) and standard output (normally the monitor) to prompt you for input and output file names. If you don't know what the standard input and output are for your machine, consult the documentation for your C compiler.

Enabling the FPROMPT Option

The FPROMPT option is enabled by defining the symbolic name, FPROMPT, at compile time. The documentation for your C compiler should contain instructions for defining symbolic names.

Box 3

A less elegant way to enable the FPROMPT option is to add the following line to the start of the source file, `poly3d.c`.

```
#define FPROMPT
```

This method should only be used if your C compiler does not allow you to define symbolic names at compile time (§8.1).

7. CONVENTIONS

7.1 Sign Conventions

Poly3D adopts the following sign conventions for various physical quantities.

Sign Convention for Tensors ($\underline{\underline{T}}$)

For normal tensor components (T_{11},T_{22},T_{33}), such as stress or strain, tension or extension is positive (Fig 3A). The shear component, T_{ij} (i j), on a surface with *positive* x_i direction is positive if it acts in the *positive* x_j direction. The shear component on a surface with outward normal pointing in the *negative* x_i direction is positive if it acts in the *negative* x_j direction.

Examples of tensor quantities include stress ($\underline{\underline{\sigma}}$) and strain ($\underline{\underline{\varepsilon}}$)

Box 9

Sign Convention for Vectors ($\underline{v}$)

The vector component, v_i, is positive if it acts in the positive x_i direction (Fig 3B).

Examples of vector quantities include displacement ($\underline{u}$), displacement discontinuity ($\underline{b}$), and traction ($\underline{t}$)

Box 10

Sign Convention for Angles of Rotation

A rotation, α_i, about the coordinate axis, $\hat{x}_i$, is positive if counterclockwise as viewed looking in the negative direction along $\hat{x}_i$ (Fig 3C).

Box 11

7.2 Objects, Elements, and Vertices

All polygonal elements in Poly3D belong to an object. An object is simply a collection of elements that represents a particular physical feature, such as a fault, that you are attempting to model (Fig. 4). An object may contain just one element, or it may contain many. The example problems presented in this manual demonstrate how objects can be used to organize and streamline Poly3D input and output.

In Poly3D a polygonal element is defined by first defining a series of vertices and then specifying the order in which they are connected to produce the element (Fig. 4). All of the vertices must lie in the same plane. Although an element itself has zero thickness, displacements are discontinuous when passing through the element from one side to the

other. The vector which describes the displacement discontinuity from one side of the element to the other is called the Burgers vector. Obviously, the direction of the Burgers vector obtained depends on which side of the element is taken as the reference. Poly3D adopts the following convention (Fig. 4):

Burgers Vector

The Burgers vector, $\underline{b}$, measures the displacement of the positive (+) side of the element relative to it's negative (-) side: $\underline{b} = \underline{u}^{+} - \underline{u}^{-}$.

Box 4

The element's positive and negative sides are distinguished by using the "right-hand rule" (Fig 4).

Positive (+) and Negative (-) Sides of an Element

Curl the fingers of the right hand in the plane of the element following the same order in which it's vertices were connected to produce the element. The thumb points from the negative (-) to the positive (+) side.

Box 5

Definition of the Element Normal ($\hat{n}$)

The unit vector orthogonal to the element plane which points from the negative (-) to the positive (+) side is the element normal, $\hat{n}$ This is the direction in which the thumb points when applying the right-hand rule of Box 5.

Box 6

7.3 Coordinate Systems

7.3.1 Default Global Coordinates

On startup, Poly3D pre-defines a global, right-handed coordinate system (x_1^g, x_2^g, x_3^g) in which $\hat{x}_1^g$ points East, $\hat{x}_2^g$ points North, and $\hat{x}_3^g$ points up (Fig. 5). For half-space problems, the origin is located at the earth's surface.

Default Global Coordinates

In the default global coordinate system (x_1^g, x_2^g, x_3^g), $\hat{x}_1^g$ points East, $\hat{x}_2^g$ points North, and $\hat{x}_3^g$ points up. For half-space problems, the origin is located at the earth's surface. The Poly3D name for default global coordinates is "global".

Box 7

7.3.2 Element Coordinates

Whenever a new element is defined, Poly3D links to it a local coordinate system (x_1^e, x_2^e, x_3^e). You may use this coordinate system to specify boundary conditions on the element and to display element information. The element coordinates are defined as follows:

Element Coordinates

Each element has linked to it an element coordinate system (x_1^e, x_2^e, x_3^e) with origin at the element center. Relative to the horizontal $x_1^g - x_2^g$ plane, $\hat{x}_1^e$ points down-dip. $\hat{x}_3^e$ is identical to element normal, $\hat{n}$ (Box 6). $\hat{x}_2^e$ points along strike in the direction given by the vector cross product $(\hat{x}_3^e \times \hat{x}_1^e)$. For horizontal elements, $\hat{x}_1^e$ and $\hat{x}_2^e$ point East and North if $\hat{x}_3^e$ points up or West and South if $\hat{x}_3^e$ points down. The Poly3D name for element coordinates is "elocal".

Box 8

7.3.3 User Coordinate Systems

Global and element coordinates are defined automatically by Poly3D. In many cases, it will be convenient to define additional right-handed coordinate systems of your own. These user coordinate systems $(\hat{x}_1^u, \hat{x}_2^u, \hat{x}_3^u)$ may have any origin and orientation you choose. Section §11.2 describes how to use and define user coordinate systems.

7.4 Input Units

Poly3D assumes dimensionally consistent units for physical quantities with dimensions of length or stress. Thus if you use kilometers for any quantity with dimensions of length (e.g. a coordinate position), you *must* use kilometers for *all* quantities with dimensions of length (e.g. displacement discontinuity). Likewise, if you use MPa for any quantity with dimensions of stress (e.g. a traction boundary condition), you *must* use MPa for *all* quantities with dimensions of stress (e.g. Young's modulus).

Physical Quantities With Dimensions of Length or Stress

Poly3D assumes dimensionally consistent units for physical quantities with dimensions of length or stress.

Box 12

Other physical quantities must be specified in the units listed below.

Strain

Strain is a dimensionless quantity. It should be specified in "units" of strain (*not* micro-strains, etc.)

Box 13

Angles of Rotation

Angles should be specified in degrees.

Box 14

7.5 Output Units

Poly3D uses the input units (Boxes 12-14) to display all computed quantities.

8. RUNNING POLY3D

8.1 Command Line Arguments

On UNIX machines, you run Poly3D by typing the program name at the system prompt. The program name may be followed by one or more options. For example, the UNIX command

```
poly3d -i infile -o outfile
```

runs Poly3D using the `infile` as the input file and `outfile` as the output file. When no input or output files are specified, Poly3D reads from the standard input and writes to the standard output. If you don't know what the standard input and output are for your machine, consult the documentation for your C-compiler.

Command Line Arguments accepted by Poly3D

`-i infile`	(optional)	Names the Poly3D input file.
`-o outfile`	(optional)	Names the Poly3D output file.

Box 15

Non-UNIX machines may require a different command to run Poly3D, such as double-clicking on an icon, that makes it difficult to specify command line arguments. In such cases, you may wish to compile Poly3D using the FPROMPT option described in §6.3.

8.2 Philosophy

We strongly advise against modifying the source code to Poly3D unless absolutely necessary. If you need to reorganize output for use in a plotting/contouring package, write a new program that converts the output file to the desired form. Likewise, if you want to add mesh generation, write a separate program which generates a Poly3D input file as its output. Many systems will allow you to chain several commands together using pipes or input/output redirects. For example, on UNIX systems you might type

```
alias mypoly3d 'meshgen | poly3d | convert'
```

This defines a new command (`mypoly3d`) which runs your mesh generation program (`meshgen`), pipes the output directly into Poly3D, and converts the resulting output (using the program, `convert`) to the proper format for your plotting/contouring package.

If you must alter the source code, *always* document your changes. Embed the distinctive string, `CHANGE` (all uppercase), in your comments, so that changes to the program will be easy to locate at a later date. Consider, for example, the following (fictional) bug fix made to a Poly3D source file.

```
/*-------------------------------------------------------------
* CHANGE: 12-Jun-1993 by Andrew L. Thomas
* Some incompetent programmer used the atan() function when
* he/she should have used atan2()!  Make sure to notify the
* Rock Fracture Project about this.
* FORMERLY:
*               alpha = atan(y/x);
*-------------------------------------------------------------*/
                alpha = atan2(y,x);
/* END CHANGES */
```

8.3 Error Messages

When Poly3D encounters an error from which it cannot recover, it prints an error message to the standard error (normally the monitor) and ceases execution. If you don't know what the standard error is for your machine, consult the documentation for your C compiler. For example, suppose the first line of the input file, `badfile.in`, attempts to assign an unknown constant

```
no_such_const  =  100.0
```

If you run Poly3D on a UNIX machine using `badfile.in` as input, you will get the following results:

```
Unix-Prompt>poly3d -i badfile.in

poly3d: Unknown constant, or incorrect format (badfile.in, line 1)
        no_such_const  =  100.0

Unix-Prompt>
```

Poly3D Error Messages

All Poly3D error messages begin with `poly3d:`. An error message *not* beginning with `poly3d:` indicates that Poly3D has encountered an error its programmer(s) did not anticipate.

Box 16

9. EXAMPLE PROBLEM #1

Perhaps the easiest way to learn about Poly3D is to work through an example problem. Here we consider the deformation at the earth's surface resulting from 1m of dip-slip on a buried, dipping, normal fault (Fig. 6). The planar fault surface is 2km-square and dips 45° to the East. Its center is buried to a depth of 1km, so the entire fault surface is under ground. For this problem, we will use Poly3D to evaluate stresses and displacements along an E-W trending line at the earth's surface that passes directly over the fault center.

The input file for this problem, `ex1.in`, is included in the standard Poly3D distribution kit. The following pages list `ex1.in` and the resulting output file, `example1.out`, generated by Poly3D (File Listings 1 & 2). You can learn much about Poly3D simply by looking over these two files. The following sections (§10-§13) dissect `ex1.in` and `example1.out` and discuss Poly3D input and output in detail. Together with additional examples presented in sections §14 and §15, they should provide you with all the information you need to set up and execute problems of your own.

10. INPUT FILE BASICS

Poly3D input files are designed to be human-readable. You should be able to read and understand a well-written input file without referring back to this manual. You must, however, remember the following input file basics.

10.1 Words

Poly3D reads and processes an input file line-by-line. After reading a line of input, Poly3D breaks each string of characters not separated by white space (spaces, tabs, etc.) into a separate "word". For example, the line

```
shear_mod   = 30000.0
```

contains three words: `"shear_mod"`, `"="`, and `"30000.0"`. Any string of characters within double quotation marks is treated as a single word, so the line

```
title1      = "Square, Buried, 45-Degree Dipping Normal Fault"
```

also contains just three words: `"title1"`, `"="`, and `"Square, Buried, 45-Degree Dipping Normal Fault"`.

10.2 Comments

Comments may be inserted anywhere in the input file using the comment character, '*'.

Comment Character

The comment character, '*', instructs Poly3D to ignore the remainder of the current input line.

Box 17

Thus, the line

```
rem_bc_type = strain            *(stress/strain)
```

contains just three words, while lines beginning with a '*', such as

```
**************************************************************************
*****
*                           Section 1: CONSTANTS
*
**************************************************************************
*****
```

contain zero words and are ignored by Poly3D. In this manual, input file comments are displayed in bold type. Well-written input files like `ex1.in` use comments liberally to maintain clarity.

10.3 Line Continuation

Some lines in Poly3D input files require a large number of columns to specify the required data. Computer or terminal screens with fixed 80-character widths may be unable to display long input lines without "wrapping" part of the text back to the left-hand margin. Because files with wrapped text are often difficult to read, Poly3D provides a line continuation character, '\'.

Line Continuation Character

A '\' at the end of a line instructs Poly3D to merge the following line with the present one.

Box 18

Because an extra space (' ') is inserted between lines as they are merged, individual words cannot be continued across lines. Thus, Poly3D converts the lines

```
This is\
really just\
one line.
```

into the single line

```
This is really just one line.
```

Note that the line continuation character, '\', has special meaning only when it appears as last character on the line. Thus, Poly3D will read and process these lines separately

```
The \ character is not in the right
position for these lines to be merged.
```

Of course, the above examples are intended only to illustrate how line continuation works. You would not want to try them in a real input file.

10.4 Names

Poly3D requires you to name many of the items you define. When providing names for a user coordinate system, observation grid, object, or vertex, remember these simple rules:

Names in Poly3D

A name may have any length and be composed of more than one word. Multiple-word names must be enclosed in double quotation marks. No two items of the same type (e.g. two observation grids) can have the same name.

Box 19

10.5 Case Sensitivity

Case Sensitivity

Poly3D is case sensitive when reading input files. Thus, the words "global", "Global", and "GLOBAL" are distinct and may not be used interchangeably.

Box 20

11. POLY3D INPUT FILES

Poly3D input files like `ex1.in` are divided into four main sections, each terminated by an end statement.

End Statement

Each section of a Poly3D input file, including the last, is terminated by an end statement. An end statement consists of the single word, `end,` on a line by itself (excluding comments).

Box 21

11.1 Section One: Constants

Poly3D input files begin with a series of 3-word assignment statements. Each assignment statement has the form

```
name = value
```

where `name` is the name of a Poly3D constant and `value` is its assigned value. If the value is omitted, as in

```
name =
```

the default or previously assigned value of the constant is unchanged. A Poly3D constant is either 1) a parameter that controls program execution or 2) a physical quantity who's value is not a function of position. Table 1 provides a complete list of Poly3D constants and their default values. Note that every Poly3D constant is assigned a value in `ex1.in`, even when that value is identical to the default. This strategy eliminates the need to remember the default values for Poly3D constants.

TABLE 1
Input File, Section One - Constants

Problem Titles
Enclose multiple-word titles in double quotes

Name	Default	Units	Description
title1	""	N/A	Problem Title (text string).
title2	""	N/A	Problem Subtitle (text string).

Options

Name	Default	Units	Description
half_space	"yes"	N/A ("yes"/"no")	Include Half-Space Effects? (§11.1.1)
null_value	-999.0	N/A	Null value to use for observation pts directly below a vertex (Box 1)
check_cond_num	"yes"	N/A ("yes"/"no")	Check matrix condition number? (§11.1.2)
print_elt_geom	"no"	N/A ("yes"/"no")	Print element geometries? (§11.1.3)
elt_geom_csys	"global"	N/A	Name of coord system to use when printing element geometries.

Elastic Constants
You must specify two (and only two)

Name	Default	Units	Description	(Symbol)
shear_mod	(none)	stress units	Shear Modulus	(μ or G)
psn_ratio	(none)	dimensionless	Poisson's Ratio	(ν)
youngs_mod	(none)	stress units	Young's Modulus	(E)
bulk_mod	(none)	stress units	Bulk Modulus	(K)
lame_lambda	(none)	dimensionless	Lame's Lambda	(λ)

Remote Boundary Conditions
*(Global coordinates; * these components must be zero for half-space problems)*

Name	Default	Units	Description
rem_bc_type	"stress"	N/A	Remote BC Type ("stress"/"strain")
s11r	0.0	stress/strain units	σ_{11}^{r} or ε_{11}^{r}
s22r	0.0	stress/strain units	σ_{22}^{r} or ε_{22}^{r}
s33r*	0.0	stress/strain units	σ_{33}^{r} or ε_{33}^{r}
s12r	0.0	stress/strain units	σ_{12}^{r} or ε_{12}^{r}
s13r*	0.0	stress/strain units	σ_{13}^{r} or ε_{13}^{r}
s23r*	0.0	stress/strain units	σ_{23}^{r} or ε_{23}^{r}

11.1.1 Half-Space Problems

The earth's surface can be treated as a traction-free boundary $(\sigma_{33} = \sigma_{31} = \sigma_{32} = 0)$.

This free surface boundary condition affects the distribution of stresses, strains, and

displacements in the upper crust. Free surface effects are included in all Poly3D calculations unless the constant, `half_space`, is set to `"no"`. There are two reasons why you may want to do this:

Reasons for Ignoring Half-Space Effects

1. You are solving a whole-space problem in solid mechanics unrelated to the earth sciences.
2. The geologic structures you are modeling *and* observation grids (§11.3) you define are buried very deeply relative to their length scale.

Box 22

Because fewer calculations are required, setting `half_space` to `"no"` may noticeably reduce the run-time required to complete Poly3D problems. Of course, increased execution speed is *not* a valid reason for ignoring half-space effects.

11.1.2 The Condition Number

Whenever traction boundary condition(s) are specified on an element(s), Poly3D must solve a system of linear equations to determine the corresponding unknown Burgers vector components (§A5). Even the best numerical methods for matrix inversion are susceptible to large roundoff errors when a matrix is ill-conditioned (close to singular). The condition number provides a quantitative measure of matrix conditioning. It is normally defined as the product of two matrix norms (Gerald and Wheatley, 1984). Poly3D uses the Gauss-Jordan method of elimination (Press et al., 1992) to solve its system of linear equations and calculates the matrix condition number as follows:

$$\mathrm{cond}(A) = \left\| A \right\|_\infty \left\| A^{-1} \right\|_\infty \tag{1}$$

where $\left\| A \right\|_\infty$ and $\left\| A^{-1} \right\|_\infty$ are the matrix-infinity norms (Gerald and Wheatley, 1984)

$$\left\| A \right\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} \left| a_{ij} \right| = \text{Maximum Row - Sum} \tag{2}$$

Meaning of the Condition Number

A small condition number indicates that the matrix is not prone to roundoff error during elimination and inversion. If the condition number is large (10^{12}), roundoff error may be significant, and you should mistrust whatever results Poly3D produces.

Box 23

User Caution: Numerical Error

Please note that matrix inversion is not the only potential source of numerical error. In other words, a small condition number is insufficient justification for blindly trusting the results of any numerical program.

Box 24

In order to find the condition number, Poly3D must calculate the inverse, A^{-1}, of matrix, A, rather than just its LU decomposition (A = LU). Thus, the condition number can be time consuming to compute. If you already know the condition number for a particular problem (e.g. from previous runs with the *same boundary conditions and element geometries*), you can instruct Poly3D not to compute it by setting the input file constant `check_cond_num` to `no`.

11.1.3 Printing Element Geometries

You may wish to depict the geometry of polygonal elements using a computer plotting package. Poly3D input files, with their use of multiple coordinate systems and separation of vertex from element definitions, are ill-suited for this purpose. You can overcome this problem by setting the input file constant `print_elt_geom` to `"yes"`. Poly3D will then print element geometries to the output file. The geometry of an element is given by the coordinates of its vertices, which are listed in the same order they were linked to produce the element. All vertex positions are printed using the coordinate system named by `elt_geom_csys`, which defaults to global coordinates. Section §12 describes the output format used to print element geometries in more detail.

11.1.4 Elastic Constants

Although two constants are sufficient to completely describe the behavior of a linear elastic material, several are in common usage. Given any two, the rest can be derived. See Jaeger and Cook (1979), chapter 5, for a review. Poly3D understands five elastic constants, and lets you choose which two of the five to define. Note that in `ex1.in`, all five of the elastic constants are listed, but only two are assigned values.

11.1.5 Remote Boundary Conditions

In Poly3D, you specify remote boundary conditions *in global coordinates* by assigning a value to each component of the symmetric remote stress ($\underline{\underline{\sigma}}^r$) or remote strain ($\underline{\underline{\varepsilon}}^r$) tensor (Fig 7). A specified remote strain is converted to remote stress using Hooke's Law (§A3). Because the earth's surface is postulated to be traction-free, the 33, 13, and 23 components of $\underline{\underline{\sigma}}^r$ and $\underline{\underline{\varepsilon}}^r$ must be zero for half-space problems.

Remote Boundary Conditions for Half-Space Problems

σ^r_{33}, σ^r_{13}, and σ^r_{23} (and hence ε^r_{33}, ε^r_{13}, and ε^r_{23}) must be zero for half-space problems

Box 25

A remote stress represents a homogeneous stress field over the entire elastic body. Poly3D calculates the total stress at a point by superposing the components of remote stress with the corresponding components of stress arising from displacement discontinuities across all boundary elements. The displacement components induced by a homogeneous remote stress are ignored when Poly3D calculates displacement at a point.

11.2 Section Two: User Coordinate Systems

In Poly3D, you will rarely use the default global coordinate system to enter data. In most cases, a user coordinate system (UCS) will be more convenient (§7.3). A UCS is defined by specifying the location of its origin and orientation of its axes relative to a *parent* coordinate system (Fig. 5). Any previously defined coordinate system, including the default global coordinates, may serve as the parent. There is no limit on the number of user coordinate systems that can be defined.

User coordinate systems are defined in the second section of Poly3D input files. Each definition requires a single, nine-word (or column) line. Table 2 lists the required data.

TABLE 2
Input File, Section Two - User Coordinate Systems

Col	Description	Suggested Col Title	Units
1	Name of user coordinate system	name	N/A

2	Name of parent coordinate system	parent	N/A
3-5	Location of Origin (x_{1o},x_{2o},x_{3o})	x*no*	length units
6-8	Rotations about parent axes (α_1,α_2,α_3)	rot*n*	degrees
9	Order in which (α_1,α_2,α_3) are applied (one of "123", "132", "213", "231", "312", "321")	rot order	N/A

11.2.1 Name and Parent

You must supply a name for each UCS you define (column 1). In addition, you must specify the name of a previously-defined coordinate system to serve as the UCS parent (column 2). Because user coordinate systems are defined before elements, element coordinates (§7.4.2) cannot serve as UCS parents.

11.2.2 Origin and Orientation

The location of a UCS origin is specified in columns 3-5 by its coordinates (x_{1o},x_{2o},x_{3o}) in the parent coordinate system (Fig. 5). The orientation of its axes is determined by three coordinate rotations (columns 6-8); one about each axis of the parent (Fig. 8). The order in which the coordinate rotations (α_1,α_2,α_3) are to be applied is given in the ninth column of UCS definitions.

The fault plane in `ex1.in` dips 45° to the East (Fig. 6). In order to specify the coordinates of fault corners, we defined a user coordinate system called FaultCS (x_1^f,x_2^f,x_3^f) whose $x_1^f - x_2^f$ plane contains the fault (Fig. 6).

```
*name      parent    x1o    x2o    x3o    rot1    rot2    rot3    rot
order
*--------- --------- ------ ------ ------ ------- ------- ------- --------
-
FaultCS    global    1.0    0.0    -1.0   0.0     45.0    0.0     321
```

The origin of FaultCS is located in default global coordinates at (1,0,-1) km (i.e. 1 km East of the global origin at a depth of 1km). The fault normal is given by $\hat{x}_3^f$, while $\hat{x}_1^f$ points down-dip and $\hat{x}_2^f$ points along strike.

You may often find it useful to define user coordinate systems (x_1^u,x_2^u,x_3^u) like FaultCS in which $\hat{x}_1^u$ points down-dip and $\hat{x}_2^u$ points along strike of the $x_1^u - x_2^u$ plane

(Fig. 9). If strike is measured clockwise from North and the dip direction is 90° clockwise from strike, the UCS should be defined as follows:

```
*name      parent    x1o    x2o    x3o    rot1    rot2    rot3    rot
order
*--------- --------- ------ ------ ------ ------- ------- ------- --------
-
...        global    ...    ...    ...    0.0     dip     -strike 321
```

where you substitute the appropriate values for *strike* and *dip*.

11.3 Section Three: Observation Grids

Poly3D is typically used to calculate the displacement, strain, or stress at specified points in an elastic body resulting from displacement discontinuities across one or more polygonal elements. An observation grid defines a series of regularly-spaced *observation points* and instructs Poly3D to calculate the displacement, strain, or stress for each one. The dimension of an observation grid determines how the observation points are distributed in space (Fig. 10).

Observation Grid Dimension

Observation grids of dimension *n* are given the following designations (Fig. 10).

Dimension	Designation
0	Observation Point
1	Observation Line
2	Observation Plane
3	Observation Network

Box 26

Observation grids are defined in the third section of Poly3D input files. There is no limit on the number of observation grids that can be defined. Each definition requires a single, nine- to fifteen-word (or column) line. Table 3 lists the required data.

User Caution: Observation Grids

Recall that displacements are discontinuous across an element surface (§5.2), and that strains and stresses along the element's edge are infinite. For this reason, you should avoid placing observation points on an element surface. Poly3D provides a different mechanism for calculating the traction and displacement discontinuity across an element (§11.4.1).

Box 27

TABLE 3
Input File, Section Three - Observation Grids

Col	Description	Suggested Col Title	Units
1	Observation grid name	name	N/A
2	Observation grid dimension	dim	N/A
3	Quantities to calculate at observation points	outp	N/A
4	Name of endpoint coordinate system	endpt csys	N/A
5	Name of observation point coordinate system	obspt csys	N/A
6	Name of output coordinate system	outp csys	N/A
7-9	Coords of beginning corner (x_{1beg},x_{2beg},x_{3beg})	x*n*beg	length units
10-12	Coords of ending corner (x_{1end},x_{2end},x_{3end})	x*n*end	length units
13-15	Num of obs pts along grid axes (n_{x1},n_{x2},n_{x3})	N*n*	N/A

11.3.1 Name and Dimension

You must supply a name and dimension for each observation grid you define (columns 1 & 2). The dimension of an observation grid is given by an integer number in the range 0-3.

11.3.2 Observation Grid Output

Column 3 is occupied by an "output string". Poly3D uses the output string to determine what data should be calculated at observation points in the grid.

Observation Grid Output String

An output string indicates what data should be calculated at observation points. The string may be composed of any combination of the following letters. No other characters (including spaces) are allowed.

Characters	Quantity
d	displacement vector
e	strain tensor
pe	principal strains
s	stress tensor
ps	principal stresses

Box 28

For example, the output string `dpsse` instructs Poly3D to calculate and print displacements, principal stresses, stresses, and strains for all points in the observation grid. The resulting output is formatted according to §12.2 below.

11.3.3 Coordinate Systems

For each observation grid you define, you must specify three coordinate systems (columns 4-6). The first is the *endpoint coordinate system* in which grid endpoints (see below) are specified *in the input file* . The second is the *observation point coordinate system* Poly3D should use when printing observation point positions *to the output file*. The third is the *output coordinate system* to use when printing the calculated displacements, strains, or stresses *to the output file*. In `ex1.in`, we wanted to calculate displacements and stresses in global coordinates along an E-W trending line at the earth's surface. For this observation line, it was most convenient to use the default global coordinates throughout.

```
* (1)    (2) (3)      (4)         (5)         (6)
*name    dim outp  endpt csys obspt csys outpt csys  ......
*------- --- ----- ---------- ---------- ---------
ObsLine  1   sd    global     global     global
```

11.3.4 0-D Observation Grids

0-D observation grids consist of a single observation point (Fig. 10A). The coordinates of the observation point are given in columns 7-9. Columns 10-15 are not required for 0-D grids.

11.3.5 1-D Observation Grids

The position and orientation of an observation line (Fig. 10B) is determined by the coordinates of its two endpoints, $\underline{x}_{beg}$ (columns 7-9) and $\underline{x}_{end}$ (columns 10-12). Column 13 gives the number of observation points to be distributed evenly along the line. Columns 14-15 are not required for 1-D grids. When calculating displacements, strains, and stresses at observation points, Poly3D traverses the line from $\underline{x}_{beg}$ to $\underline{x}_{end}$.

The observation line in `ex1.in` has beginning and ending points located at (-3,0,0) km and (3,0,0) km in global coordinates and contains seven observation points.

```
* (1)    (2)      (7)    (8)    (9)    (10)   (11)   (12) (13) (14) (15)
*name    dim ... x1beg  x2beg  x3beg  x1end  x2end  x3end  N1  N2  N3
*------- ---     ------ ------ ------ ------ ------ ------ --- --- ---
ObsLine  1       -3.0   0.0    0.0    3.0    0.0    0.0    7
```

11.3.6 2-D Observation Grids

The size and shape of a rectangular observation plane (Figs. 10C & 11) is determined by the coordinates of its endpoints, $\underline{x}_{beg}$ (columns 7-9) and $\underline{x}_{end}$ (columns 10-12). The line connecting $\underline{x}_{beg}$ and $\underline{x}_{end}$ must parallel one of the coordinate planes of the endpoint coordinate system (i.e. one of the following must be true: $x_{1beg} = x_{1end}$, $x_{2beg} = x_{2end}$, or $x_{3beg} = x_{3end}$). The resulting observation plane lies parallel to this coordinate plane, with $\underline{x}_{beg}$ and $\underline{x}_{end}$ at diagonally opposite corners. Each of the plane's four edges parallels a coordinate axis (Fig. 11).

Observation points are distributed over the surface of the observation plane in a uniform, rectangular grid. The number of points along the observation plane edges parallel to the x_i-axis is given by N_i (columns 13-15). N_i must be 1 for the coordinate axis perpendicular to the observation plane and 2 for the others.

11.3.7 3-D Observation Grids

A 3-D observation grid forms a rectangular box whose diagonally opposite corners are given by $\underline{x}_{beg}$ and $\underline{x}_{end}$ (Fig. 10D). The six sides of the box parallel the three coordinate planes of the endpoint coordinate system. Observation points are distributed throughout the box's volume in a uniform, rectangular grid. The number of points along a box edge parallel to the x_i-axis is given by N_i (columns 13-15), which must be 2 for all axes.

11.4 Section Four: Objects, Elements, and Vertices

Each line in the fourth and final section of a Poly3D input file defines an object, element, or vertex (§7.3). The first word of each line consists of a single letter which is used by Poly3D to determine the type of item being defined: "o" for object, "e" for element, and "v" for vertex. Definitions may occur in any order, subject to the following restrictions:

Restrictions on the Order of Object, Element, and Vertex Definitions

1. Because all elements must belong to a object, an object definition must precede the first element definition.

2. A vertex must already be defined before it can be used in an element.

Box 29

11.4.1 Objects

Lines beginning with an "o" define an object (§7.3). Objects are used to specify what physical feature (e.g. a fracture or fault) a particular group of elements represents. An element automatically belongs to the most recently defined object. Object definitions requires a minimum of two words (columns). Table 4 lists the required data.

TABLE 4
Object Definition
(Columns 3-4 may be omitted if no output is desired)

Col	Description	Suggested Col Title	Units
1	The letter 'o'	o	N/A
2	Object name	name	N/A
3	Output to calculate for element centers	outp	N/A
4	Coordinate system to use when printing element center positions	eltc csys	N/A

You must supply a name for each object you define in column 2. Column 3 is occupied by an "output string". Poly3D uses the output string to determine what quantities should be calculated and displayed for elements belonging to the object.

Object Output String

The output string indicates what data should be calculated for elements belonging to an object. The string may be composed of any combination of the following letters. No other characters (including spaces) are allowed.

Character	Quantity
t	Traction vector (at element centers)
b	Burgers vector (displacement discontinuity)

Box 30

Because stresses along the surface of an element are non-uniform, particularly near its edges (§5.2), tractions are always calculated at the element center.

In Poly3D output files, the position of an element in space is given by the coordinates of its center. Column 4 tells Poly3D what coordinate system to use when printing the locations of element centers belonging to the object. If no output is desired for an object,

columns 3 and 4 of the object definition may be omitted. In `ex1.in`, we define a single object named `Square Fault` and request both traction and displacement discontinuity output for the elements that belong to it.

```
(1) (2)             (3)       (4)
*o name             outp    eltc csys
*- ---------------- ------- ----------
o "Square Fault"    bt      FaultCS
```

11.4.2 Vertices

A line beginning with a "v" defines a vertex (§7.3). A vertex is just a point in space that may be used as an element corner. Vertex definitions require the six words (columns) listed in Table 5.

TABLE 5
Vertex Definition

Col	Description	Suggested Col Title	Units
1	The letter 'v'	v	N/A
2	Vertex name	name	N/A
3	Coord sys in which vertex position is specified	csys	N/A
4-6	Vertex Position (x_1,x_2,x_3)	x*n*	length units

You must assign a name to each vertex you define (column 2). Descriptive names such as `"SE fault corner"` are valid, but in many cases a simple number will suffice. In `ex1.in` we define four vertices, one for each corner of the square normal fault.

```
*(1) (2)     (3)       (4)     (5)     (6)
*v name     csys       x1      x2      x3
*- -------- ---------  ------  ------  ------
v  1        FaultCS    1.0    -1.0     0.0
v  2        FaultCS    1.0     1.0     0.0
v  3        FaultCS   -1.0     1.0     0.0
v  4        FaultCS   -1.0    -1.0     0.0
```

In `ex1.in`, vertex coordinates (columns 4-6) are given in the user coordinate system `FaultCS` (column 3). User coordinate systems often make it simpler to enter vertex and element data. They also make your input file easier to read. In `ex1.in`, it is immediately obvious that the four vertices define a square 2 km on a side. The same vertex definitions would be less transparent and more cumbersome if given in global coordinates:

```
*(1) (2)     (3)       (4)      (5)      (6)
*v name      csys      x1       x2       x3
*- --------  --------- -------  -------  --------
v  1         global    1.70711  -1.0     -1.70711
v  2         global    1.70711   1.0     -1.70711
v  3         global    0.29289   1.0     -0.29289
v  4         global    0.29289  -1.0     -0.29289
```

11.4.3 Elements

Lines beginning with an "e" define a polygonal element (§7.3). The geometry of an element is determined by the locations of its vertices and the order in which they are connected to produce the element. All of the vertices must lie in the same plane. The required data is listed in Table 6. Note that the number of element vertices (given in column 2) determines how many columns of data are required.

Table 6
Element Definitions

Col	Description	Suggested Col Title	Units
1	The letter 'e'	e	N/A
2	Number of vertices (sides)	#vert	N/A
3	Coord sys in which boundary conds are specified	BC csys	N/A
4	Boundary condition type	BC type	N/A
5-7	Magnitude of boundary cond components *n*	BC*n*	stress or length units
8-10	Names of 1st,2nd, & 3rd vertices	v*n*	N/A
11-	Names of additional vertices (4th, 5th, ...)	v*n*	N/A

In `ex1.in`, we define a single, four-sided element that represents the square fault plane.

```
*(1) (2)   (3)      (4)    (5)       (6)       (7)       (8) (9) (10) (...)
*e #vert BC csys   BC type BC1       BC2       BC3        v1 v2 v3 ...
*- ----- --------- ------- --------- --------- --------- -- -- -- --
e  4     elocal    bbb     1.0e-03   0.0       0.0        1  2  3  4
```

Columns 3-7 are used to specify boundary conditions on the element. Boundary conditions are specified by giving three components of displacement discontinuity or traction resolved at the element center. Each boundary condition component points along one axis of the coordinate system given in column 3.

Boundary Condition Coordinate System

The boundary conditions on an element may be specified in any of the following coordinate systems: default global coordinates, user coordinates, or element local coordinates.

Box 31

In columns 5-7 are listed the magnitudes of the three boundary condition components. Poly3D looks at the three character string in column 4 to determine the boundary condition type for each component.

Boundary Condition Type

Column 4 of an element definition is a three-character string whose i^{th} letter indicates the boundary condition type for the i^{th} boundary condition component. The string may be composed of any combination of the following letters. No other characters (including spaces) are allowed.

Character	Boundary Condition Type
t	Traction component (at element center)
b	Burgers vector component (displacement discontinuity)

Box 32

The remaining columns (8-) list the vertices *in the order they should be connected* to produce the element. A minimum of three vertices is required. The number of vertices listed must match the number given in column 2 of the element definition.

For the single element in `ex1.in`, we supply a complete Burgers vector for the three boundary condition components. Boundary conditions are given in the element coordinate system (§7.3.2), so b_1 points down-dip and has the desired magnitude of 1.0e-03 km (i.e. 1 m). If the vertices had been specified in the opposite order, the element would have the same shape, but its positive and negative sides would swap positions. The b1 component of displacement discontinuity would therefore have to be changed to -1.0e-03 to maintain a normal (instead of reverse) sense of slip. The resulting element definition would then look like this:

```
*e #vert BC csys   BC type BC1       BC2       BC3       v1 v2 v3 ...
*- ----- --------- ------- --------- --------- --------- -- -- -- --
e  4     elocal    bbb     -1.0e-03  0.0       0.0       4  3  2  1
```

Note that coordinate systems `elocal` and `FaultCS` happen to be identical for this element. Thus `FaultCS` could have been used in place of `elocal` (in column 3) when specifying the boundary conditions.

12. OUTPUT FILES

Poly3D output files are largely self-explanatory. Printed first are the program name and version number and the date on which it was compiled (File Listing 2). Next are listed the input file name, the problem titles, and the values calculated for the elastic constants. The null value used for observation points directly below a vertex (Box 1) is printed next, followed by the matrix condition number (§11.1.2).

If the input file constant `print_elt_geom` is set to `yes`, Poly3D will step through the objects in the order they were defined and print information on the geometry of each element. Vertex positions are given in the coordinate system named by the Poly3D constant `elt_geom_csys` (§11.1.3). The resulting output is formatted according to Table 7.

TABLE 7
Output Columns for Element Geometries

Col	Col Label	Units	Description
1	ELT	N/A	Element number within the object
2	Vertex Name	N/A	Vertex Name
3-5	X*n*	length units	Vertex Coordinates

The remainder of the output file lists quantities calculated for objects and observation grids. Poly3D processes objects first, stepping through them in the order they were defined and printing the requested output for each. When finished with the objects, Poly3D processes observation grids in the same manner.

12.1 Object Output

For each object, Poly3D prints titles giving the object name and coordinate system used to display the locations of element centers (§11.4.1). The remaining output is divided into two tables. The first gives the displacement discontinuity across each element, while the second lists the tractions resolved at element centers. Either part may be omitted if not requested in the object's output string (§11.4.1). Poly3D steps through elements in the order they were defined and prints the output for each element *in the same*

coordinate system used to specify its boundary conditions. This coordinate system is listed in the final output column. The meaning of each column for displacement discontinuity and traction output is listed in Tables 8 and 9.

TABLE 8
Output Columns for Displacement Discontinuity

Col	Col Label	Units	Description
1	ELT	N/A	Element number
2-4	X*n*C	length units	Location of element center
5	B1	length units	b_1 component of displ discontinuity ($b_1 = u_1^+ - u_1^-$)
6-7	U1(+/-)	length units	u_1 component of displ of +,- sides
8	B2	length units	b_2 component of displ discontinuity
9-10	U2(+/-)	length units	u_2 component of displ of +,- sides
11	B3	length units	b_3 component of displ discontinuity
12-13	U3(+/-)	length units	u_3 component of displ of +,- sides
14	Coord Sys	N/A	Boundary condition coord system

TABLE 9
Output Columns for Traction

Col	Col Label	Units	Description
1	ELT	N/A	Element number
2-4	X*n*C	length units	Location of element center
5-7	T*n*	stress units	t_n component of traction
8	Coord Sys	N/A	Boundary condition coord system

12.2 Observation Grid Output

For each observation grid, Poly3D prints titles giving the observation grid name and dimension, the coordinate system used to display observation point positions, and the coordinate system used to display displacement, strain, and stress output. If requested in the observation grid's output string (§11.3.2), the computed values for displacement, strain, principal strain, stress, and principal stress are displayed in separate output tables. The meaning of each output table's columns is listed in Tables 10 through 11.

TABLE 10
Output Columns for Displacement

Col	Col Label	Units	Description
1-3	X*n*	length units	Location of observation point
4-5	U*n*	length units	Displacement components

TABLE 11
Output Columns for Strain

Col	Col Label	Units	Description
1-3	X*n*	length units	Location of observation point
4	E11	dimensionless	ε_{11} component of strain
5	E22	dimensionless	ε_{22} component of strain
6	E33	dimensionless	ε_{33} component of strain
7	E12	dimensionless	ε_{12} component of strain
8	E13	dimensionless	ε_{13} component of strain
9	E23	dimensionless	ε_{23} component of strain

TABLE 12
Output Columns for Principal Strains

Col	Col Label	Units	Description
1-3	X*n*	length units	Location of observation point
4-6	N*n*	length units	Unit vector in direction of ε_1
7	E1	dimensionless	ε_1 component of principal strain
8-10	N*n*	length units	Unit vector in direction of ε_2
11	E2	dimensionless	ε_2 component of principal strain
12-14	N*n*	length units	Unit vector in direction of ε_3
15	E3	dimensionless	ε_3 component of principal strain

TABLE 13
Output Columns for Stress

Col	Col Label	Units	Description
1-3	X*n*	length units	Location of observation point
4	SIG11	stress units	σ_{11} component of stress
5	SIG22	stress units	σ_{22} component of stress
6	SIG33	stress units	σ_{33} component of stress
7	SIG12	stress units	σ_{12} component of stress
8	SIG13	stress units	σ_{13} component of stress
9	SIG23	stress units	σ_{23} component of stress

TABLE 14
Output Columns for Principal Stresses

Col	Col Label	Units	Description
1-3	X*n*	length units	Location of observation point
4-6	N*n*	length units	Unit vector in direction of σ_1
7	SIG1	stress units	σ_1 component of principal stress
8-10	N*n*	length units	Unit vector in direction of σ_2
11	SIG2	stress units	σ_2 component of principal stress
12-14	N*n*	length units	Unit vector in direction of σ_3
15	SIG3	stress units	σ_3 component of principal stress

13. EXAMPLE PROBLEM #1 (REVISITED)

Let us return our attention to the square normal fault of example problem #1 (§9). Here we consider the deformation at the earth's surface resulting from 1m of uniform dip-slip across a buried, dipping normal fault (Figs. 6 & 12A). The planar fault surface is 2km-square and dips 45° to the East. Because the fault in this problem can be represented by a single, rectangular element, we can compare the Poly3D results with those given by Dis3D, a boundary element code based on the analytical solution for rectangular dislocations (Erickson, 1986). For this problem, the displacements, strains, and stresses calculated at the earth's surface by Poly3D and Dis3D agree to within 0.01%. Therefore, only the Poly3D results are shown.

The E-W trending observation line at the earth's surface defined in `ex1.in` contains just seven observation points (Fig. 6; File Listing 1), each 1km apart. The vertical component of displacement calculated at each of these points is plotted in Fig. 12B. The continuous-looking curve against which these results are plotted was generated by placing 61 observation points along the line, each 0.1 km apart, and resolving the problem. Note the asymmetric distribution of the vertical component of surface displacement, with over 40 cm of downdrop above the hanging wall of the fault and comparatively little (< 5 cm) uplift above the footwall.

A more complete picture of surface displacements can be obtained by replacing the observation line of `ex1.in` with an observation plane. In Fig 12C, we contour the vertical component of displacement calculated using a horizontal 3 x 3 km observation plane at the earth's surface. The observation plane contains 31 x 31 points on a grid spacing of 0.2km. Using global coordinates for the endpoint coordinate system (§11.3.3), this grid was defined as follows:

```
* (1)    (2)      (7)    (8)    (9)    (10)   (11)   (12) (13) (14) (15)
*name    dim ... x1beg  x2beg  x3beg  x1end  x2end  x3end  N1   N2   N3
*------- ---     ------ ------ ------ ------ ------ ------ ---  ---  ---
ObsGrid  2       -3.0   -3.0   0.0    3.0    3.0    0.0    31   31   1
```

14. EXAMPLE PROBLEM #2

For our second example problem, we consider a planar, horizontal, circular crack 1km in diameter subjected to a uniform internal fluid pressure of 1MPa and zero remote stress (Fig. 13). For this problem, we use Poly3D to calculate the crack-wall opening distribution and the vertical component of normal stress in the neighborhood of the crack. Two cases are considered: 1) the crack is embedded in an elastic whole-space and 2) the crack is buried 0.5km below the surface of an elastic half-space (i.e. the earth's surface). The whole-space results for the pressurized, circular crack are compared with the analytical solution of Sneddon (1946).

The Poly3D input and output files for the whole-space problem (`ex2ws.in` and `ex2ws.out`) are presented on the following pages (File Listings 3 & 4). The input file, `ex2hs.in`, for the half-space problem is identical to `ex2ws.in`, with the exception that the constant, `half_space`, is set to `yes` rather than `no`. Thus only the output file, `ex2hs.out`, is shown (File Listing 5) for the half-space problem.

The circular crack surface is divided into 40 triangular elements, each with approximate boundary conditions of zero shear traction and 1MPa normal traction (Fig 13B). The crack-wall opening displacements calculated by Poly3D for the whole- and half-space problems are shown in Fig. 13D. Note that the crack in the whole-space opens symmetrically. The heavy, solid line representing the analytical solution of Sneddon (1946) closely matches the Poly3D results. Because of its interaction with the adjacent free surface, the crack in the half-space has a larger, asymmetric opening (Fig. 13D).

In Fig 14, we also plot the vertical component of stress in the neighborhood of the crack. These contours were generated by calculating the stress at 861 points distributed over a vertical observation plane passing through the crack center (Fig. 12A). Again, the symmetric stress distribution around the crack in the whole-space agrees well with the analytical results of Sneddon (1946). The stress distribution around the crack in the half-

space is asymmetric due to its interaction with the adjacent free surface. Note that the vertical component of stress goes to zero as the free surface is approached.

15. EXAMPLE PROBLEM #3

In this problem, we consider the stresses induced by a 1cm-diameter spherical void in an infinite elastic body subject to a uniaxial remote tension of $\sigma^{r}_{33} = 1\text{MPa}$ (Fig. 15A). The analytical solution for this problem is presented by Timoshenko & Goodier (1970). We have not yet used Poly3D to successfully solve this problem, but present the input file, `ex3.in`, because it provides an elegant demonstration of how multiple coordinate systems can be used to define complex surfaces with polygonal elements.

In `ex3.in`, pentagonal and hexagonal elements are assembled together in the manner of a soccer ball to produce a spherical cavity. The twelve identical pentagons are distributed evenly over the surface of the sphere. The twenty identical hexagons are formed by joining together the vertices of adjacent pentagons. For a 1cm-diameter sphere, the required pentagon radius (the distance from its center to a vertex) is 0.187cm. If a pentagon sits perpendicular to the x_3 with one vertex lying along x1 and its center at the coordinate origin (Fig 15B), the coordinates of its five vertices are given in Table 15.

TABLE 15
Coordinates of Pentagonal Element Vertices

vertex #	x1	x2	x3
1	0.187	0.000	0.000
2	0.058	0.178	0.000
3	-0.151	0.110	0.000
4	-0.151	-0.110	0.000
5	0.058	-0.178	0.000

In order to describe the geometry of the soccer ball-like cavity, we define twelve such (user) coordinate systems, `PentCS1-PentCS12`, one for each pentagon. The coordinates of all vertices on the sphere are then defined by repeating the five vertex definitions listed in Table 15 for each of the twelve coordinate systems. Pentagonal and hexagonal elements are then formed by joining vertices together in the appropriate order. The normal to each element (§7.2) points radially outward from the center of the sphere. Zero

traction boundary conditions are prescribed for each element to produce the desired free (interior) surface of the cavity.

Poly3D is not yet able to calculate a correct solution for the spherical cavity. The calculated tractions, displacement discontinuities and elastic fields show a puzzling asymmetry that appears to result from an unsolved programming error. The results for example problem #3 will be presented in future versions of this manual once these difficulties have been resolved.

APPENDIX: A
POLYGONAL ELEMENT IMPLEMENTATION

At the heart of Poly3D are the equations derived by Comninou and Dunders (1975) for an angular dislocation in an elastic half-space (Fig A1). The angular dislocation lies in a vertical plane with one leg perpendicular to the free surface. Its two legs subtend an angle, β, and extend to infinity from a common vertex, $\underline{\xi}$. The uniform displacement discontinuity across the dislocation is given by its Burgers vector, $\underline{b}$.

A1. The Angular Dislocation

At any point, $\underline{x}$, in the elastic body, the displacement component, u_i, due to an angular dislocation is a linear function of the three Burgers vector components

$$u_i = U_{ij}\left(\underline{x};\, \underline{\xi}, \beta, \omega\right) b_j \qquad \text{(A1)}$$

where the U_{ij} are *displacement influence coefficients* for the angular dislocation. More specifically U_{ij} gives the displacement component, u_i, at position, $\underline{x}$, resulting from a Burgers vector component, b_j, of unit magnitude on an angular dislocation defined by $\underline{\xi}$, β, and ω. The strain at any point in the elastic body due to an angular dislocation can be derived from equation A1 as follows

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) = E_{ijk}\left(\underline{x};\, \underline{\xi}, \beta, \omega\right) b_k \qquad \text{(A2)}$$

where E_{ijk} are the *strain influence coefficients* for the angular dislocation,

$$E_{ijk} = \frac{1}{2}\left(\frac{\partial U_{ik}}{\partial x_j} + \frac{\partial U_{jk}}{\partial x_i} \right) \qquad \text{(A3)}$$

Poly3D uses equations (A1) and (A2) when calculating displacement and strain influence coefficients for a polygonal element. Equations for U_{ij} that appear in the source code are taken directly from the results of Comninou and Dunders (1975) with the corrections listed in Appendix B. The lengthily expressions for E_{ijk} were derived by partial differentiation of the equations for U_{ij} and substituting the results into equation

(A3). This work was greatly facilitated through the use of *Mathematica* (Wolfram Research, 1991), a software package for doing symbolic mathematics on a computer. Because the equations for U_{ij} and E_{ijk} are given in local coordinates centered at the angular dislocation's vertex, Poly3D must perform the appropriate coordinate transformations when superposing them to yield U^e_{ij} and E^e_{ijk} for an entire polygonal element (§A2).

A2. Building a Polygonal Element

Jeyakumaran et al. (1992) describe how six angular dislocations with identical Burgers vector, $\underline{b}$, can be superposed to yield an arbitrarily-oriented triangular element. Each side of the triangle is broken down into two angular dislocations, which when superposed yield a *dislocation segment* (Fig A1). When the three dislocation segments representing the three sides of the triangle are superposed, the displacement discontinuities along their vertical legs cancel, leaving a displacement discontinuity only in the triangle. The total displacement resulting from a triangular dislocation is given by

$$u_i = \sum_{n=1}^{6} U^n_{ij}\, b_j \qquad \text{(A4)}$$

where U^n_{ij} are the displacement influence coefficients U_{ij} due to the n^{th} angular dislocation.

To create polygonal elements with more than three sides, we follow the same procedure as for triangular elements. In general, an N-sided polygon requires 2N angular dislocations, so the total displacement field is given by

$$u_i = \sum_{n=1}^{2N} U^n_{ij}\, b_j \qquad \text{(A5)}$$

Note that equation (A5) can be rewritten as follows

$$u_i = U^e_{ij}\left(\underline{x};\ \underline{\xi}^1, \underline{\xi}^2, \ldots \underline{\xi}^{2N}\right) b_j \quad ; \quad U^e_{ij} = \sum_{n=1}^{2N} U^n_{ij} \qquad \text{(A6)}$$

where U^e_{ij} are the displacement influence coefficients for the entire N-sided polygonal element. In a similar manner, the strain field resulting from an N-sided polygonal element is given by

$$\varepsilon_{ij} = E^e_{ijk}\left(\underline{x};\ \underline{\xi}^1, \underline{\xi}^2, \ldots \underline{\xi}^{2N}\right) b_k \quad ; \quad E^e_{ijk} = \sum_{n=1}^{2N} E^n_{ijk} \tag{A7}$$

where E^n_{ijk} are the strain influence coefficients for the n^{th} angular dislocation, and E^e_{ijk} are the strain influence coefficients for the entire N-sided polygonal element.

A3. Stress and Traction Influence Coefficients

It is a simple matter to calculate stress influence coefficients for a polygonal element, provided the strain influence coefficients and elastic constants are known. In an elastic body, stress and strain are related according to Hooke's Law

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \lambda\varepsilon_{nn}\delta_{ij} \tag{A8}$$

where μ is the shear modulus, λ is Lamé's lambda, and δ_{ij} is the Kroenecker delta.

$$\delta_{ij} = \begin{cases} 1; & (i = j) \\ 0; & (i \neq j) \end{cases} \tag{A9}$$

Substituting equation (A7) into (A8) yields

$$\sigma_{ij} = S^e_{ijk}\left(\underline{x};\ \underline{\xi}^1, \underline{\xi}^2, \ldots \underline{\xi}^{2N}\right) b_k \quad ; \quad S^e_{ijk} = 2\mu E^e_{ijk} + \lambda E^e_{nnk}\delta_{ij} \tag{A10}$$

where S^e_{ijk} are the stress influence coefficients for an N-sided polygonal element. Poly3D calculates stress influence coefficients in exactly this manner; that is, by first calculating the strain influence coefficients, E^e_{ijk}, and substituting the results into equation (A10).

The traction, $\underline{t}$, resolved at point, $\underline{x}$, on a plane with unit normal vector, $\hat{n}$, is given by Cauchy's formula

$$t_i = n_j\sigma_{ji} \tag{A11}$$

Substituting equation (A10) into (A11) yields

$$t_i = T^e_{ik}\left(\underline{x}, \hat{n};\ \underline{\xi}^1, \underline{\xi}^2, \ldots \underline{\xi}^{2N}\right) b_k \quad ; \quad T^e_{ik} = n_j S^e_{jik} \tag{A12}$$

where T^e_{ik} are the traction influence coefficients for an N-sided polygonal element. Because the stress and traction influence coefficients are easy calculate from E^e_{ij}, Poly3D does not maintain separate functions for computing S^e_{ijk} and T^e_{ik}. Instead, E^e_{ij} is calculated first. S^e_{ijk} and T^e_{ik} are then determined using equations (A10) and (A12).

A4. Superposition of Polygonal Elements

If more than one polygonal element is contained in the elastic body, the total elastic field is determined by superposition. For example, the displacement, $\underline{u}$, at point, $\underline{x}$, resulting from M polygonal elements is given by

$$u_i = \sum_{m=1}^{M} \overset{m}{U}{}^e_{ij} \overset{m}{b}_j \qquad \text{(A13)}$$

where $\overset{m}{U}{}^e_{ij}$ are the displacement influence coefficients U^e_{ij} due to the m^{th} element and $\overset{m}{b}_j$ is the Burgers vector component, b_j, on the m^{th} element.

A5. Approximate Traction Boundary Conditions

In order to calculate displacements, strains, and stresses in the elastic body, we must know all three components of the Burgers vector, $\underline{b}$ for each polygonal element. When displacement discontinuity boundary conditions (b_1,b_2,b_3) are given for each element, we can proceed directly to superposition, as in equation (A13). However, when an approximate traction boundary condition component, $\overset{\alpha}{t}_i$ is specified for an element, α, we must first determine the Burgers vector component $\overset{\alpha}{b}_i$ required to produce $\overset{\alpha}{t}_i$ at the element center.

Recall that traction is non-uniformly distributed over an element with uniform displacement discontinuity. Thus, specifying a traction component at the element center only constitutes an *approximate* boundary condition. The traction, $\overset{\alpha\beta}{t}_i$, at the center of element α due to the displacement discontinuity on an N-sided polygonal element, β, is given by

$$ {}^{\alpha\beta}t_i = {}^{\alpha\beta}T^{e}_{ik}\left(\underline{x}, \underline{n}; \underline{\xi}^{1}, \underline{\xi}^{2}, \ldots \underline{\xi}^{2N} \right) {}^{\beta}b_k \qquad \text{(A14)} $$

The total traction, ${}^{\alpha}t_i$, at the center of element α is found by superposition over all M elements

$$ {}^{\alpha}t_i = \sum_{\beta=1}^{M} {}^{\alpha\beta}T^{e}_{ik}\ {}^{\beta}b_k \qquad \text{(A15)} $$

Each traction boundary condition component, ${}^{\alpha}t_i$, leads to one equation of the form (A15) and one unknown ${}^{\alpha}b_i$. In general, if a total of Q traction boundary condition components are specified on the polygonal elements in an elastic body, we must solve a system of Q linear equations for Q unknowns (decompose a Q by Q matrix) in order to determine complete Burgers vectors for all M elements.

A6. Singularities

Because the displacement discontinuity across an angular dislocation is uniform, the strains and stresses along its vertical and plunging legs are infinite (§5.2). When superposed with other angular dislocations to create a polygonal element, these singularities disappear except along the portion of the plunging leg that is incorporated into the polygonal element.

When computing the stress or strain at a point, Poly3D calculates the contribution of each angular dislocation separately. Therefore, it will return an infinite intermediate result (NaN) for stress at any point on an angular dislocation leg, *even though that singularity should cancel upon superposition with other dislocations*. The singularity on the dipping leg can be avoided by simple geometric tricks, but the vertical leg remains an unavoidable nuisance (Fig. A2). For this reason, Poly3D cannot calculate the stress or strain at any point directly beneath an element vertex (i.e. along the vertical leg). Thus, no element can be defined with its center directly below another's vertex. For observation points below a vertex, Poly3D indicates that the elastic field quantities could not be calculated by printing the constant, `null_value` (§11.1), as their value.

APPENDIX B
TYPOGRAPHICAL ERRORS IN COMNINOU & DUNDURS (1975)

Several typographical errors appear in the equations given by Comninou and Dundurs (1975) for the displacement field induced by an angular dislocation in an elastic half-space . The corrections incorporated into Poly3D are listed below.

Equation (20), last line, first term:

$$\frac{y^2(\bar{y}_3 - a)\cot\beta}{\bar{R}(\bar{R} + \bar{z}_3)}\{ \ldots \quad \text{should be} \quad \frac{y_2(\bar{y}_3 - a)\cot\beta}{\bar{R}(\bar{R} + \bar{z}_3)}\{ \ldots$$

Equation (23), first line, last term:

$$\{\log(R - z_3) + \log(R + \bar{z}_3)\} \quad \text{should be} \quad \{\log(R - z_3) + \log(\bar{R} + \bar{z}_3)\}$$

Equation (23), second line, last term:

$$\ldots + \frac{1}{\bar{R}(\bar{R} - \bar{z}_3)}\Big\} \quad \text{should be} \quad \ldots + \frac{1}{\bar{R}(\bar{R} + \bar{z}_3)}\Big\}$$

Equation (27), first line, first term:

$$(1 - 2\nu)\Big\{\frac{y_2}{\bar{R} + \bar{y}_2} \ldots \quad \text{should be} \quad (1 - 2\nu)\Big\{\frac{y_2}{\bar{R} + \bar{y}_3} \ldots$$

Equation (27), first line, last term:

$$\ldots - \frac{y^2\cos\beta}{\bar{R} + \bar{z}_3}(\cos\beta + \ldots \quad \text{should be} \quad \ldots - \frac{y_2\cos\beta}{\bar{R} + \bar{z}_3}(\cos\beta + \ldots$$

Equation (28), first line, last term:

$$\ldots + \frac{\bar{z}_1}{\bar{R} + \bar{y}_3}(\cos\beta + \ldots \quad \text{should be} \quad \ldots + \frac{\bar{z}_1}{\bar{R} + \bar{z}_3}(\cos\beta + \ldots$$

APPENDIX C
SOURCE FILE CONTENTS AND DEPENDENCIES

The C language code for Poly3D is distributed over several *source* and *include* files (Table C1). The source files (with names ending in `.c`) contain most of the actual code. Include files (with names ending in `.h`) are used to define function prototypes, external variables, constants, and macros that are shared between source files. Fig. C1 depicts the dependencies between source and include files. Changes made to the program will not take effect until the affected files have been recompiled. On many computers, a makefile may simplify the task of compiling and maintaining Poly3D. A sample makefile for UNIX systems is provided in File Listing C1.

TABLE C1
Poly3D Source Files

Source File	Include File	Contents/Purpose
`poly3d.c`	(none)	Main program and support functions
`elastic.c`	`elastic.h`	Routines for solving equations in linear elasticity
`getopt.c`	`getopt.h`	A function for processing command line arguments
`getwords.c`	`getwords.h`	A function for reading lines of input and dividing them into their component words
`infcoeff.c`	`infcoeff.h`	Routines for calculating displacement and strain influence coefficients for the angular dislocation.
`matrix.c`	`matrix.h`	Functions that perform 3-D matrix, vector, and tensor operations
`nr.c`	`nr.h`	Functions adapted from *Numerical Recipes* (§2.1)
`nrutil.c`	`nrutil.h`	Utility functions taken from *Numerical Recipes* (§2.1)
`safetan.c`	`safetan.h`	"Safe" versions of the tan(), atan(), and atan2() functions that check for and avoid singularities
(none)	`pi.h`	An include file that defines the value of PI = 3.1415...

APPENDIX D
SOURCE CODE LISTINGS

This appendix contains the complete source code for Poly3D. The name and starting page number for each source file is listed in the table of contents.

REFERENCES CITED

Brown, R. L., 1975, A dislocation approach to plate interaction: Ph.D. Thesis, Massachusetts Institute of Technology, Department of Earth and Planetary Sciences, 449 p.

Comninou, M. A. and Dunders, J., 1975, The angular dislocation in a half-space: Journal of Elasticity, v. 5, p. 203-216.

Crouch, S. L. and Starfield, A. M., 1983, *Boundary element methods in solid mechanics: With applications in rock mechanics and geological engineering*: London, Unwin Hyman, 322 p.

Erickson, L. L., 1986, A three-dimensional dislocation program with applications to faulting in the earth: M.S. Thesis, Stanford University.

Fung, Y. C., 1969, *A first course in continuum mechanics*: Englewood Cliffs, NJ, Prentice-Hall, 301 p.

Gerald, C. F. and Wheatley, P. O., 1984, *Applied Numerical Analysis*: Reading, Massachusetts, Addison-Wesley, 579 p.

Jaeger, J. C. and Cook, N. G. W., 1979, *Fundamentals of Rock Mechanics*: London, Chapman and Hall, 593 p.

Jeyakumaran, M., Rudnicki, J. W. and Keer, L. M., 1992, Modeling slip zones with triangular dislocation elements: Seismological Society of America Bulletin, v. 82, p. 2153-2169.

Kernighan, B. W. and Ritchie, D. M., 1988, *The C programming language*: Englewood Cliffs, New Jersey, Prentice Hall, 272 p.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., 1992, *Numerical recipes in C: The art of scientific computing*: Cambridge, Cambridge University Press, 994 p.

Sneddon, I. N., 1946, The distribution of stress in the neighborhood of a crack in an elastic solid: Proc. Roy. Soc., Section A, v. 187, p. 229-260.

Timoshenko, S. P. and Goodier, J. N., 1970, *Theory of Elasticity*: New York, McGraw-Hill, 567 p.

Wolfram Research, I., 1991, *Mathematica*: Champaign, Illinois, Wolfram Research, Inc., 961 p.

Yoffe, E. H., 1960, The angular dislocation: Philosophical Magazine, v. 5, p. 161-175.