**SANDIA REPORT**
SAND2006-5654
Unlimited Release
Printed September 2006

# Guide to Coupled Electrostatic-Structural Analyses with Arpeggio

Vicki L. Porter

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2006-5654
Unlimited Release
Printed September 2006

# Guide to Coupled Electrostatic-Structural Analyses with Arpeggio

Vicki L. Porter
Computational Solid Mechanics and Structural Dynamics Department
Engineering Sciences Center
Sandia National Laboratories
Box 5800
Albuquerque, NM 87195-0827

**Abstract**

Many applications in micromechanical systems (MEMS) involve electrostatically actuated parts. Arpeggio is a code for facilitating loose coupling between computational mechanics modules in a parallel computing environment. This document describes how to use Arpeggio for coupled elecromechanical analyses using examples commonly encountered in MEMS applications, namely the response of structures to loads imposed by electrostatic fields. For this type of analysis, Arpeggio is used to couple Adagio, a three-dimensional finite element code for nonlinear, quasistatic or implicit dynamic analysis of three-dimensional structures, with BEM, a boundary integral method code for the analysis of electrostatic fields. This guide describes the methodology used for the loose coupling and the commands the user needs in an input file to perform such an analysis. All commands related to coupled analyses are described and examples are provided.

Intentionally Left Blank

**Acknowledgments**

The author would like to thank Samuel Subia and James Overfelt for their timely contributions to coding in BEM and the SIERRA framework to facilitate the coupled analyses and Jordan Massad for his willingness to put the code to the test. Sections of this manual that relate to domain commands and input file conventions common to all Sierra applications have been taken in large part from the Presto Users' Manual written by Richard Koteras.

Intentionally Left Blank

# Contents

# List of Figures

# Chapter 1

# Overview

## 1.1 SIERRA Architecture

Arpeggio is a multiphysics application code built on the SIERRA framework. The SIERRA framework provides services for data management in a parallel computing environment. It also imposes a specific hierarchy on any code architecture built on the framework, and subsequently, on the organizational structure of the application user's input file. Therefore, it is advantageous for the user of any SIERRA application code to be familiar with the components of this hierarchical structure. Details of this structure can be found in Reference [1], but a brief summary is included here.

The hierarchical structure of all SIERRA application codes is shown in Figure 1.1. This represents a layered structure with the foundational block being the core services, or domain, provided by the SIERRA framework itself. Every analysis will have only a single domain. One or more procedures lie directly on top of the domain, each one controlling time stepping and coupling definitions over a segment of an analysis. Each procedure, in turn, supports blocks for regions, each representing the solution of a set of partial differential equations (PDE's) , for solution control which defines the sequence of region and data transfer executions and the size of the time steps for the analysis, and for definitions of the data to be transferred between meshes. This layered structure is reflected in the user's input file through a concept of *scope.*

Commands for the domain are in the outermost scope of the user's input file and communicate directly with core services, such as support for mesh input, libraries for defining material parameters, and blocks for defining parameters for any of the linear solvers that are accessible through the SIERRA system. Entities defined and parameterized in the outermost, domain scope are available to any of the nested scopes. An application can have only a single domain.

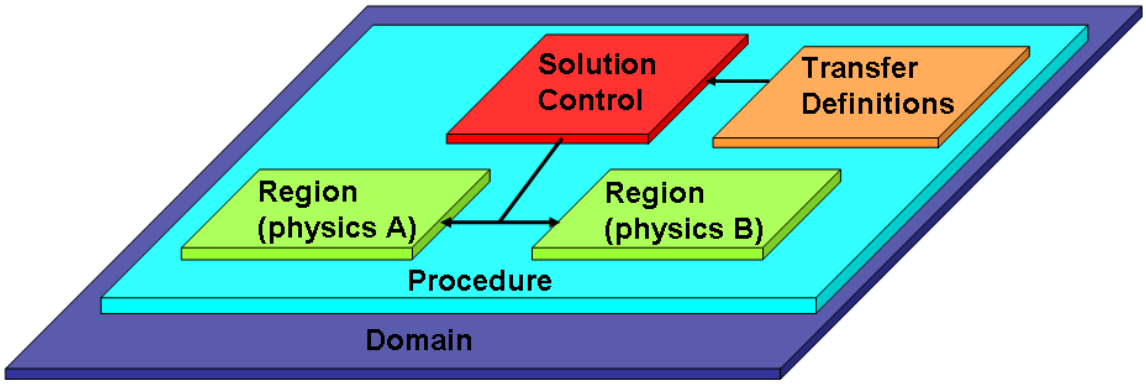


Figure 1.1: Major components of the SIERRA architecture.

The layer lying directly on top of the domain, and thus a nested scope within the domain, is that for procedures. A SIERRA domain may support one or more procedures, a procedure being the SIERRA entity for controlling the time stepping through the solution. Nested within each procedure will be one or more physics modules, called regions. The procedure defines how to procede through a solution in a nested solution control scope and how data is to be transferred between its regions using nested transfer definition scopes.

The SIERRA nomenclature for a physics module is a region. In most cases, the physics module and hence the region represents the solution of a single PDE, although tightly-coupled PDE's may also be advanced through a single region. Each region sequentially executes its physics on its associated mesh for the current time step passed to it by the procedure. Regions include commands for defining mesh input, results output, boundary conditions, and designation of the solver to be used.

Loosely-coupled, multiphysics analyses are accomplished in SIERRA by the sequential execution of two or more regions over over each procedure time step of a simulation. The specific order of the executions is defined by the user in the solution control scope. Each region is responsible for advancing its associated field data to the end of the time step while field data transferred to it from other regions is held constant. For the simplest form of loose coupling between two regions, the first region is executed to advance its field data to the end of the time step and its data is passed to the second region. The specification for the data to be transferred is contained in a transfer definition block. The second region then advances its own field data to the end of the time step, at which time its pertinent field data is passed back to the first region. This process continues to iterate over the same time step until a user-defined convergence is reached in each region. The analysis process then repeats for subsequent time steps until the termination time for the simulation is reached.

Three-dimensional, electromechanical analyses are accomplished in Arpeggio by coupling a region for finite element structural analysis with another region for boundary element analysis of an electrostatic field. The structural analysis may be either quasistatic, in which

case an Adagio region is used, or implicit dynamic, in which case an Andante region is specified. In this document, the nomenclature ”Adagio/Andante” is used to designate that either type of region is appropriate. A BEM region is used for electrostatic analysis.

It should be noted that, while it is possible in Arpeggio to couple a dynamic structural analysis with an electrostatic boundary element analysis, it may not always be appropriate. For this approach to be appropriate, the regime of interest in the structural time response must be large compared to that of the time for any transient response of the electric field. This condition is true for the example problems in this document.

## 1.2 Files Associated with a Coupled Analysis

Two types of input files are required to run Arpeggio. The first type of file is a binary file in the exodusII format that contains the mesh data. (Often called a genesis file, it is an exodusII file format but contains no results information). The second type of input file is an ASCII text file that contains parameter data and the user's commands for executing the analysis. The remainder of this section describes the specific files needed for a coupled electrostatic-structural analysis using Adagio/Andante and BEM regions in Arpeggio. While other types of coupling are possible in Arpeggio, they are outside the scope of this guide.

### 1.2.1 Mesh Data Files

Coupled analyses using Adagio/Andante and BEM regions in Arpeggio requires the user to have two mesh files, one defining a finite element volume discretization of the structures and the other defining a surface mesh over the structures for the boundary integral calculation. The surface mesh for the boundary integral calculation is not required to have coincident nodes with the surface of the volume element mesh. These files may be created with any pre-processing tool that can create an exodusII file.

### 1.2.2 User Input File

An Arpeggio analysis is controlled through a user-generated text file. Conventions for the commands in this file are given in Chapter 2. The input file contains commands for such things as defining load functions, material parameters and boundary conditions, time step control, solver parameters, and the steps to be followed for coupling. Description of the commands needed for coupling are contained in Chapters 3 through 8 . Commands that are specific to Adagio/Andante or BEM regions that do not directly affect coupling are beyond

the scope of this manual, but can be found in the *Adagio/Andante User Reference Manual* [2] and the *BEM Users Manual* [3]. Examples of user input files are included in Chapters 9 and 10.

### 1.2.3 Output Files

Numerical results of an analysis are output in exodusII files that contain the original mesh information as well as any results the user has requested. For a coupled analysis, both structural and electrostatic results files will be generated. Also generated during an analysis is a log file, a text file containing a repeat of the input file, pertinent information regarding the progression of the analysis, and a time summary at the end. If errors are generated either during the parsing of the input file or during the analysis itself, they are written to the log file.

# Chapter 2

# Input File Conventions

## 2.1 File Structure

The hierarchical design of SIERRA codes is reflected in the user's input file by the concept of scope. Each scope in the input file is defined by a command block initiated with a line command starting with the keyword "begin" and terminated with a line command starting with the keyword "end". A command block may contain line commands and/or other command blocks which define scopes nested within it. The major scopes of the input file mimic the layered architecture of SIERRA illustrated in Figure 1.1. A scope nested within another scope is analogous to a layer lying atop another layer.

For example, the outermost scope in the input file corresponds to the SIERRA domain and includes commands and command blocks that are independent of the physics such as definitions of functions, materials, linear solvers, and mesh descriptions. Any nested scopes then have access to these definitions. The domain scope is delineated as

```
BEGIN SIERRA my_problem

END SIERRA my_problem
```

where `BEGIN` and `SIERRA` are the keywords to initiate the command block, and `END` and `SIERRA` are the keywords to end it. The domain scope can also be terminated simply by using

```
END
```

The string `my_problem` is a user-specified name for this domain scope.

The typical Arpeggio input file will have the structure shown below. The major scopes—domain, procedure, solution control, regions, and transfer definitions — are delineated with

input lines for command blocks. Comment lines (lines beginning with #) indicate some of the key scopes that may appear within the major scopes. These major scopes and their nested commands are described in Chapters 3 through 8 of this guide.

```
BEGIN SIERRA <string>some_name
  #
  # All command blocks and command lines in the domain
  # scope appear here. The PROCEDURE command
  # block is the beginning of the next scope.
  #
  # function definitions
  # material descriptions
  # solver definitions
  # mesh file descriptions
  #
  BEGIN PROCEDURE <string>procedure_name
    #
    BEGIN REGION ADAGIO|ANDANTE <string>structural_region_name
      #
      # All command blocks and command lines in the
      # region scope appear here
      #
      # specification for output of results
      # specification for restart
      # specification of boundary and load conditions
      # definition of contact
      # specification of solver
      #
    END [REGION ADAGIO|ANDANTE <string>structural_region_name]
    #
    BEGIN REGION BEM <string>electrostatic_region_name
      #
      # All command blocks and command lines in the
      # region scope appear here
      #
      # specification of solver
      # specification of PDE
      # specification for output of results
      # specification for restart
      # specification of boundary conditions
      # specifications for post-processing
      #
    END [REGION BEM <string>electrostatic_region_name]
    #
    BEGIN SOLUTION CONTROL <string>control_name
```

```
      #  definition of analysis flow
      #  specification of data transfers
      #  time step control.
    END [SOLUTION CONTROL <string>control_name
    #
    BEGIN TRANSFER <string>transfer_name
      #
      #  definition of data transfers between
      #  mesh parts belonging to different regions
      #
    END [TRANSFER <string>transfer_name]
    #
  END [PROCEDURE <string>procedure_name]
  #
END [SIERRA <string>some_name]
```

## 2.2 Conventions for Command Descriptions

This section describes conventions for input commands for a SIERRA application. A number of the individual command lines discussed in this text appear spread over more than one text line simply to accomodate the margins of this document. Therefore, the continuation symbols that would used to continue lines in an actual input file (`/#` and `/$`, Section 2.3.2) are not used here because they are not required in the input file. The description of command lines will clearly indicate all of the keywords, delimiters, and values that constitute a complete command line. As an example, the `DEFINE POINT` command line is

```
DEFINE POINT <string>point_name WITH COORDINATES
     <real>value_1 <real>value_2 <real>value_3
```

When real values are substituted, this line can easily be typed on a single line in the input file. However, if desired, the user could spread it over two lines as follows

```
DEFINE POINT center WITH COORDINATES /#
  10.0 144.0 296.0
```

where the `/#` symbol implies the first line is continued onto the second line.

### 2.2.1 Keywords

Keywords for a command are shown in uppercase letters in this guide. For actual input, any mixture of case can be used.

### 2.2.2 User-Specified Input

Lowercase letters are used to show user-specified input. User-supplied input may be a real number, an integer, a string, or a string list. The type (real, integer, string, string list) description is enclosed by angle brackets, <>, and precedes the user-supplied input. This type string is for description only and is not included in the actual input file. For example, if the description of an input command appears as

```
USE FUNCTION = <string>function_name
```

the user would type

```
USE FUNCTION = my_name
```

where `my_name` is a string the user has chosen to give a function definition.

Valid user input consists of the following:

| | |
|---|---|
| `<integer>` | Integer data is a single integer number. |
| `<real>` | Real data is a single real number. It may be formatted with the usual conventions, such as `1234.56` or `1.23456e+03`. |
| `<string>` | String data is a single string. |
| `<string list>` | A string list consists of multiple strings separated by white space, a comma, or white space combined with a comma. |

### 2.2.3 Optional Input

Square brackets, [ ], represent optional input within a command line. However, any command line that is itself optional in its entirety will be described as such within the text.

### 2.2.4 Default Values

A value enclosed by parentheses, (), appearing after the user input denotes the default value. For example,

```
SCALE FACTOR = <real>scale_factor(1.0)
```

implies the default value for `scale_factor` is 1.0. Any value a user specifies will override

the default.

### 2.2.5 Multiple Options for Values

Quantities separated by the "|" symbol indicate that one and only one of the possible choices must be selected. For example,

```
EXPANSION RADIUS = <string>SPHERICAL|CYLINDRICAL
```

implies that expansion radius must be defined as `SPHERICAL` or `CYLINDRICAL`. This convention is also used for some line command options within a begin/end block. For example,

```
SURFACE = <string>surface_name |
     NODE SET = <string>nodelist_name
```

designates that either a `SURFACE` or a `NODE SET` line must appear in the command block.

Quantities separated by the "/" symbol can appear in any combination, but any one quantity in the sequence can appear only once. For example,

```
COMPONENTS = <string>X/Y/Z
```

implies that components can equal any combination of X, Y, and Z. Any value (X or Y or Z) can appear at most once, and at least one value of X, Y, or Z must appear. Some examples of valid expressions in this case are

```
COMPONENTS = Z
```

```
COMPONENTS = Z X
```

```
COMPONENTS = Y X Z
```

and

```
COMPONENTS = Z Y X
```

## 2.3 Style Guidelines

This section gives recommendations for the overall organization and appearance of an file that will enhance the readability.

### 2.3.1 Comments

Any text on a line that follows a "#" symbol or a "$" symbol is ignored. These symbols can therefore be used to include comments on the end of a command line. If the first nonblank character in a line is a # or $, the entire line is a comment line.

### 2.3.2 Continuation Lines

Any command line can be split across lines in the input file by placing a "\#" pair of characters (or "\$") at the end of the line. The following line is then taken to be a continuation of the preceding line, and together they comprise a single line command. Note that everything after the line-continuation pair of characters is discarded, including the end-of-line.

### 2.3.3 Case

Almost all of the character strings in the input lines are case insensitive. For example, the `BEGIN SIERRA` keywords could appear as

```
BEGIN SIERRA
```

or

```
begin sierra
```

or

```
Begin Sierra
```

A `SIERRA` command block can begin with

```
BEGIN SIERRA BEAM
```

and end with

```
END SIERRA beam
```

Case is important only for file name specifications and for user-defined variable names. Any reference to a file name, such as a mesh input file, must exactly match the case used in the filename itself. Likewise, if variable is defined, all references to that variable at other places in the input file must exactly match the original definition, including case. Instances where case is important will be noted in the text.

### 2.3.4 Commas

Commas in input lines are ignored.

### 2.3.5 Blank Spaces

It is recommended that everything be separated by blank spaces. For example, a command line of the form

```
node set = nodelist_10
```

is preferred over

```
node set= nodelist_10
```

or

```
node set =nodelist_10
```

Both of the above two lines are correct, but it is easier to check the first form (the equal sign surrounded by blank space) in a large input file.

The command parser will accept the line

```
BEGIN SIERRABEAM
```

but it is harder to check this line for the correct spelling of the keywords and the intended domain name than the line

```
BEGIN SIERRA BEAM
```

It is possible to introduce hard-to-detect errors because blank spaces are ignored by the command parser. Suppose the line

```
begin definition for functions my_func
```

is typed rather than the correct form,

```
begin definition for function my_func
```

For the incorrect form of this command line (in which `functions` is used rather than `function`), the parser will generate a string name of

```
s my_func
```

for the function name rather than the expected name of

```
my_func
```

If a function named `my_func` is referenced, the parser will generate an error because the list of function names will include `s my_func` but not `my_func`.

### 2.3.6 General Format of the Command Lines

In the vast majority of cases, command lines have the form

```
keyword = value .
```

### 2.3.7 Delimiters

It is recommended that the `=` sign be used when a delimiter is required, although `is`, and `are` are also valid delimiters for most commands. The `=` sign is recommended as the delimiter of choice because it provides a strong visual cue for separating keywords from values. By relying on the `=` sign as a delimiter, it will be much easier to proofread an input file and to edit with "cut and paste" operations.

### 2.3.8 Order of Commands

There are no requirements for ordering commands within a scope. Both the input sequence

```
BEGIN PRESCRIBED DISPLACEMENT
  NODE SET = nodelist_10
  COMPONENT = X
  FUNCTION = cosine_curve
END PRESCRIBED DISPLACEMENT
```

and the input sequence

```
BEGIN PRESCRIBED DISPLACEMENT
  FUNCTION = cosine_curve
  COMPONENT = X
  NODE SET = nodelist_10
END PRESCRIBED DISPLACEMENT
```

are valid and produce the same result. However, command lines and command blocks must appear in the proper scope.

### 2.3.9 Abbreviated END Specifications

It is possible to terminate a command block without including the keywords that identify the block. For example, a specific instance of the prescribed displacement boundary condition,

```
BEGIN PRESCRIBED DISPLACEMENT
```

can be terminated simply with

```
END
```

as opposed to

```
END PRESCRIBED DISPLACEMENT
```

Both the short termination (`END` only) and the long termination (`END` followed by identification, or name, of the command block) are valid. It is recommended that the long termination be used for any lengthy command block.

### 2.3.10 Indentation

When constructing an input file, it is useful to indent scopes nested inside other scopes. Command lines within a command block should also be indented in relation to the lines defining the command block. This indentation will make scopes more visually apparent in the input file.

## 2.4 Naming Conventions Associated with the Exodus II Database

When the mesh file has an Exodus II format, there are three basic conventions that apply to user input for referencing parts of the mesh data. First, for a mesh file with the Exodus II format, an Exodus II side set is referenced as a surface. In SIERRA, a surface consists of element faces plus all of the nodes and edges associated with these faces. A surface definition can be used not only to select a group of faces but also to select a group of edges or a group of nodes that are associated with those faces. In the case of boundary conditions, a surface definition can be used not only to apply boundary conditions that typically use surface specifications (pressure) but also to those that use nodes(fixed displacement components). For nodal boundary conditions that use the surface specification, all of the nodes associated with the faces on a specific surface will have this boundary condition applied to them. The specification for a surface identifier in the following chapters is `surface_name`. It typically has the form `surface_integerid`, where `integerid` is the integer identi-

fier for the ExodusII side set. If the side set number is 125, the value of `surface_name` would be `surface_125`. It is also possible to generate an alias for the side set and use this for `surface_name`. If `surface_125` is aliased to `outer_skin`, then `surface_name` becomes `outer_skin` in the actual input line.

Second, an Exodus II node set is referenced as a nodelist. A nodelist can be used only for cases where a group of nodes will be used without regard to the faces or elements to which the nodes belong. The specification for a nodelist identifier in the following chapters is `nodelist_name`. It typically has the form `nodelist_integerid`, where `integerid` is the integer identifier for the Exodus II node set. If the node set number is 225, the value of `nodelist_name` would be `nodelist_225`. It is also possible to generate an alias for the node set and use this for `nodelist_name`. If `nodelist_225` is aliased to `inner_skin`, then `nodelist_name` becomes `inner_skin` in the actual input line.

Third, an element block is referenced as a block. The specification for an element block identifier in the following chapters is `block_name`. It typically has the form `block_integerid`, where `integerid` is the integer identifier for the block. If the element block number is 300, the value of `block_name` would be `block_300`. It is also possible to generate an alias for the block and use this for `block_name`. If `block_300` is aliased to `big_chunk`, then `block_name` becomes `big_chunk` in the actual input line.

# Chapter 3

# Domain Scope Commands

## 3.1 Establishing the Domain Scope

The outermost scope in the input file corresponds to the SIERRA domain. It includes commands and command blocks that are independent of the physics such as definitions of functions, material parameters, linear solver parameters, and mesh descriptions. Any nested scopes then have access to these definitions.

The domain scope is delineated as

```
BEGIN SIERRA my_problem

END SIERRA my_problem
```

where `BEGIN` and `SIERRA` are the keywords to initiate this command block, and `END` and `SIERRA` are the keywords to end it. The domain scope can also be terminated simply by using

```
END
```

The string `my_problem` is a user-specified name for this domain scope.

## 3.2 Title

```
TITLE <string list>title_description
```

A `TITLE` command line can be used to include a text description of the analysis. The `title_description` will be copied to any ExodusII results files.

## 3.3 Functions

```
BEGIN DEFINITION FOR FUNCTION <string>function_name
  TYPE = <string>CONSTANT|PIECEWISE LINEAR|
  ZERO PIECEWISE LINEAR|ANALYTIC
  ABSCISSA = <string>abscissa_label
  ORDINATE = <string>ordinate_label
  BEGIN VALUES
    <real>value_1   [<real>value_2
    <real>value_3    <real>value_4
    ...              <real>value_n]
  END [VALUES]
  EVALUATE EXPRESSION = <string>"analytic_expression1;
    analytic_expression2;..."
END [DEFINITION FOR FUNCTION <string>function_name]
```

A number of features are driven by a user-defined description of the dependence of one variable on another. For instance, the prescribed displacement boundary condition used by an Andante region requires the definition of a time-versus-displacement relation. SIERRA provides a general method of defining these relations as functions using the `DEFINITION FOR FUNCTION` command block, as shown above.

There is no limit to the number of functions that can be defined. All function definitions must appear within the domain scope.

A description of the various parts of the `DEFINITION FOR FUNCTION` command block follows:

- The string `function_name` is a user-selected name for the function that is unique to the function definitions within the input file. This name is used to refer to this function in other locations in the input file.
- The `TYPE` command line has four options to define the type of function. The string value can be `CONSTANT`, `PIECEWISE LINEAR`, `ZERO PIECEWISE LINEAR`, or `ANALYTIC`.
- The `ABSCISSA` command line provides a descriptive label for the independent variable ($x$-axis) with the string `abscissa_label`. This command line is optional.
- The `ORDINATE command` line provides a descriptive label for the dependent variable ($y$-axis) with the string `ordinate_label`. This command line is optional.
- The `VALUES` command block consists of the real values `value_1` through `value_n`, which describe the function. This command block must be used if the value on

the TYPE command line is CONSTANT, PIECEWISE LINEAR, or ZERO PIECEWISE LINEAR. For a CONSTANT function, only one value is needed. For a PIECEWISE LINEAR or ZERO PIECEWISE LINEAR function, the values are $(x, y)$ pairs of data that describe the function. The values are nested inside the VALUES command block.

PIECEWISE LINEAR and ZERO PIECEWISE LINEAR behave differently for the case where the abscissa value passed to the function exceeds the abscissa value in the VALUES command block. In the case of a PIECEWISE LINEAR function, for any abscissa value passed to the function that is greater than the last abscissa value in the VALUES command block, the last ordinate value is used for the function value. For example, if a PIECEWISE LINEAR function named my_func describes a time history for a pressure load where the pressure increases from 0 to 50,000 psi from time 0.0 sec to time $1.0 \times 10^{-3}$ sec. The last time specified in the function is $1.0 \times 10^{-3}$. If the final analysis time is $2.0 \times 10^{-3}$ sec, from the time $1.0 \times 10^{-3}$ to the time $2.0 \times 10^{-3}$, the value for this function (my_func) will be 50,000 psi. If my_func is specified as a ZERO PIECEWISE LINEAR function, the value for the function would be zero from time $1.0 \times 10^{-3}$ to the time $2.0 \times 10^{-3}$.

- The EVALUATE EXPRESSION command line consists of one or more user-supplied algebraic expressions. This command line must be used if the value on the TYPE command line is ANALYTIC. See the rules and options for composing algebraic expressions discussed below.

A DEFINITION FOR FUNCTION command block cannot contain both a VALUES command block and an EVALUATE EXPRESSION command line.

***Rules and options for composing algebraic expressions.*** Algebraic expressions for the EVALUATE EXPRESSION command line are written using a C-like format. Each algebraic expression is terminated by a semicolon. The entire set of algebraic expressions, whether a single expression or several, is enclosed in a single set of double quotes.

An expression is evaluated with x as the independent variable. We first provide several simple examples and then list the options available in the algebraic expressions.

Example: Return sin(x) as the value of the function.

```
begin definition for function fred
  type is analytic
  evaluate expression is "sin(x);"
end definition for function fred
```

Example: In this example, the commented out table is equivalent to the evaluated expression.

```
begin definition for function pressure
  type is analytic
  evaluate expression is "x <= 0.0 ? 0.0 : (x < 0.5 ? x*200.0
    : 100.0)"
    #     begin values
    #       0.0      0.0
    #       0.5    100.0
    #       1.0    100.0
    #     end values
end definition for function pressure
```

The following functionality is currently implemented for the expressions:

Operators

```
+ - * / == != > < >= <= ! & | && || ? :
```

Parentheses

```
()
```

Math functions

```
abs(x), absolute value of x
mod(x, y), modulus of x|y
ipart(x), integer part of x
fpart(x), fractional part of x
```

Power functions

```
pow(x, y), x to the y power
pow10(x), x to the 10 power
sqrt(x), square root of x
```

Trigonometric functions

```
acos(x), arccosine of x
asin(x), arcsine of x
atan(x), arctanget of x
atan2(y, x), arctanget of y/x, signs of x and y
  determine quadrant (see atan2 man page)
cos(x), cosine of x
```

```
cosh(x), hyperbolic cosine of x
sin(x), sine of x
sinh(x), hyperbolic sine of x
tan(x), tangent of x
tanh(x), hyperbolic tangent of x
```

Logarithm functions

```
log(x), natural logarithm of x
ln(x), natural logarithm of x
log10(x), the base 10 logarithm of x
exp(x), e to the x power
```

Rounding functions

```
ceil(x), smallest integral value not less than x
floor(x), largest integral value not greater than x
```

Random functions

```
rand(), random number between 0.0 and 1.0, not including 1.0
randomize(), random number between 0.0 and 1.0, not
  including 1.0
srand(x), seeds the random number generator
```

Conversion functions

```
deg(x), converts radians to degrees
rad(x), converts degrees to radians
recttopolr(x, y), magnitude of vector x, y
recttopola(x, y), angle of vector x, y
poltorectx(r, theta), x coordinate of angle theta at
  distance r
poltorecty(r, theta), y coordinate of angle theta at
  distance r
```

***Constants.*** There are two predefined constants that may be used in an expression. These two constants are `e` and `pi`.

```
e = e = 2.7182818284...
pi = π  = 3.1415926535...
```

## 3.4 Axes, Directions, and Points

```
DEFINE POINT <string>point_name WITH COORDINATES
  <real>value_1 <real>value_2 <real>value_3
DEFINE DIRECTION <string>direction_name WITH VECTOR
  <real>value_1 <real>value_2 <real>value_3
DEFINE AXIS <string>axis_name WITH POINT
  <string>point_1 POINT <string>point_2
DEFINE AXIS <string>axis_name WITH POINT
  <string>point DIRECTION <string>direction
```

A number of features require the definition of geometric entities. For instance, the prescribed displacement boundary condition requires a direction definition, and the cylindrical velocity initial condition requires an axis definition. Definition of points, directions, and axes occurs in the domain scope so they may be accessed from any of the regions.

The `DEFINE POINT` command line is used to define a point:

```
DEFINE POINT <string>point_name WITH COORDINATES
  <real>value_1 <real>value_2 <real>value_3
```

*where*

- The string `point_name` is a name for this point. This name must be unique to all other points defined in the input file.
- The real values `value_1`, `value_2`, and `value_3` are the $x$, $y$, and $z$ coordinates of the point.

The `DEFINE DIRECTION` command line is used to define a direction:

```
DEFINE DIRECTION <string>direction_name WITH VECTOR
  <real>value_1 <real>value_2 <real>value_3
```

*where*

- The string `direction_name` is a name for this direction. This name must be unique to all other directions defined in the input file.
- The real values `value_1`, `value_2`, and `value_3` are the $x$, $y$, and $z$ magnitudes of the direction vector.

There are two command lines that can be used to define an axis. The first `DEFINE AXIS` command line uses two points:

```
DEFINE AXIS <string>axis_name WITH POINT
  <string>point_1 POINT <string>point_2
```

*where*

- The string `axis_name` is a name for this axis. This name must be unique to all other axes defined in the input file.
- The strings `point_1` and `point_2` are the names for two points defined in the input file via a `DEFINE POINT` command line.

The second `DEFINE AXIS` command line uses a point and a direction:

```
DEFINE AXIS <string>axis_name WITH POINT
  <string>point DIRECTION <string>direction
```

*where*

- The string `axis_name` is a name for this axis. This name must be unique to all other axes defined in the input file.
- The string `point` is the name of a point defined in the input file via a `DEFINE POINT` command line.
- The string `direction` is the name of a direction defined in the input file via a `DEFINE DIRECTION` command line.

## 3.5 Mesh File Identification

```
BEGIN FINITE ELEMENT MODEL <string> model_name
  DATABASE NAME = <string> file_name
  DATABASE TYPE = <string> file_format
  #
  #  Other mesh specific commands needed for some regions
  #
END [FINITE ELEMENT MODEL <string> model_name]
```

Each physics module (region) of a coupled application operates on a mesh. For electrostatic-structural computations using Arpeggio, two unstructured meshes are required. These are specified in separate `FINITE ELEMENT MODEL` command blocks. Each will be referred to within a region scope by the string `model_name`. The structural part of the analysis requires a finite element volume mesh containing solid elements such as hexagonal bricks, and/or structural elements such as shells and beams that use one- and two-dimensional topologies to represent three-dimensional behavior. The electrostatic analysis requires a surface mesh that encloses one or more three-dimensional volumes.

The inclusion of the keywords `"FINITE ELEMENT"` in the command block denotes that the mesh is an unstructured grid and not a reference to the computational method. Thus, even the boundary integral method in a BEM region uses a finite element model, in this case an unstructured grid that consists only of surface elements.

The string `file_name` in `DATABASE NAME` line specifies the name of the file containing the input mesh. This name is case sensitive and therefore must match the name of the file exactly. The string `file_format` in the `DATABASE TYPE` line specifies the mesh file format. Currently `exodusII` is the only supported option for `file_format`.

Other command blocks nested in the FINITE ELEMENT MODEL block are needed for meshes used by Adagio/Andante regions. Detailed explanations of these commands are included in the Adagio/Andante User Reference Manual [2].

## 3.6 Solver Definitions

Several solvers are available in the SIERRA framework [4] [**?**]. An Adagio/Andante region may use a sparse matrix linear solver for a preconditioner to the core conjugate gradient solver and a BEM region requires a linear solver for solution of its dense nonsymmetric matrix. The parameters for any linear solvers to be used are specified in command blocks in the domain scope. In this manner, multiple regions may access the same solver.

The linear solver most commonly used as a preconditioner in an Adagio analysis is the FETI solver. The command block used to specify the parameters for this solver is

```
BEGIN FETI EQUATION SOLVER <string> feti_solver_name
  #
  #  Command lines to specify solver parameters
  #
END [FETI EQUATION SOLVER <string>feti_solver_name]
```

Detailed descriptions of the command lines to define parameters for the FETI solver are included in the Adagio/Andante User Manual [2].

The BEM region requires specification of a linear solver. Because a matrix formed in the boundary integral method is a dense matrix that may be nonsymmetric, care must be exercised in the choice of solver. The most commonly used BEM solver is from the Trilinos package [6].

```
BEGIN TRILINOS EQUATION SOLVER <string> dense_solver_name
  SOLUTION METHOD = BICGSTAB
  PRECONDITIONING METHOD = NONE
  MAXIMUM ITERATIONS = <Integer>iterations
  RESIDUAL NORM TOLERANCE = <Real> tolerance
END [TRILINOS EQUATION SOLVER <string> dense_solver_name]
```

The stabilized biconjugate solver specified by `SOLUTION METHOD = BICGSTAB` is appropriate for the dense, nonsymmetric matrices that result from the electrostatic PDE's. Currently, there are no preconditioning methods appropriate for this solver that are supported through the SIERRA solver services. The command lines `MAXIMUM ITERATIONS` and `RESIDUAL NORM TOLERANCE` are used to control the convergence behavior of the iterative solver.

## 3.7 Procedure Scope

```
BEGIN PROCEDURE <string>procedure_name
  #
  #  Commands to define simulation control
  #
  #  Commands to define regions
  #
  #  Commands to define data transfers between regions
  #
END [PROCEDURE <string>procedure_name]
```

The procedure command block is nested within the domain scope. The procedure scope is established with a `BEGIN PROCEDURE` command and terminated with `END [PROCEDURE]`. The internal structure of the procedure command block is described in subsequent chapters.

# Chapter 4

# Procedure Scope Commands

## 4.1 Structure of the Procedure Command Block

```
BEGIN PROCEDURE <string>procedure_name
  #
  #  Adagio/Andante Region Command Block
  #
  #  BEM Region Command Block
  #
  #  Solution Control Command Block
  #
  #  Transfer Command Blocks
  #
END [PROCEDURE <string>procedure_name]
```

A SIERRA procedure command block is used to define which physics modules (regions) are needed and how they are to be coupled. The procedure block consists of nested command blocks that define the regions, the solution control, and the transfer of data between regions. For the purpose of this guide, the regions that will be coupled are an Adagio/Andante region for structural analysis and a BEM region for electrostatic analysis.

## 4.2 Adagio or Andante Region

```
BEGIN ADAGIO REGION <string>adagio_region_name
  #
  #  Commands to define quasistatic structural analysis
  #
```

```
END [ADAGIO REGION <string>adagio_region_name]


BEGIN ANDANTE REGION <string>andante_region_name
  #
  #  Commands to define implicit dynamic structural analysis
  #
END [ANDANTE REGION <string>andante_region_name]
```

Three-dimensional finite element structural analysis is specified in Arpeggio using one of two types of regions. An Adagio region is used for quasistatic analyses while an Andante region is appropriate for implicit dynamics analyses. Both of these regions are further described in the Adagio/Andante User Reference Manual [2]. Most command blocks and command lines contained within the Adagio/Andante region scope pertain only to the structural analysis and do not affect coupling. Therefore the descriptions these commands will not be repeated in this document. Commands that are directly related to coupling are described in Chapter 5.

## 4.3 BEM Region

```
BEGIN BEM REGION <string>bem_region_name
  #
  #  Commands to define electrostatic analysis
  #
END [BEM REGION <string>bem_region_name]
```

Electrostatic computations using a boundary integral method are defined within Arpeggio using a BEM region. Use of a BEM region is further described in the BEM Users' Guide [3]. Many command blocks and command lines contained within the BEM region scope do not affect coupling and therefore descriptions will not be repeated in this document. Commands that are either directly related to coupling or specific to electrostatic computations are described in this document in Chapter 6.

## 4.4 Solution Control

```
BEGIN SOLUTION CONTROL <string>solution_control_name
  #
  #  Commands to define sequence of region and transfer
  #  executions
```

```
  #
  #  Commands to define time stepping
  #
END [SOLUTION CONTROL <string>solution_control_name]
```

A solution control block in the procedure scope is used to define the flow through a coupled simulation. The solution control block defines the time stepping to be used as well as the execution order for regions and transfer of data between regions. Detailed descriptions of the command blocks and command lines nested within a solution control block are included in Chapter 7.

## 4.5 Transfer

```
BEGIN TRANSFER <string>transfer_name
  #
  #  Commands to define data transfer between regions
  #
END [TRANSFER <string>transfer_name]
```

Transfer command blocks are used to define which data is to be transferred between regions during a coupled simulation. A transfer block must be defined for each field to be transferred from the mesh of one region to the mesh of the other region. Transfer block can also be used to define new fields within a region. Detailed descriptions of the command lines used to transfer of data between regions are included in Chapter 8.

# Chapter 5

# Adagio/Andante Region Scope Commands

## 5.1 Structure of the Adagio/Andante Region Command Block

```
BEGIN ADAGIO/ANDANTE REGION <string>structural_region_name
  #
  # Mesh definition line command
  #
  # Options line command (LumpedMass for Andante)
  #
  # Kinematic boundary conditions blocks
  #
  # Load definitions (including transferred fields) blocks
  #
  # Solver definition block
  #
  # Definition of results output block
  #
END [ADAGIO/ANDANTE REGION <string>structural_region_name]
```

The purpose of the Adagio/Andante Region block command is to define the parameters for the structural component of the analysis. The structural analysis is based on nonlinear, three-dimensional finite element methods. Either an Adagio region is specified for quasistatic analysis or an Andante region for implicit dynamic analysis. As shown below, the region block includes a line command for specification of the mesh and nested block commands for definition of both kinematic boundary conditions and loads, definition of

the nonlinear solver, and specification of the variables to be output to the results database. In both cases, the mesh specified must define the volume of a three-dimensional structure. In addition for an Andante analysis, an `options=lumpedMass` line command must be included.

Each command line and nested command block needed in an Adagio/Andante region block, as well as currently supported element types, is discussed in detail in the Adagion/Andante User Reference Manual [2]. However, coupled problems require a special use of the pressure load block that is explained here.

## 5.2 Coupled Pressure Load Definition

```
BEGIN PRESSURE
  SURFACE = <string>pressure_surface_id
  FIELD VARIABLE = <string>transfer_pressure_name
END [PRESSURE]
```

In a coupled electrostatic-structural analysis, the boundary element code BEM is used to compute electrostatic pressures. This pressure field must be transferred to the structural mesh. Details of the transfer of the pressure field are defined in a transfer (Chapter 8. The Adagio/Andante region must contain a pressure load block command to use the received pressure field as a load on the structural mesh.

The `SURFACE = <string>pressure_surface_id` line defines the surface (sideset in exodusII nomenclature) over which the pressure is to be applied, and `pressure_ surface_id` must correspond to a sideset identifier in the exodusII input mesh file following the format described in Section 2.4. This surface must also be a subset of the surface defined in a transfer command block that specifies which surfaces in the structural mesh are to receive electrostatic pressures sent from the electrostatic analysis.

The `FIELD VARIABLE = <string>transfer_pressure_name` tells the Adagio/Andante region which pressure variable to use in the load computation. There must be a transfer command block in the procedure scope that defines this variable name.

# Chapter 6

# BEM Region Scope Commands

## 6.1 Structure of the BEM Region Command Block

```
BEGIN BEM REGION <string>bem_region_name
  #
  # Mesh definition line
  #
  # Solver specification line
  #
  # Deform geometry line
  #
  # Boundary element zone definition block
  #
  # Post-processing commands block
  #
  # Definition of results output block
  #
END [BEM REGION <string>bem_region_name]
```

For coupled electrostatic-structural analyses using Arpeggio, electrostatic pressures are computed using the boundary integral equation method by specifying a BEM region. The BEM region contains line commands for specifying the surface mesh, the solver, and that a deformed geometry is to be used. BEM is more general than an Adagio region because several different types of partial differential equations (PDE's) are supported. The details of the PDE to be used are defined within a nested boundary element zone command block. In addition, BEM contains a post-processing command block to define derived variables that can then be specified along with intrinsic variables in a results output command block and in transfer blocks. A general description of the BEM region along with the theory can

be found in the BEM User's Manual [3]. The specific commands needed to perform computation of electrostatic pressures on a mesh that has been deformed using the structural displacements are discussed in the following sections of this document.

## 6.2 Mesh Specification

```
USE FINITE ELEMENT MODEL <string> model_name
```

In this command line, `model_name` refers to the name on a `FINITE ELEMENT MODEL` command block that must appear in the domain scope (Chapter 3). The keywords `FINITE ELEMENT` here refer to the fact that the mesh is an unstructured grid of the surfaces enveloping any bodies to be included in the electrostatic analysis.

## 6.3 Solver Designation

```
USE DENSE SOLVER = AZTEC00 WITH <string>solver_parameters_name
```

This command specifies the use of the AztecOO solver that is a component of the Trilinos solver package. The parameters for the solver must be included in a `BEGIN TRILINOS EQUATION SOLVER <string>solver_parameters_name` command block in the domain scope as described in Chapter 3.

## 6.4 Deformed Geometry Specification

```
DEFORM GEOMETRY USING TRANSFER
    <string>transfer_displacements_name
```

This commands specifies that the electrostatic computations will be computed on a mesh deformed using the displacement field variable named `transfer_displacements_name`. This `transfer_displacements_name` is the displacement field received by the BEM region from the surface of the structural mesh as defined in a `TRANSFER` command block in the domain scope. Transfer blocks are described in Chapter 8.

## 6.5 Boundary Element Zone

```
BEGIN BE ZONE <string>be_zone_name
  PDE = LAPLACE
  MATERIAL PARAMETER FOR LAPLACE EQUATION PERMITTIVITY
      = <real>permittivity_value
  SOLUTION DOMAIN = EXTERNAL
  BC QUASISTATIC UNIFORM DIRICHLET AT
      BLOCK_<integer>exodus_block_id
      <string>potential_variable_name
      USING FUNCTION <string>function_name
      [SCALED BY <real>scale_factor]
  DIRECT FORMULATION FOR SCALAR <string>potential_variable_name
      AND <string>derivative_variable_name WITH
      CONTINUOUS|DISCONTINUOUS LINEAR ELEMENTS
END [BE ZONE <string>be_zone_name]
```

A BEM region can be used to specify a number of different (PDE's) that are to be solved over the surface mesh. The type of PDE, its physical parameters, its solution domain and the boundary conditions on the domain are defined within the `BE ZONE` command block. This block also contains a command to define the type of boundary integral formulation to be used along with the variable names and order of interpolation to be used for solution of the specified PDE on this zone.

### 6.5.1 Partial Differential Equation

```
PDE = LAPLACE
```

For electrostatics, the appropriate PDE is a Laplace equation.

### 6.5.2 Direct Formulation

```
DIRECT FORMULATION FOR SCALAR <string>potential_variable_name
    AND <string>derivative_variable_name WITH
    DISCONTINUOUS|CONTINUOUS LINEAR ELEMENTS
```

This command specifies that a direct boundary integral formulation is to be used for derivation of the discretized PDE. For electrostatic problems, the two unknowns in the equation are a scalar potential and the magnitude of the gradient of the potential normal to the boundary. For a BE ZONE, the user is required to supply the names for these variables denoted

above by potential_variable_name and derivative_variable_name. For electrostatic analyses, the potential is the voltage and the magnitude of the derivative is the normal of the electric field at the boundary. If CONTINUOUS is specified, the variables are continuous across elements. At edges and corners, therefore, a single value of flux is computed based on an average of the normals to the surfaces that contain that edge or corner. For a discontinuous formulation, the variables are computed at points inside each element and are therefore discontinuous between elements. Finally, the type of element interpolation is specified by the keyword LINEAR. Currently only linear element interpolations are supported.

### 6.5.3 Solution Domain

```
SOLUTION DOMAIN = EXTERNAL
```

In general, the potential for the Laplace equation can be solved internal or external to the boundaries of the meshed parts. For MEMS problems, the domain of interest for computing the electric field is the airspace (assumed infinite) that envelops the MEMS parts and the boundaries of this airspace volume. As illustrated in Figure 6.1, the domain is external to the bodies enclosed by the surface mesh. The boundary consists of the surfaces of bodies in the domain and a far-field imaginary (not meshed) boundary at infinity. In effect, any enclosed bodies are ”holes” in the domain, hence the normals to the surfaces of an enclosed body must point toward the center of the body to be consistent with an outward normal to the external domain.

### 6.5.4 Permittivity Parameter

```
MATERIAL PARAMETER FOR LAPLACE EQUATION PERMITTIVITY
    = <real>permittivity_value
```

This command specifies the electrical permittivity value for the material in the domain being evaluated. For MEMS problems in which the solution domain is external, this would be the permittivity of the airspace enveloping the part.

### 6.5.5 Dirichlet Boundary Condition

```
BC QUASISTATIC UNIFORM DIRICHLET AT
    BLOCK_<integer>exodus_block_id
    <string>potential_variable_name
```

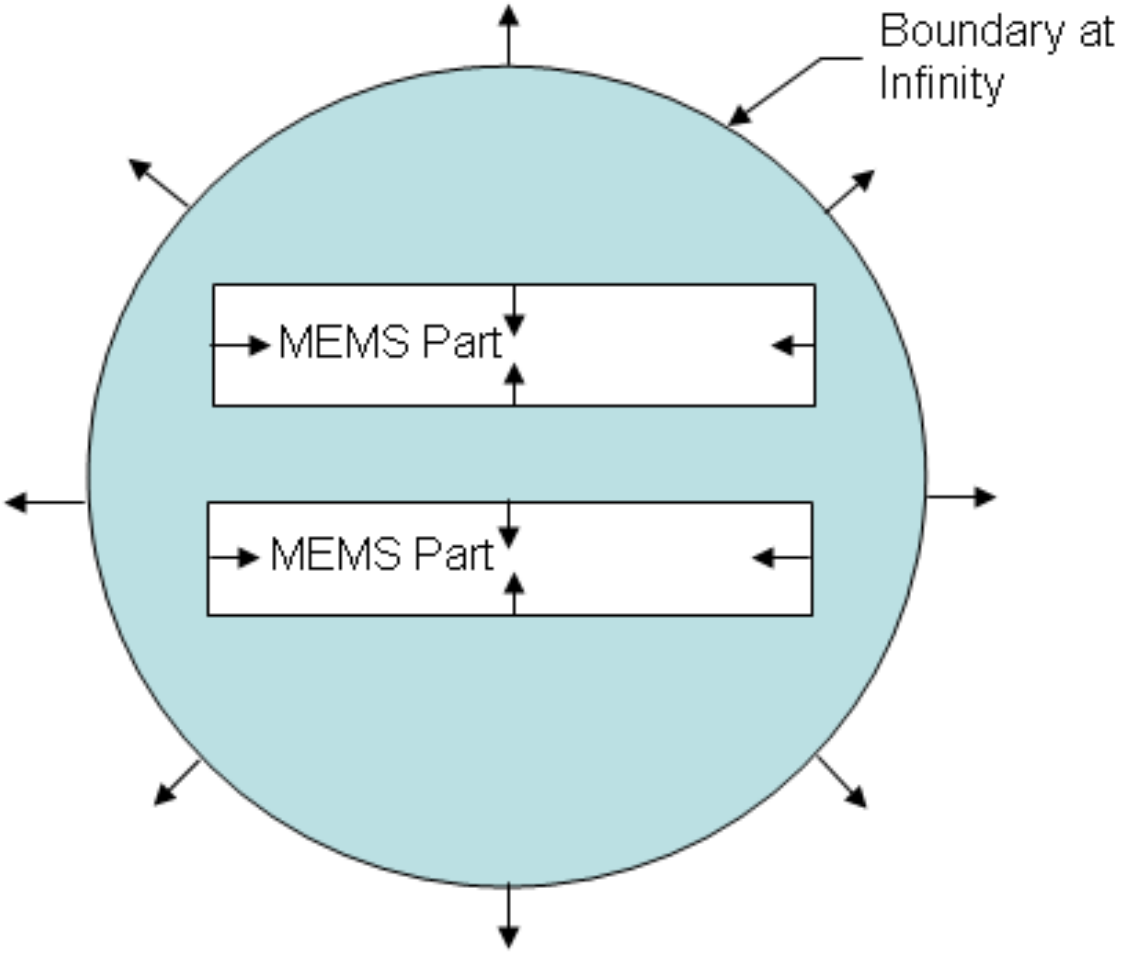


Figure 6.1: External domain for boundary integral equation formulation.

```
USING FUNCTION <string>function_name
[SCALED BY <real>scale_factor]
```

This command specifies the boundary condition on elements of the exodus element block with id `exodus_block_id`. A `BC QUASISTATIC` line is required for each element block in the exodus file. The union of the element blocks must contain every exterior surface of every part and a boundary condition must be specified on every element block.

In general, the boundary condition at any point on the domain boundary in the boundary integral formulation of Laplace's equation is the specification of either the potential (dirichlet condition) or the flux (Neumann condition). In MEMS applications, the voltage is the potential specified on the surfaces of the MEMS parts, hence a dirichlet condition is required. The integral over the far-field boundary falls out of the formulation so no boundary conditions are necessary at infinity.

The keyword QUASISTATIC indicates that the boundary condition is a function of time with the actual dependence on time defined by a FUNCTION function_name command block in the domain scope. The boundary condition magnitude specified in the FUNCTION function_name can be easily modified by including the optional keywords SCALED BY followed by a scale_factor. In this case, every magnitude in FUNCTION function_name will be multiplied by scale_factor.

## 6.6 Post Processing

```
BEGIN POST PROCESS <string>post_process_name
  PROCESS EQUATION LAPLACE ON ZONE <string>be_zone_name
  EVALUATE ELECTRIC_PRESSURE AS <string>bem_e_press_name
  COMPUTE ON SURFACE_<string>exodus_surface_id FACES
END [POST PROCESS <string>post_process_name]
```

This command block specifies further computations to be performed on the variables of the Laplace equation to obtain derived quantities of use to the analyst. For coupled electrostatic pressure, the normal component of the electric field on the boundary is used to compute pressure on the meshed surfaces of the parts. This pressure field with user-designated name bem_e_press_name will be the field variable transferred to the Adagio/Andante region. Because the structural region requires pressure as a face variable, bem_e_press_name must be computed as a face variable.

### 6.6.1 Process Laplace Equation

```
PROCESS EQUATION LAPLACE ON ZONE <string>be_zone_name
```

This line command specifies that the post processing for this command block is to use the Laplace equation solution defined in the BE ZONE specified with the name be_zone_name.

### 6.6.2 Evaluate Electric Pressure

```
EVALUATE ELECTRIC_PRESSURE AS <string>bem_e_press_name
```

Use of the keyword ELECTRIC_PRESSURE in this line command specifies that the electric pressure is to be computed from the solution variables. This variable is then available to the user for transferring to the structural mesh and for results output by referencing bem_e_press_name in the appropriate command blocks.

### 6.6.3 Surface Faces Computation

```
COMPUTE ON SURFACE_<integer>exodus_surface_id FACES
```

This line command specifies that the `ELECTRIC_PRESSURE` denoted in the previous command line is to be computed on the faces of the exodus surface with id `exodus_surface_id`. Because `bem_e_press_name` is the field that will be transferred to the Adagio/Andante region for use in a `PRESSURE` command block, it must be computed as a face variable as designated in this command line with the `FACES` keyword. Use one `COMPUTE ON` command line for each exodus surface that will be needed to transfer the electrical pressure to the structural region.

# Chapter 7

# Solution Control Scope Commands

## 7.1 Structure of the Solution Control Command Block

```
BEGIN SOLUTION CONTROL <string>solution_control_name

  USE SYSTEM <string>system_name

  BEGIN INITIALIZE <string>initialize_name
  END [INITIALIZE <string>initialize_name]

  BEGIN SYSTEM <string>system_name

    USE INITIALIZE <string>initialize_name

    SIMULATION TERMINATION TIME = <real>sim_term_time_value

    BEGIN TRANSIENT <string>transient_name

      BEGIN NONLINEAR <string>nonlinear_sequence_name
        TRANSFER <string>transfer_name
        ADVANCE <string>region_name
      END [NONLINEAR <string>nonlinear_sequence_name

    END [TRANSIENT <string>transient_name]

  END [SYSTEM <string>system_name]

  BEGIN PARAMETERS FOR TRANSIENT <string>transient_name
```

```
    START TIME = <real>transient_start_time_value
    TERMINATION TIME = <real>transient_termination_time_value

    BEGIN PARAMETERS FOR ADAGIO/ANDANTE
        REGION <string>adagio/andante_region_name
      #
      #  Definition of time stepping and time integration
      #  parameters for this time period.
      #
    END [PARAMETERS FOR ADAGIO/ANDANTE
        REGION <string>adagio/andante_region_name]

    BEGIN PARAMETERS FOR BEM REGION <string>bem_region_name
      #
      #  Definition of time stepping parameters
      #  for this time period.
      #
    END [PARAMETERS FOR BEM REGION <string>bem_region_name]

  END [PARAMETERS FOR TRANSIENT <string>transient_name]

  BEGIN PARAMETERS FOR NONLINEAR <string>nonlinear_sequence_name
    CONVERGED WHEN "(logical statement for convergence)"
  END [PARAMETERS FOR NONLINEAR <string>nonlinear_sequence_name]

END [SOLUTION CONTROL <string>solution_control_name]
```

The `SOLUTION CONTROL` command block controls the flow of the coupled code. It contains nested blocks for defining the sequence of region and transfer executions and time stepping for each time period of the analysis.

## 7.2 Use System

```
USE SYSTEM <string>system_name
```

This command specifies that the system named `system_name` is to be used for this analysis, where `system_name` must be the name of a `SYSTEM` command block somewhere in the `SOLUTION CONTROL` scope. The user may define multiple `SYSTEM` command blocks in the input file, but only one is used for any particular analysis.

## 7.3 Initialize

```
BEGIN INITIALIZE <string>initialize _name
  ADVANCE <string>structural_region_name
  ADVANCE <string>bem_region_name
END [INITIALIAZE <string>initialize_name]
```

The `INITIALIZE` command block defines steps that the coupled code must take to initialize the individual regions. It will be used in a `SYSTEM` command block.

## 7.4 System

```
BEGIN SYSTEM <string>system_name
  USE INITIALIZE <string>initialize_name

  BEGIN TRANSIENT <string>transient_name
    #
    # Commands to define simulation sequence loop for
    # each time step.
    #
  END [TRANSIENT <string>transient_name]

  SIMULATION TERMINATION TIME = <real>simulation_term_time
END [SYSTEM <string>system_name]
```

The `SYSTEM` command block gives the simulation sequence for the coupled analysis. This block contains a reference to an `INITIALIZE` block that defines the initialization sequence, one or more `TRANSIENT` command blocks that define the simulation sequence and time steps for individual time period(s), and a line command specifying the termination time for the entire simulation.

Start times, termination times, and time step size for each `TRANSIENT` command block must be defined in separate `PARAMETERS FOR TRANSIENT` blocks (Section 7.5) having the same `transient_name`. If multiple `TRANSIENT` blocks are used, they cannot have overlapping time periods and together must cover the entire simulation period without gaps in the time.

### 7.4.1 Use Initialize

```
USE INITIALIZE <string>initialize_name
```

This line command specifies which INITIALIZE command block is to be used for this SYSTEM. The name initialize_name must correspond to a name on an INITIALIZE command block somewhere in the SYSTEM scope.

### 7.4.2 Transient

```
BEGIN TRANSIENT <string>transient_name

  BEGIN NONLINEAR <string>convergence_name
    TRANSFER <string>bem_displacement_transfer
    ADVANCE <string>bem_region_name
    TRANSFER <string>bem_to_structural_transfer_name
    ADVANCE <string>structural_region_name
    TRANSFER <string>structural_to_bem_transfer_name
  END [NONLINEAR <string>convergence_name]

  TRANSFER <string>bem_pressure_transfer

END [TRANSIENT <string>transient_name]
```

The total simulation time may be divided into a sequence of time periods, each defined by a TRANSIENT command block giving the solution sequence for a time period. The actual values to be used for the beginning time, termination time, and time steps for the time period are given in a related PARAMETERS FOR TRANSIENT transient_name command block (Section 7.5) that must exist elsewhere in the SOLUTION CONTROL scope. Unlike most commands in the input file, the commands within the nested NONLINEAR block and the subsequent TRANSFER command line must be given in the correct sequence because this order defines the sequence of physics and data transfers to be executed in the coupled simulation.

The contents of the NONLINEAR block and the subsequent TRANSFER line shown here should be used exactly in this order for a coupled electrostatic-structural analysis. The user-defined string following each ADVANCE keyword must be the name of either the structural region or the BEM region as in the order indicated. The region name must match that given on a ADAGIO|ANDANTE REGION or a BEM REGION command block in the procedure scope. The user-defined string following a TRANSFER keyword must match the name of a TRANSFER command block of the correct type in the procedure scope. The four types of TRANSFER command blocks needed for an electrostatic-structural coupled analysis are described in Chapter 8.

The line commands included within a NONLINEAR command block define the order that physics and transfers are executed within one iteration of a time step. Convergence is

checked at the end of each iteration using the criteria established in a PARAMETERS FOR NONLINEAR converge_name block (Section 7.6) that must appear in the SOLUTION CONTROL scope. The TRANSFER bem_pressure_transfer occurs after a time step has converged before the first iteration of the next time step commences.

### 7.4.3 Simulation Termination Time

```
SIMULATION TERMINATION TIME = <real>simulation_term_time
```

This command line gives the termination time for the entire analysis defined in the SYSTEM command block. The value for simulation_term_time must be less than or equal to the termination time of the last time period defined by a TRANSIENT block.

## 7.5 Parameters for Transient

```
BEGIN PARAMETERS FOR TRANSIENT <string>transient_name
  START TIME = <real>period_start_time
  TERMINATION TIME = <real>period_termination_time

  BEGIN PARAMETERS FOR ADAGIO|ANDANTE REGION
      <string>structural_region_name
    #
    #  Commands to define time step size and
    #  and time integration parameters
    #
  END [PARAMETERS FOR ADAGIO|ANDANTE REGION
      <string>structural_region_name]

  BEGIN PARAMETERS FOR BEM REGION
      <string>bem_region_name
    #
    #  Command to define time step size
    #
  END [PARAMETERS FOR ADAGIO|ANDANTE REGION
      <string>bem_region_name]

END [PARAMETERS FOR TRANSIENT <string>transient_name]
```

Each TRANSIENT block used to define the analysis sequence must have a corresponding PARAMETERS FOR TRANSIENT <string>transient_name command block that contains line commands for defining the start and termination times and nested command

blocks for the time step size to be used by each region. These times will be applied to the TRANSIENT block of the same name that is defined in the SYSTEM scope. Subcycling over one of the regions is not currently supported, so the time step size defined in the PARAMETERS FOR BEM REGION must match the size defined in the PARAMETERS FOR ADAGIO|ANDANTE REGION block.

### 7.5.1 Start Time

```
START TIME = <real>period_start_time
```

This command defines the start time for this time period.

### 7.5.2 Termination Time

```
TERMINATION TIME = <real>period_termination_time
```

This command defines the termination time for this time period.

### 7.5.3 Time Parameters for Adagio Region

```
BEGIN PARAMETERS FOR ADAGIO REGION
    <string>structural_region_name

  TIME INCREMENT = <real>time_step_size
  # OR #
  NUMBER OF STEPS = <integer>number_of_steps

END [PARAMETERS FOR ADAGIO REGION
    <string>structural_region_name]
```

This command block defines the time step size to be used for an Adagio region when performing a quasistatic structural analysis. The user may specify either the time increment or the number of steps over the time period.

### 7.5.4 Time Parameters for Andante Region

```
BEGIN PARAMETERS FOR ANDANTE REGION
```

```
    <string>structural_region_name

  TIME INCREMENT = <real>time_step_size
  # OR #
  NUMBER OF STEPS = <integer>number_of_steps

  TIME INTEGRATION RULE = HHT
  HHT INTEGRATION PARAMETER ALPHA = <real>alpha
  HHT INTEGRATION PARAMETER BETA = <real>beta
  HHT INTEGRATION PARAMETRE GAMMA = <real>gamma

END [PARAMETERS FOR ANDANTE REGION
    <string>structural_region_name]
```

This command block is used to define the structural region time step size and integration parameters for an implicit dynamic simulation in which the structural region is an Andante region. The user specifies either the actual time increment to be used or the number of steps into which the period is to be divided.

The time integration scheme currently supported is that derived Hilber, Hughes, and Taylor (HHT) which has three parameters: alpha, beta, and gamma. The default values assure unconditional stability and no dissipation. Although some numerical dissipation can be introduced using different values, caution must be exercised as there are limits on these values and how they interact as described in Reference [7].

### 7.5.5 Time Parameters for BEM Region

```
BEGIN PARAMETERS FOR BEM REGION
    <string>bem_region_name

  TIME INCREMENT = <real>time_step_size
  # OR #
  NUMBER OF STEPS = <integer>number_of_steps

END [PARAMETERS FOR BEM REGION
    <string>bem_region_name]
```

This command block sets the time step for the electrostatic region. The line command used for the electrostatic region must match that used for the structural region because region subcycling is currently not supported.

## 7.6 Parameters for Nonlinear

```
BEGIN PARAMETERS FOR NONLINEAR
    <string>nonlinear_sequence_name

    CONVERGED WHEN "(logical statement for convergence)"

END [PARAMETERS FOR NONLINEAR
    <string>nonlinear_sequence_name]
```

This command block defines the logic for determining when the loop defined in the `NONLINEAR` command block with the name `nonlinear_sequence_name` has reached convergence. When the convergence criteria defined by the user in the `CONVERGED WHEN` line command is met, the analysis will move on to the next time step of the analysis.

The syntax of the `CONVERGED WHEN` line command follows the normal C++ coding rules for conditional statements. Both the quote marks and the parentheses are required and words in the enclosed statement are case sensitive. The specific form needed for a coupled electrostatic-structural analysis is shown below.

```
CONVERGED WHEN "(  1 < CURRENT_STEP
    && <string>structural_region_name.norm(0.0)
                  < <real>structural_tolerance
    && <string>bem_region_name.norm(0.0) < <real>bem_tolerance
    || <integer>max_iteration < CURRENT_STEP  )"
```

For this form of the statement, the structural norm used is that of the velocity field. Convergence is computed by comparing the ratio of the difference of the velocity norm from the previous iteration and the current iteration divided by the current velocity norm to the user-specified `structural_tolerance`. The norm of the electrostatic solution field is used to compute a ratio for the BEM region in a similar manner. If the computed ratio for both regions is smaller than the user-specified tolerance or the number of iterations is greater than `max_iteration` the solution will proceed to the next time step.

# Chapter 8

# Transfer Scope Commands

## 8.1 Structure of the Transfer Command Block

```
BEGIN TRANSFER <string>transfer_name
  #
  #  Interpolation type
  #
  #  Send and receive surfaces
  #
  #  Name and state of send and receive fields
  #
  #  Search parameters for interpolation
  #
END [TRANSFER <string>transfer_name]
```

Transfers of data between the meshes associated with different regions or from one field variable to another on the same mesh can be requested. Details of each type of transfer are defined within a `TRANSFER` block. The transfer is then invoked by a reference to `transfer_name` in a `TRANSIENT` command block in the solution control scope.

For a loosely coupled electrostatic-structural analysis using arpeggio, four types of `TRANSFER` blocks are required. Two are inter-region transfers that interpolate data from the surface of the structural mesh to the boundary integral mesh and vice versa. The other two are transfers of data within the mesh of the BEM region itself. These latter two transfer blocks are required to get BEM displacement data into the correct state for a time update and to define a convergence variable. In this chapter, each of these four types of transfers is described in the specific form it is used for a BEM/Adagio coupled analysis. Here, ”adagio” is used to imply either an adagio region or an andante region.

In general, keywords and user-chosen names in the input file are case-insensitive. However, in any command line where a field variable is defined, the choice of case is up to the user, but it will be case sensitive. Therefore, any other reference to this variable in the input file must be consistent in case with the original definition.

## 8.2 Adagio/Andante to BEM Transfer

```
BEGIN TRANSFER <string>adagio_to_bem_transfer_name
  INTERPOLATE SURFACE NODES FROM <string>adagio_region_name
      TO <string>bem_region_name
  SEARCH SURFACE GAP TOLERANCE = <real>surface_tolerance
  SEARCH GEOMETRIC TOLERANCE = <real>normal_tolerance
  SEND BLOCK SURFACE_<integer>adagio_surface_exodus_id
      TO SURFACE_<integer>bem_surface_exodus_id
  SEND FIELD DISPLACEMENT STATE NEW
      TO <string>transfer_displacement_name STATE NEW
END [TRANSFER <string>adagio_to_bem_transfer_name]
```

The first type of transfer moves the adagio displacement field data at the end of the current iteration to BEM. In the `BEGIN TRANFER` command line, `adagio_to_bem_transfer_name` is a user-defined string. It must match the string in a `TRANSFER` command line used in a `NONLINEAR` command block in the solution control scope.

### 8.2.1 Surface Interpolation Designation

```
INTERPOLATE SURFACE NODES FROM <string>adagio_region_name
    TO <string>bem_region_name
```

This command line specifies that data from the surface nodes of the structural region are to be transferred to the electrostatic region mesh via a surface interpolation. Because the structural volume mesh and the electrostatic surface mesh are not congruent (even if there is a one-to-one coincidences of all surface mesh nodes), an interpolation must be performed in order to correctly transfer the data between meshes. Furthermore, for an interpolative transfer, there is no requirement that the surface nodes of the two meshes are coincident. The strings `adagio_region_name` and `bem_region_name` must match the names assigned on the `ADAGIO/ANDANTE REGION` and the `BEM REGION` command blocks, respectively, in the procedure scope.

### 8.2.2 Send and Receive Surface Definitions

```
SEND BLOCK SURFACE_<integer>adagio_surface_exodus_id1
    [SURFACE_adagio_surface_exodus_id2 ...]
    TO SURFACE_<integer>bem_surface_exodus_id1 [
    SURFACE_<integer>bem_surface_exodus_id2 ...]
```

The `SEND BLOCK` line command defines the surfaces that contain the nodes to be used in the transfer. In this command, `adagio_surface_exodus_id's` must be sidesets defined in the mesh for the `adagio_region_name` and the `bem_surface_exodus_id's` must match sidesets on the mesh for the `bem_region_name`. As indicated here, lists of surfaces may be used within a single `TRANSFER` command block, or multiple `TRANSFER` command blocks may be used. In either case, in any particular `TRANSFER` command block, the surface defined by the union of the structural sidesets must include topologically the surface defined by the list of BEM sidesets within a numerical tolerance. In other words, every node in the list of BEM sidesets must lie within the surface defined by the union of the Adagio sidesets.

### 8.2.3 Search Tolerance on Surface Elements

```
SEARCH SURFACE GAP TOLERANCE = <real>surface_tolerance
```

Interpolation of data between different meshes is performed by finding the parametric location of each node in the receive surface within an element of the send surface. For nodes that may lie near the edge of an element, the `surface_tolerance` specified in this command line is used to determine whether the point actually lies within the element. The value for `surface_tolerance` is a distance in the appropriate mesh units that is some fraction of a typical element edge length.

### 8.2.4 Search Tolerance for Normals to Surface

```
SEARCH GEOMETRIC TOLERANCE = <real>normal_tolerance
```

For a surface to surface transfer, `normal_tolerance` establishes a small distance normal to the sending surface for which a node of the receive surface is considered to actually lie within that send surface element. In other words, this `normal_tolerance` allows a small gap to exist between the meshed surfaces in the two regions that represent the same physical surface.

### 8.2.5 Send and Receive Field Definitions

```
SEND FIELD DISPLACEMENT STATE NEW
    TO <string>transfer_displacements_name STATE NEW
```

In this command, `DISPLACEMENT` is the name for the displacement field registered internally in the Adagio region code and thus appears as a keyword rather than a user-chosen name. On the contrary, the name for the field of transferred displacements in the BEM region is left to the user to choose and the string `transfer_displacements_name` must match exactly the name the user has defined in the `DEFORM GEOMETRY USING TRANSFER <string>transfer_displacements_name` command line in the BEM region scope.

## 8.3 BEM to Adagio/Andante Transfer

```
BEGIN TRANSFER <string>bem_to_adagio_transfer_name
  INTERPOLATE SURFACE ELEMENTS FROM <string>bem_region_name
      TO <string>adagio_region_name
  SEARCH SURFACE GAP TOLERANCE = <real>surface_tolerance
  SEARCH GEOMETRIC TOLERANCE = <real>normal_tolerance
  SEND BLOCK SURFACE_<integer>bem_surface_exodus_id
      TO SURFACE_<integer>adagio_surface_exodus_id
  SEND FIELD <string>bem_e_press_name STATE NEW
      TO <string>transfer_pressure_name STATE NONE
END [TRANSFER <string>bem_to_adagio_transfer_name]
```

This command block defines the transfer of the electric pressure field calculated in the BEM region to the surface of the mesh for the adagio region. In the `BEGIN TRANFER` command line, `bem_to_adagio_transfer_name` is a user-defined string. It must match a string in a `TRANSFER` command line used in a `NONLINEAR` command block in the solution control scope.

### 8.3.1 Surface Interpolation Designation

```
INTERPOLATE SURFACE ELEMENTS FROM <string>bem_region_name
    TO <string>adagio_region_name
```

This command line specifies that data from the surface elements (faces) of the electrical region are to be transferred to the surface of the structural region mesh via a surface

interpolation. The strings bem_region_name and adagio_region_name must match the names assigned in the BEM REGION and the ADAGIO/ANDANTE REGION command blocks, respectively, in the procedure scope.

### 8.3.2 Send and Receive Surface Definitions

```
SEND BLOCK SURFACE_<integer>bem_surface_exodus_id1
    [SURFACE_bem_surface_exodus_id2 ...]
    TO SURFACE_<integer>adagio_surface_exodus_id1
    [SURFACE_<integer>adagio_surface_exodus_id2 ...]
```

The SEND BLOCK line command defines the surfaces that contain the nodes to be used in the transfer. In this command, bem_surface_exodus_id's must be sidesets defined in the mesh for the bem_region_name and the adagio_surface_exodus_id's must match sidesets on the mesh for the adagio_region_name. As indicated here, lists of surfaces may be used within a single TRANSFER command block, or multiple TRANSFERcommand blocks may be used. In either case, in any particular TRANSFER command block, the surface defined by the union of the BEM sidesets must include topologically the surface defined by the list of Adagio sidesets within a numerical tolerance. In other words, every node in the list of Adagio sidesets must lie within the surface defined by the union of the BEM sidesets.

### 8.3.3 Search Tolerance on Surface Elements

```
SEARCH SURFACE GAP TOLERANCE = <real>surface_tolerance
```

Interpolation of data between different meshes is performed by finding the parametric location of each node in the receive surface within an element of the send surface. For nodes that may lie near the edge of an element, the surface_tolerance specified in this command line is used to determine whether the point actually lies within the element. The value for surface_tolerance is a distance in the appropriate mesh units that is some fraction of a typical element edge length.

### 8.3.4 Search Tolerance for Normals to Surface

```
SEARCH GEOMETRIC TOLERANCE = <real>normal_tolerance
```

For a surface to surface transfer, normal_tolerance establishes a small distance normal to the sending surface for which a node of the receive surface is considered to actually

lie within that send surface element. In other words, this `normal_tolerance` allows a small gap to exist between the meshed surfaces in the two regions that represent the same physical surface.

### 8.3.5 Send and Receive Field Definitions

```
SEND FIELD <string>bem_e_press_name STATE NEW
    TO transfer_pressure_name STATE NONE
```

For loose electrostatic-structural coupling, an electric field is computed over the mesh in its current deformed configuration. The normal of the electric field is used to compute a pressure on each face of the electrical mesh, and this electric pressure is then passed to the structural code so that it can be used to compute nodal loads. For arpeggio, the form of the command to pass electric pressure from a mesh associated with a BEM region to one associated with an Adagio or Andante region is shown above.

In this command, the string `bem_e_press_name` is chosen by the user but must be exactly (including case) a string specified in the `EVALUATE ELECTRIC_PRESSURE AS <string> bem_e_press_name` command line of the `BEGIN POST PROCESS` command block in the BEM region scope. Likewise, the string `transfer_pressure_name` must be the same as specified as the `FIELD VARIABLE` in a `BEGIN PRESSURE` command block in the Adagio region scope.

## 8.4 BEM to BEM Electric Pressure Transfer

```
BEGIN TRANSFER <string>bem_to_bem_epress_transfer_name
  COPY SURFACE ELEMENTS FROM <string>bem_region_name
      TO <string>bem_region_name
  SEND FIELD <string>bem_e_press_name STATE NEW
      TO previousSolution STATE NONE
END [TRANSFER <string>bem_to_bem_epress_transfer_name]
```

In order to test convergence of a time step in the loosely coupled algorithm, a computed quantity for each region at the end of each iteration must be saved as a registered variable named `previousSolution` so that it may be compared to the solution at the end of the next iteration. Adagio has its own internally registered `previousSolution` but for the BEM region, it must be registered and filled through use of this `BEGIN TRANSFER` block. For coupled electrostatic-structural analyses, the electric pressure field is an appropriate variable for the convergence check so it is copied to `previousSolution`. Because it is

a registered name, the keyword previousSolution is case sensitive. The transfer is invoked by the TRANSFER <string>bem_to_bem_epress_transfer_name command line in a TRANSIENT command block in the solution control scope.

### 8.4.1 Surface Element Copy Designation

```
COPY SURFACE ELEMENTS FROM <string>bem_region_name
    TO <string>bem_region_name
```

This type of transfer is simply a copy from one field to another in the same mesh. Because the electric pressure is computed as a face variable on the elements of the BEM mesh, the appropriate copy for this TRANSFER block is between surface elements.

### 8.4.2 Send and Receive Electric Pressure Field

```
  SEND FIELD <string>bem_e_press_name STATE NEW
      TO previousSolution STATE NONE
```

This command line copies the data in the registered field bem_e_press_name to the registered field previousSolution. The convergence algorithm in the code expects a variable specifically named previousSolution so it appears here as a keyword and not as a string chosen by the user. Because the string previousSolution will be used by the transfer operators to register the field in the code, it is case sensitive and must be typed in exactly this form.

## 8.5 BEM to BEM Displacements Transfer

```
BEGIN TRANSFER <string>bem_to_bem_displ_transfer_name
  COPY VOLUME NODES FROM <string>bem_region_name
      TO <string>bem_region_name
  SEND FIELD <string>transfer_displacements_name STATE NEW
      TO <string>transfer_displacement_name STATE OLD
END [TRANSFER <string>bem_to_bem_displ_transfer_name]
```

The BEM region, when used by itself without coupling, is for single-pass static computations and has no concept of state. In order to output the displacements to the BEM results file so that BEM data can be plotted on a deformed mesh plot, the displacement field must

exist in state OLD at the end of a time step. Therefore, the displacement field in BEM must be copied from `STATE NEW` to `STATE OLD`. This is accomplished using a transfer over all nodes of the BEM region and is invoked by the `TRANSFER <string>bem_to_bem_displ_transfer_name` command line in a `TRANSIENT` command block in the solution control scope.

### 8.5.1 Nodal Copy Designation

```
COPY VOLUME NODES FROM <string>bem_region_name
    TO <string>BEM_region_name
```

This type of transfer is simply a copy from one field to another in the same mesh. In the case of a BEM region, the ”volume” is the same as the surface in the sense that both contain exactly the same nodes.

### 8.5.2 Send and Receive Displacement Field

```
SEND FIELD <string>transfer_displacements_name STATE NEW
    TO <string>transfer_displacements_name STATE OLD
```

The BEM displacement field in `STATE NEW` when it is received from Adagio, but it must be copied to `STATE OLD` in order to be available for output to the BEM results file at the end of a time step. This command simply defines the copy of the transferred displacement field from `STATE NEW` to `STATE OLD`.

# Chapter 9

# Quasistatic Microbeam Example

In this chapter, an example is given of a quasistatic structural analysis of an electrostatically-actuated microbeam. After a description of the problem, the details of the mesh development are provided. Finally, the input file that can be used to run this problem is described in detail and some typical results are shown.

## 9.1 Problem Description

This example is a microbeam that is a common component of electrostatically-activated MEMS switches and sensors. In this case the the microbeam is part of an array of beams used for spectral chemical analysis [8]. Figure 9.1 shows an illustration (not to scale) of one beam of this array. Each beam of the array is 500 microns long, 10 microns wide, and 2 microns thick. The beam is fixed at both ends and has an actuating electrode near each end. When a voltage is applied to the electrodes, electrostatic forces are imposed on the beam near each end, pulling the center toward the substrate. As the beam deforms, the electric field increases in intensity because the gap between the beam and the electrodes decreases. This increase in electrostatic load further deforms the beam. Eventually, an equilibrium position is reached in which the stiffness of the beam exactly counteracts the electrostatic pressure load. Different voltages applied to the electrodes for multiple beams will impose different curvatures throughout the array giving rise to an optical diffraction grating. Because the load is dependent on the deformation, this design is best analyzed using coupled electrostatic-structural analysis.

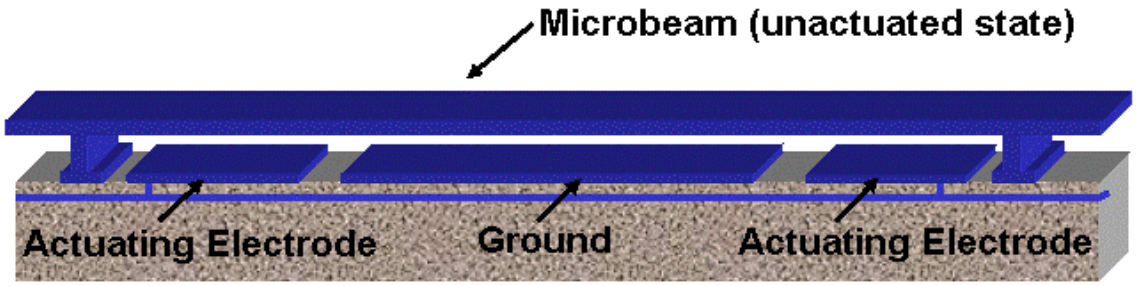


Figure 9.1: Single microbeam of a polychomator array.

## 9.2 Discretized Model

For coupled electrostatic-structural analysis using Arpeggio, two mesh files are required. The first is a volume mesh of the structure for the structural analysis and the second is a surface mesh enclosing the structure for the electrostatic analysis. The CUBIT journal file used to create these meshes is shown in Appendix A. In this example the length units used are microns.

### 9.2.1 Structural Mesh

The structural mesh for this microbeam and a section of the substrate is shown in Figure 9.2.

The structural mesh consists of 8-node hexahedral elements. As can be noted in the CUBIT journal file in Appendix A and shown in Figure 9.2, the beam has two nodesets defined, one spanning the cross-section of each end of the beam. Because this beam is symmetrical about the middle cross-section, only one-half of the beam is modelled. A sideset (a set of nodes associated with the element faces of a surface) is designated that includes all the external element faces of the microbeam. This sideset will be used to transfer the beam deformation to the electrostatic surface mesh and to receive the electric pressures transferred from the electrostatic mesh.

In this case the substrate is included in the structural mesh to provide a contact surface. All the nodes of the substrate will be fixed in the structural analysis.

### 9.2.2 Electrical Surface Mesh

The surface mesh for the electrostatic analysis is shown in Figure 9.3 and the CUBIT journal file used to create it in Appendix A. This mesh is comprised of 4-noded shell

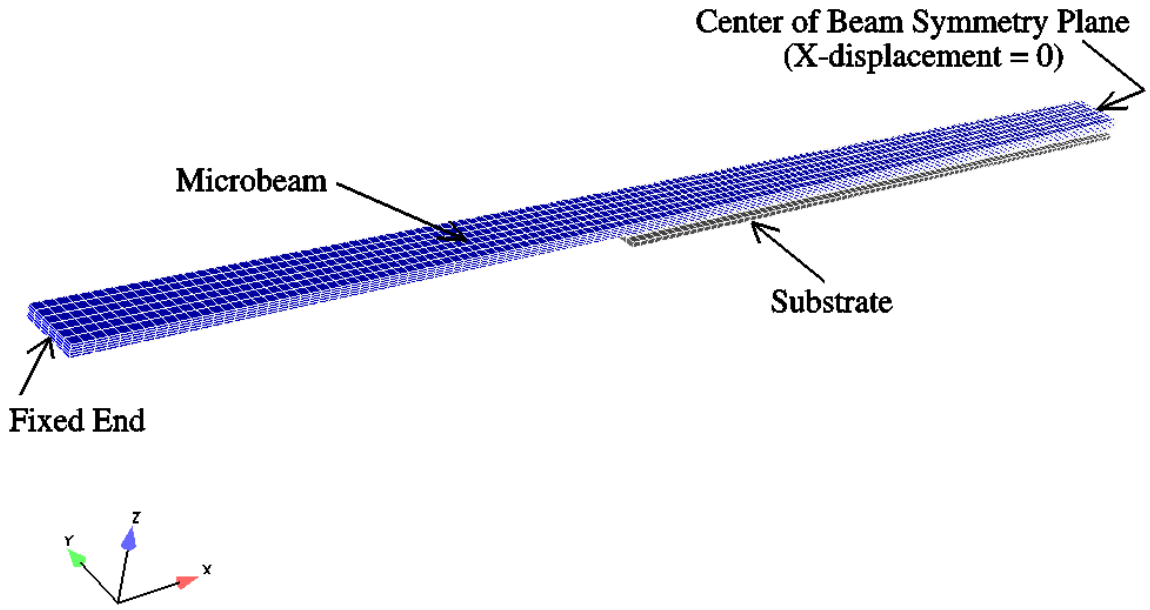


Figure 9.2: Structural volume discretization of one-half of a microbeam.

elements for both the microbeam and the electrode. A shell element is a topologically two-dimensional element that exists in three-dimensional space (as opposed to a quadrilateral element that is flat and is only defined in two dimensions). Because a shell lives in three-dimensional space, it has a normal vector associated with the nodal connectivity. To correctly orient the normals of the boundary mesh, the elements must all have normals that point inward to the body because the domain to be solved is external to the body. For an explanation of consistent normals for an external domain, see Section 6.5.3. (By default, CUBIT will assign the normal of a shell element consistent with the normal of the surface from which it is derived and surfaces external to a volume will have outward-pointing normals. Therefore, it is necessary to include the line

```
SURFACE ALL NORMAL OPPOSITE
```

in the CUBIT input file to get shell elements with inward-pointing normals.)

For this problem, one-half of the beam is modelled. This assumes that the electric field imposed by an electrode near one end of the beam is negligible near the center of the beam and everywhere on the opposite half of the beam.

For the electrostatic mesh, any parts that may potentially have different voltages applied must be designated as separate element blocks. Therefore, for this example, the elements of microbeam surfaces are designated as element block 1, and those on the electrode are in element block 2.

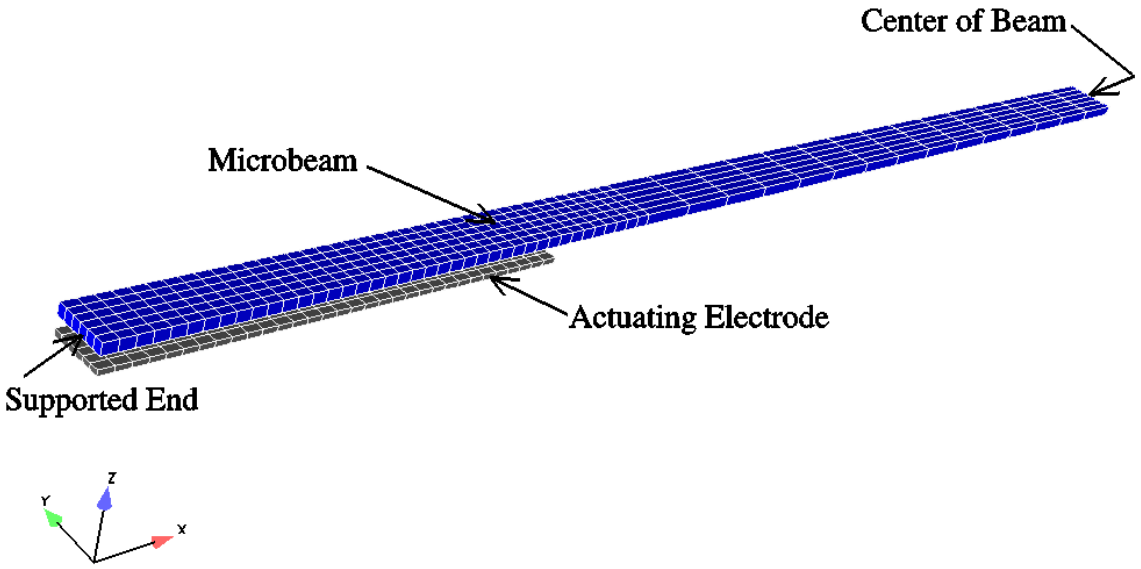


Figure 9.3: Electrical surface discretization of one-half of a microbeam.

For this example, a sideset is designated for each surface of the microbeam. These sidesets will be the surfaces used to compute electric pressure and to transfer this pressure to the corresponding surfaces of the structural mesh.

## 9.3 Input File

The input file for this coupled analysis is discussed in detail in this section. To facilitate explanation, comments are dispersed through the file after every few lines of input. Commands that are part of the input file are delineated from explanatory text by using a computer font and indentation. The input file is repeated, uninterrupted, at the end of the section.

As required of all input files for SIERRA applications, the first line is the `BEGIN SIERRA` statement. Immediately following the SIERRA statement, a title line is shown. In addition to adding information for the user, the text following the `title` keyword will be carried through to the exodus results and restart databases. In this example the use of the `\$` continuation symbol means that the full title is ”Diffraction grating microbeam in microns, MPa, mN, V units”.

The next three lines are comment lines. Any characters after a # or a $ on a line are ignored by the parser.

```
begin sierra quasistatic_MEMS_beam
```

```
  title  Diffraction grating microbeam  \$
  in microns, MPa, mN, V units

#  Model one-half of beam for symmetry
#    Fixed at one end, x-symmetry at other end
$  Only load is electrostatic
```

The following command blocks define the history of applied voltages using functions. The name `voltage` will be used later in the file in the boundary condition for the electrodes in the electrostatic analysis. For this quasistatic analysis, time and voltage load step are synonymous. The `zero` function will be used to apply a constant voltage of 0.0 to the microbeam.

```
  begin definition for function voltage
    type is piecewise linear
    begin values
      0.0      0.0
      200.0    200.0
    end values
  end definition for function voltage

  begin definition for function zero
    type is constant
    begin values
       0.0
    end values
  end definition for function zero
```

The next block of input defines the material model and its physical parameters for the structural part of the analysis. In this case a linear elastic material model ( keyword `ELASTIC` on the `BEGIN PARAMETERS FOR MODEL` line) is specified, requiring material parameters for Young's modulus and Poisson's ratio. A value for density is also required but this value is not actually used for a quasistatic analysis.

```
  begin property specification for material linear_elastic
    density         = 1.0
    begin parameters for model elastic
      youngs modulus  = 160.e9
      poissons ratio  = 0.3
    end parameters for model elastic
  end property specification for material linear_elastic
```

Next the finite element models are defined. Note that the name of the file on the `DATABASE NAME` line is case sensitive. Both the `structural` and the `electrical` models are contained in files with the exodusII database format.

For the structural mesh, additional parameters must be assigned to each of the two element blocks in the file. In this case, all the elements of both blocks will be given that material parameters given above in the `PROPERTY SPECIFICATION FOR MATERIAL` block named `linear_elastic`. In addition, the line command

```
solid mechanics use model Elastic
```

specifies solid elements with an elastic constitutive model.

For the model used in the electrostatic calculation, no additional lines are required beyond specifying the filename and type of the file containing the mesh.

```
begin finite element model structural
  Database Name = microbeam_struct.g
  Database Type = exodusII

  begin parameters for block block_1 block_2
    material linear_elastic
    solid mechanics use model Elastic
  end parameters for block block_1 block_2
end finite element model structural

begin finite element model electrical
  Database Name = microbeam_bem.g
  Database Type = exodusII
end finite element model electrical
```

The parameters for the solvers to be used in the Adagio and BEM regions are defined in the following input blocks. The boundary integral method used for the electrostatics will always result in a dense matrix that is nearly always nonsymmetrical. Therefore, the appropriate solution method is an iterative biconjugate gradient algorithm from the Trilinos package [6]. Currently, no preconditioners are supported in SIERRA for this solution method. Because it is an iterative method, the user must define a maximum number of iterations and a residual norm tolerance. If the residual norm tolerance is not satisfied within the specified maximum number of iterations, the code will abort.

```
### BEM solver
```

```
   begin trilinos equation solver az_params
     solution method = bicgstab
     preconditioning method = none
     maximum iterations = 110
     residual norm tolerance = 1.e-14
   end
```

Adagio uses a nonlinear conjugate gradient iterative solver for its core solver, but can use a linear solver to obtain a preconditioner. The most common linear solver used is the FETI solver [9]. When used in a parallel analysis, the FETI solver utilizes an iterative method requiring the user to specify the maximum number of allowable iterations and a residual norm tolerance. The other parameters shown here are have worked well for several structural analysis applications.

```
###  Adagio preconditioner linear solver
   begin feti equation solver feti
     maximum iterations = 500
     maximum orthogonalization = 500
     preconditioning method = dirichlet
     residual norm tolerance = 1e-4
     corner algorithm = 3
     corner dimensionality = 3
     corner augmentation = none
     coarse solver = sparse
     local solver  = sparse
   end
```

The next two lines are a comment line followed by the line command that initiates the Arpeggio procedure scope.

```
######  PROCEDURE INPUT  ####################

  begin procedure Argo_Procedure
```

The next block of code initiates the region scope for the structural part of the analysis. Because this is a quasistatic analysis, an Adagio region is used. Here it is given the name `ADAGIO_REGION` which will be used on all other commands throughout the input file that refer to this Adagio region.

The first command line shown here in the region scope specifies that the mesh to be used is that defined in the `FINITE ELEMENT MODEL structural` command block in the domain scope.

```
###  ADAGIO REGION INPUT  ####################
     begin adagio region ADAGIO_REGION
       use finite element model structural
```

The next command block specifies which fields are to be output to the exodusII results file. The name `microbeam_struct.e` will be the name of the output file and is thus case sensitive. Nodal, element, and global variables may be requested for output to the results file. The first name following an = sign, is the variable name as used in the Adagio code, and the last string is a user-defined name that will appear in the results file. Vectors and tensors will have appropriate subscripts (x, y, xy, etc.) appended to these output names. For a complete list of variables available for output, see the Adagio/Andante User Reference Manual [2].

The face variable, `pressure_external_transfer` is a variable that results from the coupling with BEM. It is a user-defined string that must match (including case) the name specified in a pressure boundary condition block and a transfer block as shown later in this section.

```
### output description ###
begin Results Output output_adagio
  Database Name = microbeam_struct.e
  Database Type = exodusII
  At Step 0, Increment = 1
  nodal Variables = force_external as f_ext
  nodal Variables = velocity as vel
  nodal Variables = displacement as displ
  nodal Variables = residual as resid
  nodal Variables = contact_slip_velocity as slipVel
  nodal Variables = contact_normal_direction as fcnor_dir
  nodal Variables = contact_tangential_traction_magnitude
    as fctan
  nodal Variables = contact_normal_traction_magnitude
    as fcnor
  nodal Variables = contact_slip_direction_current
    as slipinc_dir
  nodal Variables = contact_slip_increment_current
    as slipinc
  nodal Variables = contact_status as cElem
  nodal Variables = contact_area as cArea
  face Variables = pressure_external_transfer
    as epress
  element Variables = rotated_stress as stress
  global Variables = timestep as timestep
end results output output_adagio
```

The next set of input lines specifies the kinematic boundary conditions for the structural analysis. In this problem, the left end of the beam (`nodelist_1`) is fixed, so a `FIXED DISPLACEMENT` command block is used to specify all three coordinate directions to be fixed. Due to symmetry about the center of the beam, only the left half of the beam is meshed. Therefore, kinematic conditions to simulate symmetry are imposed on a nodeset that lies on the right end of the mesh (`nodelist_2`), a plane located at the middle cross-section of the entire beam. Because the length of the beam is along the x-axis, the symmetry condition is that the x-component of the displacement is fixed. Finally, part of the substrate is included in the analysis for contact purposes, but for this example, all of its nodes are specified as fixed.

```
### definition of BCs ###
  ### boundary conditions on beam

  begin fixed displacement
    node set = nodelist_1
    components = x y z
  end fixed displacement

  begin fixed displacement
    node set = nodelist_2
    components = x
  end fixed displacement


  ### fixed substrate nodes
  begin fixed displacement
    node set = nodelist_10
    components = x y z
  end fixed displacement
```

The next command block defines the loading on the beam. For coupled analyses, the loading is a pressure that is transferred from the electrostatic analysis to the surface of the MEMS parts. In reality, there are pressures exerted on both the microbeam and the electrodes, but because the substrate is not allowed to move, there is no reason to impose a load on it. Therefore, the only pressures that are transferred for this example problem are those on the exterior surface of the microbeam. The sideset in the exodusII file for the structural region has id = 100 and encompasses all six exterior surfaces. The name of the field variable `pressure_external_transfer` is chosen by the user and must match exactly (including case) the name specified in a `TRANSFER` block that appears in the procedure scope and defines the transfer of data from the BEM region to the Adagio region.

```
##  surfaces for pressure transfer
       begin pressure
         surface = surface_100
         field variable = pressure_external_transfer
       end pressure
```

Next, the contact parameters are defined. This example uses frictionless contact.

```
####  Contact  ######
    begin contact definition
      enforcement = frictionless
```

In the next set of input lines, two contact surfaces are specified and named. The first, named `beam`, consists of sideset 2 in the exodusII mesh file. It is the bottom surface of the beam. The second, named `substrate` consists of sideset 7 which is the top surface of the substrate. Later `beam` is specified to be the slave surface and `substrate` to be the master surface for the contact enforcement. Other parameters defined in the `interaction` command block define the tolerances to be used for the search algorithm in determining whether a node of the slave surface has penetrated a face of the master surface. For complete details of the contact algorithm and syntax and interaction parameters, see the Adagio/Andante User Reference Manual [2].

```
      contact surface beam  contains surface_2
      contact surface substrate contains surface_7
       begin surface normal smoothing
         angle = 30.0
         distance = .25
         resolution = EDGE
       end
       begin interaction
         master = substrate
         slave = beam
         normal tolerance = .1
         tangential tolerance = .05
         capture tolerance = .05
       end
     end contact definition
```

The final set of input lines in the Adagio region defines the solver algorithm beginning with the loadstep predictor. Adagio initially predicts the deformation for the end of the current time step using the velocities at the end of the last step. The numerical factors at the end of this line command are multipliers on the predicted velocity. The first value, 1.0, tells Adagio to use exactly the velocity at the current time. The second factor is a multiplier

used only for the very first time step of a time period. In this case, the factor is zero, so the predicted state at the beginning of a new time period will be the same as that computed at the end of the last time period. For a full description of the options available for the predictor, the user is referred to the Adagio/Andante User Reference [2].

```
####  predictor  ######
 loadstep predictor using scale factor = 1.0, 0.0
```

A multilevel solver must be used for problems involving contact. Level 0 of the solver is the core nonlinear conjugate gradient solver. Level 1 is the solver used for the contact constraint problem whose parameters are given within the `CONTROL SLIDING CONTACT` command block. Here a target relative residual and the values for maximum and minimum iterations are specified. If the contact residual does not converge to the target relative residual value (5.e-3), within 400 level 1 iterations, the code will abort. For a full description of the parameters on the multilevel solver, the user is referred to the Adagio/Andante User Reference Manual [2].

```
####  solver     ######
  Begin adagio multilevel solver
    level 1 predictor = none
    Begin control sliding contact controlled_slide
      Target Relative Residual   = 5.e-3
      Maximum Iterations = 400
      Minimum Iterations = 1
    end control sliding contact controlled_slide
```

The next block of input defines the parameters for the core solver. Once the contact constraints are established in a level 1 iteration, they are held fixed and the conjugate gradient solver is applied to the resulting model problem. The line commands that specify the parameters for the core solver are described in detail in the Adagio/Andante User Reference Manual [2]. In most cases, the parameters used for this example should work, although it is the responsibility of the user to determine a value for the target relative residual that provides a sufficiently converged solution for the analysis.

```
    Begin adagio solver cg
       Target Relative Residual = 1.e-4
       Maximum Iterations = 800
       Minimum Iterations = 0
       Orthogonality measure for reset = .5
       Line Search type secant
```

The efficiency of any conjugate gradient solver is dependent on the quality of the preconditioner. In this example, a full tangent preconditioner obtained using the FETI linear solver

is specified. In most cases, this is a very efficient preconditioner. For analyses run on multiple processors, a penalty method must be used for the contact constraints. Therefore, this example input includes the line command `constaint enforcement = penalty` along with a following line that sets the penalty factor at a value of 100. The line command `small number of iterations = 30` specifies that the preconditioner is only to be reformulated at the beginning of a loadstep if the contact constraints have changed or the previous loadstep took more than 30 iterations to converge. When conditions have not changed substantially from one step to the next, reusing the old preconditioner can save a significant amount of computation time.

```
      begin full tangent preconditioner
        linear solver = feti
        constraint enforcement = penalty
        penalty factor = 100
        balance probe = 1
        probe factor = .0001
        small number of iterations = 30
        maximum resets for modelproblem = 2
      end full tangent preconditioner
    end adagio solver cg

  end adagio multilevel solver
```

The next line specifies the end of the Adagio region scope.

```
end adagio region ADAGIO_REGION
```

The next major block of input is the electrostatic BEM region scope. The first command shown here within the region scope specifies the `FINITE ELEMENT MODEL` command block in the domain scope that defines the name and type of file that contains the mesh data.

```
### BEM REGION INPUT  #############

  begin bem region BEM_REGION
    use finite element model electrical
```

The following line specifies use of the aztecoo solver from the Trilinos package. This is a suitable solver for the dense, nonsymmetric matrices that result from use of the boundary integral method. The name `az_params` must match the name of a solver parameter block provided in the domain scope.

```
    use dense solver = aztecoo with az_params
```

The following line specifies that a deformed geometry is to be used rather than the original model coordinates. The field used to determine the deformed shape is the `displacements` field that must be defined in a `TRANSFER` block in the procedure scope.

```
deform geometry using transfer displacements
```

The next block of input lines defines the boundary element zone. Following the `BEGIN BE ZONE` command, the type of partial differential equation to be solved is specified. The appropriate equation for electrostatics is Laplace's equation.

```
begin be zone bem
  PDE = Laplace
```

The next line specifies a `DIRECT FORMULATION` and the solution variables, in this case two scalars, `V` and `E_norm`. The first variable, `V` defines the name for the potential and the second variable `E_norm` is the normal of the flux on the boundary. These names are chosen by the user. Any case may be used, but wherever else they are used in the input file, they must always match the case of this original definition. This line also specifies the use of continuous linear elements, currently the only supported interpolation order in BEM.

```
DIRECT FORMULATION for scalar V and E_norm with continuous
  linear elements
```

The next line specifies that the solution domain is external to the MEMS part. That is, the equation will be integrated over a boundary the includes the surfaces of any MEMS parts and the boundary at infinity. Thus, specification of an `external` means that the solution domain is the air space surrounding the parts.

```
SOLUTION DOMAIN = external
```

The Laplace equation for electrostatics requires one physical parameter, permittivity, for which a value is specified in the next line. Because the solution domain is the surrounding airspace, the value to be given for the permittivity is that of the surrounding air.

```
material parameter for laplace equation permittivity
  = 8.85e-12
```

The next two input lines define the boundary conditions on the mesh. For a typical MEMS analysis, the potential (voltage) is defined at every point on the surfaces of the MEMS parts.

In this example, block_1 consists of all elements on the surface of the beam, an element block in the exodusII meshfile with id=1. The elements of block_2 are those which enclose the electrode. Every element of block_1 is given a voltage that corresponds to the current time as given by the FUNCTION command block with the name voltage in the domain scope. The second line specifies that the voltage on block number 2 is given by the FUNCTION named zero. Alternatively, the voltage function with a scale factor of 0.0 could have been used. This completes the input required for the BE ZONE scope.

```
      BC QUASISTATIC UNIFORM dirichlet at block_1 V
         using function voltage scaled by 1.0
      BC QUASISTATIC UNIFORM dirichlet at block_2 V
         using function zero scaled by 1.0

   end be zone bem
```

The previous input block defined the solution of the primary variables on the finite part of the boundary external to the MEMS parts. The next block of input specifies computation of additional secondary variables on the faces of specified surfaces during a post processing step. Like other command blocks in the SIERRA syntax, pressure is a user-defined name on the BEGIN POST PROCESS line. In the EVALUATE line command, the string electric_pressure is a BEM-defined name and is therefore case˙sensitive. The string e_press is a user-defined name that must match the variable name (including case) used in a TRANSFER block in the procedure scope that defines the transfer of electric pressure from the BEM region to the Adagio region. Because the Adagio region requires a face variable for the transferred electrical pressure, the COMPUTE ON command line specifies that e_press is to be computed on the faces of surface_101.

```
   begin post process pressure
      process equation laplace on zone bem
      evaluate electric_pressure as e_press
      compute on surface_101 faces
   end post process pressure
```

The next block specifies the results file name and format, as well as which variables are to output. The string displacements must match the string used in the DEFORM GEOMETRY USING TRANSFER command line in the BEM region scope. The variable names V and E-norm must match the solution variable names defined in the DIRECT FORMULATION command line in the BE ZONE command block. Finally, the variable name e_press must match that in the EVALUATE electric_pressure line in the POST PROCESS block in the BEM region scope.

```
   BEGIN RESULTS OUTPUT LABEL output_bem
```

```
      database Name = polybeam_bem.e
      database Type = exodusII
      at step 0, increment = 1
      title parallel plate
       nodal Variables = displacements as displ
       nodal variables = V as volt
       nodal variables = E_norm as E_norm
       face variables = e_press as elec_press
    END RESULTS OUTPUT LABEL output_bem
```

The next line signifies the end of the bem region scope.

```
    end bem region BEM_REGION
```

The next major block of input in the procedure scope is the `SOLUTION CONTROL` command block. The first line signifies the beginning of the solution control scope. The subsequent line specifies that a `SYSTEM` named `coupled_em` will be used for this analysis. This `SYSTEM` is defined later in the solution control scope. The user may define multiple `SYSTEM's` in the file, but only one may be used for any particular analysis.

```
###       SOLUTION CONTROL       ###
    begin solution control solution_control
    use system coupled_em
```

The next block of input defines a set of code operations that initialize the analysis. For an electrostatic analysis, `ADVANCE` line commands are required for each of the regions. The names on the `ADVANCE` lines must match the names the user has assigned on the `BEGIN ADAGIO REGION adagio_region_name` and `BEGIN BEM REGION bem_region_name` lines that define the structural and electrostatic region scopes.

```
    begin initialize transient_init
      advance ADAGIO_REGION
      advance BEM_REGION
    end initialize transient_init
```

The system scope specifies the sequence of region executions and the data transfers for a load step within each time period of a simulation. For this example, the code is to be initialized using the sequence defined in the `INITIALIZE transient_init` block above. The next line command defines the termination time for the complete simulation.

```
    begin system coupled_em
      use initialize transient_init
      simulation termination time = 115.0
```

The total simulation time for this example is divided into three time periods, each delineated by `BEGIN TRANSIENT` and `END [TRANSIENT]` commands. The line commands within a nested `NONLINEAR` command block in each transient scope give the order of execution of regions and transfers within an iteration of an individual time step. This sequence of steps will be iterated in until a convergence criteria established by the user in a corresponding `BEGIN PARAMETERS FOR NONLINEAR converge_step_name` block elsewhere in the solution control scope with the same name. Command lines with the `TRANSIENT` command block but outside the `NONLINEAR` command block are executed only once per time step in the order given. All these lines must appear in the order they are to be executed. The input line order shown here should be the same for all time periods in which both regions are to be executed.

```
begin transient t1
  begin nonlinear converge_step1
    transfer bem_to_bem_1
    advance BEM_REGION
    transfer bem_to_adagio
    advance ADAGIO_REGION
    transfer adagio_to_bem
  end nonlinear converge_step1
  transfer bem_to_bem2
end transient t1

begin transient t2
  begin nonlinear converge_step2
    transfer bem_to_bem_1
    advance BEM_REGION
    transfer bem_to_adagio
    advance ADAGIO_REGION
    transfer adagio_to_bem
  end nonlinear converge_step2
  transfer bem_to_bem2
end transient t2

begin transient t3
  begin nonlinear converge_step3
    transfer bem_to_bem_1
    advance BEM_REGION
    transfer bem_to_adagio
    advance ADAGIO_REGION
    transfer adagio_to_bem
  end nonlinear converge_step3
  transfer bem_to_bem2
end transient t3
```

```
end system coupled_em
```

The next block of input defines parameters to be used for each of the three time periods (TRANSIENT blocks) defined in the coupled_em system block above. Each PARAMETERS FOR TRANSIENT block specifies a start time and a termination time for this time period, and the time stepping within each region. The time periods defined by all the transient blocks must be contiguous without overlapping any time, and the final termination time must equal simulation termination time specified in the solution control scope. Within each TRANSIENT block are nested PARAMETERS FOR ADAGIO REGION and PARAMETERS FOR BEM REGION blocks which define the number of time steps each region will take for this time period.

```
begin parameters for transient t1
  start time = 0.0
  termination time = 100.
  begin parameters for adagio region ADAGIO_REGION
     number of time steps = 8
  end parameters for adagio region ADAGIO_REGION
  begin parameters for bem region BEM_REGION

    number of time steps = 8
  end parameters for bem region BEM_REGION
end parameters for transient t1

begin parameters for transient t2
  start time = 100.
  termination time = 110.
  begin parameters for adagio region ADAGIO_REGION
     number of time steps = 2
  end parameters for adagio region ADAGIO_REGION
  begin parameters for bem region BEM_REGION
     number of time steps = 2
  end parameters for bem region BEM_REGION
end parameters for transient t2

begin parameters for transient t3
  start time = 110.
  termination time = 115.
  begin parameters for adagio region ADAGIO_REGION
     number of time steps = 5
  end parameters for adagio region ADAGIO_REGION
  begin parameters for bem region BEM_REGION
     number of time steps = 5
  end parameters for bem region BEM_REGION
```

```
end parameters for transient t3
```

The final block of input lines in the solution control scope are those defining the convergence criteria to be used for each NONLINEAR iteration loop. In this example, the same convergence criteria is specified for all three NONLINEAR blocks.

The converged when "(...)" command line contains C++ logical statements. The first condition, 1 < CURRENT_STEP, specifies that this is not the initial pass through the iteration loop. This statement is necessary because convergence is checked using results of the current iteration with results of the previous iteration. The keyword CURRENT_STEP is syntax used in the code for solution control and is case sensitive. The second condition, ADAGIO_REGION.norm(0.0) < .001, states that the change in the norm of the Adagio region solution from the previous iteration to the current iteration must be less than .001 times the norm of the solution at the previous iteration to satisfy convergence. Likewise, the third condition, BEM_REGION.norm(0.0) < .1, sets a maximum allowed change in the norm of the BEM region solution from the previous to the current iteration. Finally, if all three of the first three conditions are not met, the last condition, 100 < CURRENT_STEP puts a limit of 100 on the number of iterations allowed. Note that this last condition prevents an infinite loop but the code will proceed to the next time step when the maximum number of iterations is reached, regardless whether the conditions for the convergence criteria are met.

```
begin parameters for nonlinear converge_step1
  converged when "(1 < CURRENT_STEP &&
    ADAGIO_REGION.norm(0.0) < .001 &&
    BEM_REGION.norm(0.0) < .1) ||
    (100 < CURRENT_STEP)"
end parameters for nonlinear converge_step1


begin parameters for nonlinear converge_step2
  converged when "(1 < CURRENT_STEP &&
    ADAGIO_REGION.norm(0.0) < .001 &&
    BEM_REGION.norm(0.0) < .1) ||

   (100 < CURRENT_STEP)"
end parameters for nonlinear converge_step2

begin parameters for nonlinear converge_step3
  converged when "(1 < CURRENT_STEP &&
    ADAGIO_REGION.norm(0.0) < .001 &&
    BEM_REGION.norm(0.0) < .1) ||
    (100 < CURRENT_STEP)"
end parameters for nonlinear converge_step3
```

The following line signifies the end of the solution control scope.

```
    end solution control solution_control
```

The next several blocks of input lines each define a type of data transfer. Coupled electrostatic-mechanical analysis with Arpeggio requires four different types of transfer. The user-assigned name on the block is used in the NONLINEAR command blocks to execute a particular transfer. The transfers are discussed below in the order that they appear in the NONLINEAR command blocks

The first transfer that needs to be executed is one within the BEM region itself. This transfer simply copies the field e_press that was computed in the POST PROCESS block of the BEM region to a variable called previousSolution. Because e_press was computed as a face variable, this transfer uses a COPY SURFACE ELEMENTS command line. The name previousSolution is a variable used internally in solution control to test convergence and must always appear exactly as shown in this example. The states new and none will be the same for every analysis.

```
####    DEFINE TRANSFERS    #############

      begin transfer bem_to_bem_1
        copy surface elements from BEM_REGION to BEM_REGION
        send field e_press state new to previousSolution state none
      end transfer bem_to_bem_1
```

The second transfer to be executed is that of the electrostatic pressure from the BEM region to the Adagio region. The first line in this transfer block specifies that this is a transfer of surface element (face) data and the second line specifies exactly which surfaces (sideset id's in the exodusII mesh databases) are involved. In this example, the mesh for the BEM region has a sideset with id 101 and the mesh for the Adagio region has a sideset with id 100. The third line identifies the fields to be transferred. The user-defined name e_press must match the name of the field computed in the POST PROCESS command block in the BEM region, and the user-defined name pressure_external_transfer must match the name of the field in the PRESSURE command block in the Adagio region. The state of these fields to be transferred (new for e_press and none for pressure_external_transfer) will be the same for every analysis. The last two lines define values for search tolerances as described in Section 8.3.

```
      begin transfer bem_to_adagio
        interpolate surface elements from BEM_REGION to ADAGIO_REGION
        send block surface_101 to surface_100
        send field e_press state new to pressure_external_transfer
```

```
    state none
  search surface gap tolerance = 1.e-10
  search geometric tolerance = 1.e-10
end transfer bem_to_adagio
```

The third type of transfer is that of the displacement field computed in the Adagio region to the BEM region so that the electrostatic computation can be computed on the updated, deformed geometry. Like the transfer of pressure, the transfer of displacements is a surface interpolation and the second line in the block defines the surfaces involved. For this transfer, the Adagio name and state for the displacement field is always `displacement state new`. The BEM name for the displacement field, in this case `displacements`, is chosen by the user but must match the name on the `DEFORM GEOMETRY USING TRANSFER` line command in the BEM region scope. This state of this field is always `new`.

```
begin transfer adagio_to_bem
  interpolate surface nodes from ADAGIO_REGION to BEM_REGION
  send block surface_100 to surface_101
  send field displacement state new to displacements
    state new
  search surface gap tolerance = 1.e-10
  search geometric tolerance = 1.e-10
end transfer adagio_to_bem
```

The last transfer used is executed at the end of a transient time step after the nonlinear sequence has converged. It simply copies the BEM field with user-defined name `displacements` from state new to state old. Except for the user-defined names, the syntax for this block will be the same for all BEM-Adagio analyses. The name `displacements` must match the name given to this field on the `DEFORM GEOMETRY USING TRANSFER` command line in the BEM region scope.

```
begin transfer bem_to_bem2
  copy volume nodes from BEM_REGION to BEM_REGION
  send field displacements state new to displacements
    state old
end transfer bem_to_bem2
```

Finally, the last two lines signal the end of the procedure scope and then the end of the SIERRA domain scope.

```
  end procedure Argo_Procedure
end sierra quasistatic_MEMS_beam
```

Below is the entire input file without interruption.

```
begin sierra quasistatic_MEMS_beam
  title  Diffaction grating microbeam  \$
         in MKS units
#  Model one-half of beam for symmetry
#    Fixed at one end, x-symmetry at other end
$  Only load is electrostatic

  begin definition for function voltage
    type is piecewise linear
    begin values
      0.0      0.0
      200.     200.
    end values
  end definition for function voltage

  begin definition for function zero
    type is constant
    begin values
       0.0
    end values
  end definition for function zero

  begin property specification for material linear_elastic
    density         = 1.0
    begin parameters for model elastic
      youngs modulus  = 160.e9
      poissons ratio  = 0.0
    end parameters for model elastic
  end property specification for material linear_elastic

  begin finite element model structural
    Database Name = microbeam_struct.g
    Database Type = exodusII

    begin parameters for block block_1 block_2
      material linear_elastic
      solid mechanics use model Elastic
    end parameters for block block_1 block_2

  end finite element model structural

  begin finite element model electrical
    Database Name = microbeam_bem.g
    Database Type = exodusII
  end finite element model electrical
```

```
### BEM solver
   begin trilinos equation solver az_params
     solution method = bicgstab
     preconditioning method = none
     maximum iterations = 110
     residual norm tolerance = 1.e-10
   end

### Adagio preconditioner linear solver

  begin Aztec equation solver my_aztec
    solution method = cg
    restart iterations = 50
    preconditioning method = dd-ilut
    residual norm tolerance = 1.e-4
  END Aztec EQUATION SOLVER my_aztec

   begin feti equation solver feti
     maximum iterations = 200
     maximum orthogonalization = 500
     preconditioning method = dirichlet
     residual norm tolerance = 1e-4
     corner algorithm = 3
     corner dimensionality = 3
     corner augmentation = none
     coarse solver = sparse
     local solver  = sparse $ iterative $ sparse
     num local subdomains = 16
   end


######  PROCEDURE INPUT  ######################

  begin procedure Argo_Procedure

###  ADAGIO REGION INPUT  ####################
     begin adagio region ADAGIO_REGION
       use finite element model structural

       ### output description ###
       begin Results Output output_adagio
         Database Name = microbeam_struct.e
         Database Type = exodusII
         At Step 0, Increment = 1
```

```
        nodal Variables = force_external as f_ext
        nodal Variables = velocity as vel
        nodal Variables = displacement as displ
        nodal Variables = residual as resid
        nodal Variables = gradient_direction as g
        nodal Variables = search_direction as p
        nodal Variables = contact_slip_velocity as slipVel
        nodal Variables = contact_normal_direction as fcnor_dir
        nodal Variables = contact_tangential_traction_magnitude
          as fctan
        nodal Variables = contact_normal_traction_magnitude
          as fcnor
        nodal Variables = contact_slip_direction_current
          as slipinc_dir
        nodal Variables = contact_slip_increment_current
          as slipinc
        nodal Variables = contact_status as cElem
        nodal Variables = contact_area as cArea
        face Variables = pressure_external_transfer
          as epress
        element Variables = rotated_stress as stress
        global Variables = timestep as timestep
      end results output output_adagio

      ### definition of BCs ###


### boundary conditions on beam

      begin fixed displacement
        node set = nodelist_1
        components = x y z
      end fixed displacement

      begin fixed displacement
        node set = nodelist_2
        components = x
      end fixed displacement


###  fixed ground nodes

      begin fixed displacement
        node set = nodelist_10
        components = x y z
```

```
       end fixed displacement

##  surfaces for pressure transfer
       begin pressure
         surface = surface_100
         field variable = pressure_external_transfer
       end pressure


####  Contact  ######
     begin contact definition
       enforcement = frictionless
       contact surface beam  contains surface_2
       contact surface substrate contains surface_7
       begin surface normal smoothing
         angle = 30.0
         distance = .25
         resolution = EDGE
       end
       begin interaction
         master = substrate
         slave = beam
         normal tolerance = .1
         tangential tolerance = .05
         capture tolerance = .05
       end
     end contact definition


  ####  predictor  ######
     loadstep predictor using scale factor = 1.0, 0.0

  ####  solver     ######


  Begin adagio multilevel solver
      level 1 predictor = none
      Begin control sliding contact controlled_slide
        Target Relative Residual   = 5.e-3
        Maximum Iterations = 400
        Minimum Iterations = 1
      end   control sliding contact controlled_slide

       Begin adagio solver cg
         Target Relative Residual = 1.e-4
```

```
        Maximum Iterations = 800
        Minimum Iterations = 0
        Orthogonality measure for reset = .5

        Line Search type secant
        begin full tangent preconditioner
          linear solver = feti
          constraint enforcement = penalty
          penalty factor = 100
          balance probe = 1
          probe factor = .0001
          small number of iterations = 30
          maximum resets for modelproblem = 2
        end full tangent preconditioner
      end   adagio solver cg

    end   adagio multilevel solver

     end adagio region ADAGIO_REGION

### BEM REGION INPUT  #############

    begin bem region BEM_REGION
      use finite element model electrical
      use dense solver = aztecoo with az_params
      deform geometry using transfer displacements

      begin be zone bem
        PDE = Laplace
        material parameter for laplace equation permittivity
          = 8.85e-12
        solution domain = external

        BC QUASISTATIC UNIFORM dirichlet at block_1 V
          using function voltage scaled by 1.0
        BC QUASISTATIC UNIFORM dirichlet at block_2 V
          using function zero scaled by 1.0

        DIRECT FORMULATION for scalar V and E_norm with continuous
          linear elements

      end be zone bem

      begin post process pressure
         process equation laplace on zone bem
```

```
         evaluate electric_pressure as e_press
         compute on surface_101 faces
      end post process pressure

      BEGIN RESULTS OUTPUT LABEL output_bem
         database Name = microbeam_bem.e
         database Type = exodusII
         at step 0, increment = 1
         title parallel plate
          nodal Variables = displacements as displ
          nodal Variables = physical_coordinates as pcoord
          nodal variables = V as volt
          nodal variables = E_norm as E_norm
          face variables = e_press as e_press
      END RESULTS OUTPUT LABEL output_bem

      end bem region BEM_REGION


###       SOLUTION CONTROL       ###
    begin solution control description

      use system main

      begin initialize mytransient_init
        advance ADAGIO_REGION
        advance BEM_REGION
      end initialize mytransient_init

      begin system main

        use initialize mytransient_init

        simulation termination time = 115.0

        begin transient t1
          begin nonlinear converge_step1
            transfer bem_to_bem_1
            advance BEM_REGION
            transfer bem_to_adagio
            advance ADAGIO_REGION
            transfer adagio_to_bem
          end nonlinear converge_step1
          transfer bem_to_bem2
        end transient t1
```

```
  begin transient t2
    begin nonlinear converge_step2
      transfer bem_to_bem_1
      advance BEM_REGION
      transfer bem_to_adagio
      advance ADAGIO_REGION
      transfer adagio_to_bem
    end nonlinear converge_step2
    transfer bem_to_bem2
  end transient t2

  begin transient t3
    begin nonlinear converge_step3
      transfer bem_to_bem_1
      advance BEM_REGION
      transfer bem_to_adagio
      advance ADAGIO_REGION
      transfer adagio_to_bem
    end nonlinear converge_step3
    transfer bem_to_bem2
  end transient t3

end system main

begin parameters for transient t1
  start time = 0.0
  termination time = 100.
  begin parameters for adagio region ADAGIO_REGION
     number of time steps = 8
  end parameters for adagio region ADAGIO_REGION
  begin parameters for bem region BEM_REGION
     number of time steps = 8
  end parameters for bem region BEM_REGION
end parameters for transient t1

begin parameters for transient t2
  start time = 100.
  termination time = 110.
  begin parameters for adagio region ADAGIO_REGION
     number of time steps = 2
  end parameters for adagio region ADAGIO_REGION
  begin parameters for bem region BEM_REGION
     number of time steps = 2
  end parameters for bem region BEM_REGION
```

```
    end parameters for transient t2

    begin parameters for transient t3
      start time = 110.
      termination time = 115.
      begin parameters for adagio region ADAGIO_REGION
         number of time steps = 5
      end parameters for adagio region ADAGIO_REGION
      begin parameters for bem region BEM_REGION
         number of time steps = 5
      end parameters for bem region BEM_REGION
    end parameters for transient t3

    begin parameters for nonlinear converge_step1
      converged when "(1 < CURRENT_STEP
          && ADAGIO_REGION.norm(0.0) < .001
          && BEM_REGION.norm(0.0) < .1)
          ||  (100 < CURRENT_STEP)"
    end parameters for nonlinear converge_step1


    begin parameters for nonlinear converge_step2
      converged when "(1 < CURRENT_STEP
          && ADAGIO_REGION.norm(0.0) < .001
          && BEM_REGION.norm(0.0) < .1)
          ||  (100 < CURRENT_STEP)"
    end parameters for nonlinear converge_step2

    begin parameters for nonlinear converge_step3
      converged when "(1 < CURRENT_STEP
          && ADAGIO_REGION.norm(0.0) < .001
          && BEM_REGION.norm(0.0) < .1)
          ||  (100 < CURRENT_STEP)"
    end parameters for nonlinear converge_step3

  end solution control description



####    DEFINE TRANSFERS    #############


    begin transfer bem_to_adagio
      interpolate surface elements from BEM_REGION to ADAGIO_REGION
      send block surface_101 to surface_100
```

```
        send field E_PRESS state new to
           pressure_external_transfer state none
        search surface gap tolerance = 1.e-10
        search geometric tolerance = 1.e-10
      end transfer bem_to_adagio

      begin transfer adagio_to_bem
        interpolate surface nodes from ADAGIO_REGION to BEM_REGION
        send block surface_100 to surface_101
        send field displacement state new to
           displacements state new
        search surface gap tolerance = 1.e-10
        search geometric tolerance = 1.e-10
      end transfer adagio_to_bem

      begin transfer bem_to_bem_1
        copy surface elements from BEM_REGION to BEM_REGION
        send field E_PRESS state new to
           previousSolution state none
      end transfer bem_to_bem_1

      begin transfer bem_to_bem2
        copy volume nodes from BEM_REGION to BEM_REGION
        send field displacements state new to
           displacements state old
      end transfer bem_to_bem2

  end procedure Argo_Procedure

end sierra quasistatic_MEMS_beam
```

## 9.4 Results

Figure 9.4 is a plot of the displacement of the center of the beam toward the substrate as a function of applied voltage. At approximately 110 volts, the center of the beam contacts the substrate located 1.6 microns below the beam. The applied electrostatic pressure then continues to hold the beam against the substrate so the displacement stays at -1.6 microns.

Figure 9.5 shows the distribution of electrical pressure on the bottom of the microbeam at the maximum applied voltage. The only area where the electric pressure is not nearly zero is on the area lying directly above the electrode. Over this area, the electrical pressure increases from left to right because the gap under the deformed beam is decreasing from

the fixed end at the left toward the center on the right.

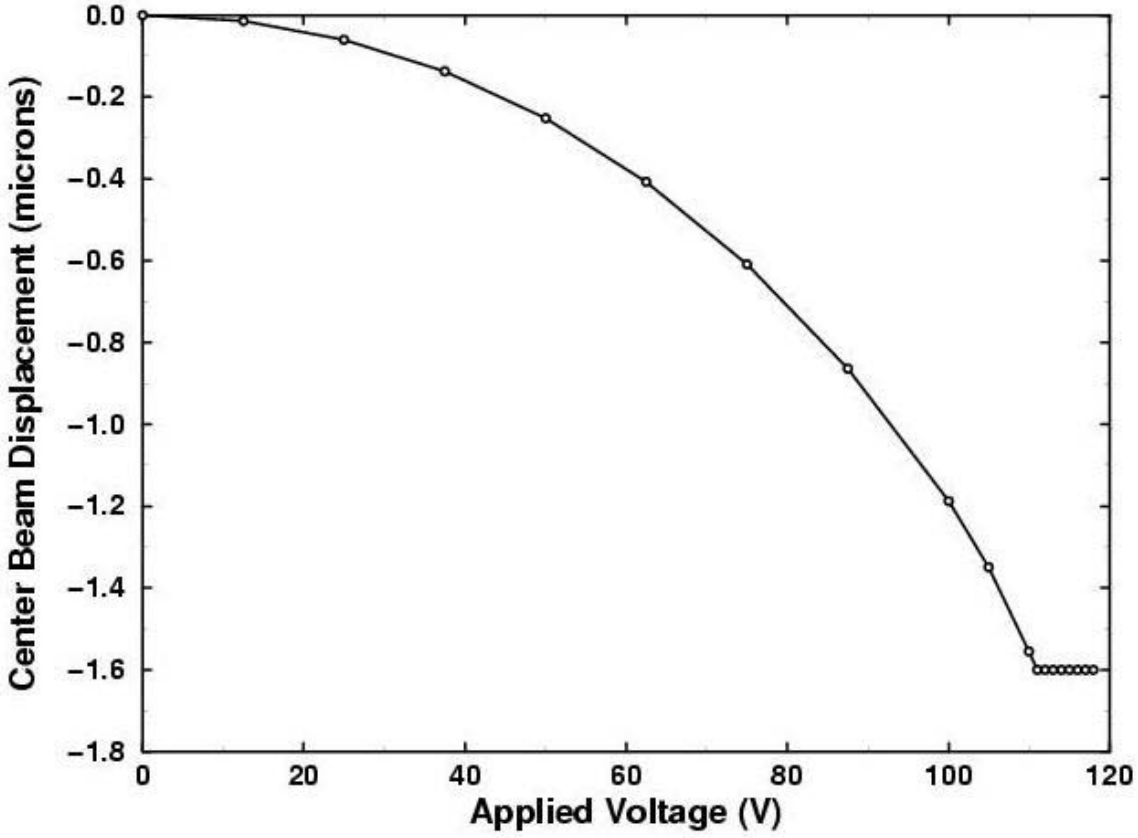


Figure 9.4: Displacement of center of microbeam.

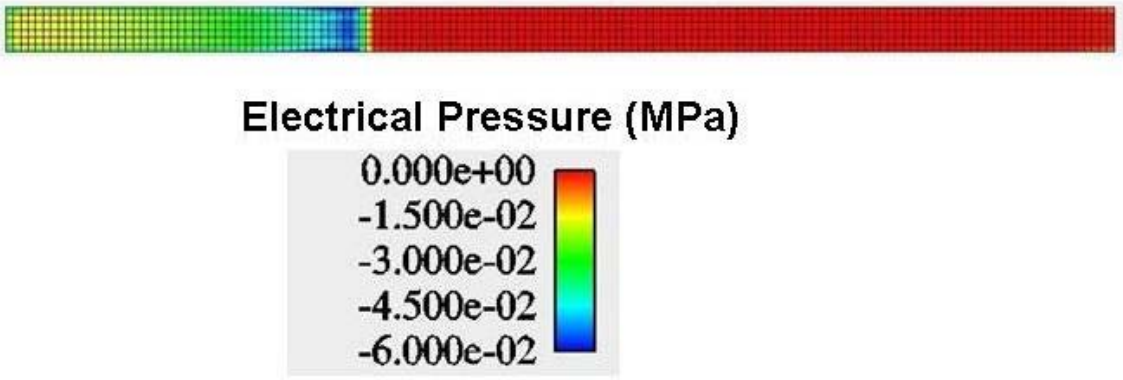


Figure 9.5: Distribution of electrical pressure on bottom of microbeam.

# Chapter 10

# Dynamic Microbeam Example

In this chapter, an example is given of an implicit dynamic analysis of an electrostatically-actuated microbeam. This problem geometry is the same as the quasistatic beam analysis of Chapter 9. The only difference here is that the beam is loaded with a voltage pulse and an implicit dynamic analysis via an Andante region is used. Because the time for the electric field to stabilize is much smaller than the time scale for structural deformation (in this case the duration of the voltage pulse and the period of the fundamental beam bending mode), an electrostatic analysis is appropriate. This example will illustrate how flexible coupling definitions made possible through solution control can be utilized to tailor the analysis to be more efficient.

## 10.1 Problem Description

This example is the implicit dynamic analysis for an electrostatically-activated microbeam that is part of an array of beams used for spectral chemical analysis. The design of this MEMS array is described in Reference [8]. Figure 9.1 shows a illustration (not to scale) of one beam of this array. Each beam of the array is 500 microns long, 10 microns wide, and 2 microns thick. The beam is fixed at both ends and has an actuating electrode near each end. When a voltage is applied to the electrodes, electrostatic forces are imposed on the beam near each end, pulling the center toward the substrate. As the beam deforms, the electric field increases in intensity because the gap between the beam and the electrodes decreases. This increase in electrostatic load further deforms the beam. Because the load is dependent on the deformation, this is best solved using a coupled electrostatic-structural analysis. For the implicit dynamic analysis, the actuating pads are pulsed with a voltage of short duration and then the potential is dropped back to zero. The voltage pulse imposes a pressure pulse that starts the beam moving and the beam continues to vibrate after the voltage is shut off due to the kinetic and internal energies induced. Therefore, unlike the

quasistatic problem, the dynamic problem requires a solution be computed even when the electrostatic load is zero.

## 10.2 Discretized Model

For coupled electrostatic-structural analysis using Arpeggio, two mesh files are required. The first is a volume mesh of the structure for the structural analysis and the second is a surface mesh enclosing the structure for the electrostatic analysis. The structural and electrical meshes used for this implicit dynamic analysis are the same as those used for the quasistatic analysis described in Section 9.2.

## 10.3 Input File

The input file for this coupled analysis is discussed in detail in this section. To facilitate explanation, comments are dispersed through the file after every few lines of input. Commands that are part of the input file are delineated from explanatory text by using a computer font and indentation. The input file is repeated, uninterrupted, at the end of the section.

The input file for an implicit dynamic structural analysis coupled with the electrostatic computation is very similar to that for a quasistatic structural analysis. The main differences are the definition of the voltage history, the time scale used in the `TRANSIENT` block definitions, and the use of an Andante region rather than an Adagio region. The commands within an Andante region are also mostly the same as those in an Adagio region. However, for the benefit of the reader, the entire input file will be included here even though much of it is a repeat of Chapter 9.

As required of all input files for SIERRA applications, the first line is `BEGIN SIERRA` statement. Immediately following the SIERRA statement, a title line is shown. In addition to adding information for the user, the text following the `TITLE` keyword will be carried through to the exodus results and restart databases. In this example the use of the `\$` continuation symbol means that the full title is ”Diffraction grating microbeam in MKS units”.

The next three lines are comment lines. Any characters after a # or a $ on a line are ignored by the parser.

```
begin sierra dynamic_MEMS_beam
  title  Diffraction grating microbeam  \$
  in MKS units
```

```
#  Model one-half of beam for symmetry
#    Fixed at one end, x-symmetry at other end
$  Only load is electrostatic
```

The next command block defines the history of applied voltages using functions. The name `voltage` will be used later in the file in the boundary condition for the electrodes in the electrostatic analysis. The `zero` function will be used to apply a constant voltage of 0.0 to the microbeam.

```
  begin definition for function voltage
    type is piecewise linear
    begin values
      0.0      0.0
      2.e-6    100.
      8.e-6    100.
      10.e-6    0.
      16.e-6   0.0
     end values
  end definition for function voltage

  begin definition for function zero
    type is constant
    begin values
       0.0
    end values
  end definition for function zero
```

The next block of input defines the material model and its physical parameters for the structural part of the analysis. In this example, a linear elastic material model ( keyword `ELASTIC` on the `BEGIN PARAMETERS FOR MODEL` line) is specified, requiring material parameters for Young's modulus and Poisson's ratio. For dynamics problems, the density of the material must also be given.

```
  begin property specification for material linear_elastic
    density         = 1.928e4  #kg/m^3
    begin parameters for model elastic
      youngs modulus  = 106.6e9 #Pa
      poissons ratio  = 0.4
    end parameters for model elastic
  end property specification for material linear_elastic
```

Next the finite element models are defined. Note that the name of the file on the `DATABASE NAME` line is case sensitive. Both the `structural` and the `electrical` models are contained in exodusII databases.

For the structural mesh, additional parameters must be assigned to each of the two element blocks in the file. In this case, all the elements of both blocks will be given that material parameters given above in the `PROPERTY SPECIFICATION FOR MATERIAL` block named `linear_elastic`. In addition, the line command

```
solid mechanics use model Elastic
```

specifies solid elements with an elastic constitutive model.

For the model used in the electrostatic calculation, no additional lines are required beyond specifying the filename and type of the file containing the mesh.

```
begin finite element model structural
  Database Name = microbeam_struct.g
  Database Type = exodusII

  begin parameters for block block_1 block_2
    material linear_elastic
    solid mechanics use model Elastic
  end parameters for block block_1 block_2
end finite element model structural

begin finite element model electrical
  Database Name = microbeam_bem.g
  Database Type = exodusII
end finite element model electrical
```

The parameters for the solvers to be used in the Andante and BEM regions are defined in the following input blocks. The boundary integral method used for the electrostatics will always result in a dense matrix that is nearly always nonsymmetrical. Therefore, the appropriate solution method is an iterative biconjugate gradient algorithm from the Trilinos package [6]. Currently, no preconditioners are supported in SIERRA for this solution method. Because it is an iterative method, the user must define a maximum number of iterations and a residual norm tolerance. If the residual norm tolerance is not satisfied within the specified maximum number of iterations, the code will abort.

```
### BEM solver
```

```
   begin trilinos equation solver az_params
     solution method = bicgstab
     preconditioning method = none
     maximum iterations = 110
     residual norm tolerance = 1.e-14
   end
```

Andante uses a nonlinear conjugate gradient iterative solver for its core solver, but can use a linear solver to obtain a preconditioner. The most common linear solver used is the FETI solver [9]. When used in a parallel analysis, the FETI solver uses an iterative method requiring the user to specify the maximum number of allowable iterations and a residual norm tolerance. The other parameters shown here have worked well for several structural analysis applications.

```
###  Andante preconditioner linear solver
   begin feti equation solver feti
     maximum iterations = 500
     maximum orthogonalization = 500
     preconditioning method = dirichlet
     residual norm tolerance = 1e-4
     corner algorithm = 3
     corner dimensionality = 3
     corner augmentation = none
     coarse solver = sparse
     local solver  = sparse
   end
```

The next two lines are a comment line followed by the line command that initiates the Arpeggio procedure scope.

```
######  PROCEDURE INPUT  ####################

  begin procedure Argo_Procedure
```

The next set of lines initiates the structural region scope. Because this is an implicit dyanamic structural analysis, an Andante region is used. Here it is given the name `ANDANTE_REGION` which will be used on all other commands throughout the input file that refer to this region. The next command line specifies that the mesh to be used for the region is that defined in the `FINITE ELEMENT MODEL structural` command block defined in the domain scope.

```
###  ANDANTE REGION INPUT  ###################
     begin andante region ANDANTE_REGION
       use finite element model structural
```

The next command block specifies which fields are to be output to the exodusII results file. The name `microbeam_struct.e` will be the name of the output file and is thus case sensitive. Nodal, element, and global variables may be requested for output to the results file. The first name following an = sign, is a variable name as defined internally in Andante, and the last string is a user-defined name that will appear in the results file. Vectors and tensors will have appropriate subscripts (x, y, xy, etc.)appended to these output names. For a complete list of variables available for output, see the Adagio/Andante User Reference Manual [2].

The face variable, `pressure_external_transfer` is a variable that results from the coupling with BEM. It is a user-defined string that must match (including case) the name specified in a pressure boundary condition block and a transfer block as shown later in this section.

```
### output description ###
begin Results Output output_andante
  Database Name = microbeam_struct.e
  Database Type = exodusII
  At Step 0, Increment = 1
  nodal Variables = force_external as f_ext
  nodal Variables = velocity as vel
  nodal Variables = displacement as displ
  nodal Variables = acceleration as accl
  nodal Variables = residual as resid
  nodal Variables = contact_slip_velocity as slipVel
  nodal Variables = contact_normal_direction as fcnor_dir
  nodal Variables = contact_normal_traction_magnitude
    as fcnor
  nodal Variables = contact_status as cElem
  nodal Variables = contact_area as cArea
  face Variables = pressure_external_transfer
    as epress
  element Variables = rotated_stress as stress
  global Variables = timestep as timestep
end results output output_andante
```

The next set of input lines specifies the kinematic boundary conditions for the structural analysis. In this problem, the left end of the beam (`nodelist_1`) is fixed, so a `FIXED DISPLACEMENT` command block is used to specify all three coordinate directions to be fixed. Due to symmetry about the center of the beam, only the left half of the beam is meshed. Therefore, kinematic conditions to simulate symmetry are imposed on a nodeset that lies on the right end of the mesh (`nodelist_2`), a plane located at the middle cross-section of the entire beam. Because the length of the beam is along the x-axis, the symmetry

condition is that the x-component of the displacement is fixed. Finally, part of the substrate is included in the analysis for contact purposes, but for this example, all of its nodes are specified as fixed.

```
### definition of BCs ###
  ### boundary conditions on beam

  begin fixed displacement
    node set = nodelist_1
    components = x y z
  end fixed displacement

  begin fixed displacement
    node set = nodelist_2
    components = x
  end fixed displacement


  ### fixed substrate nodes
  begin fixed displacement
    node set = nodelist_10
    components = x y z
  end fixed displacement
```

The next command block defines the loading on the beam. For coupled analyses, the loading is a pressure that is transferred from the electrostatic analysis to the surface of the MEMS parts. In reality, there are pressures exerted on both the microbeam and the electrodes, but because the substrate is not allowed to move, there is no reason to impose a load on it. Therefore, the only pressures that are transferred for this example problem are those on the exterior surface of the microbeam. The sideset in the exodusII file for the structural region has id = 100 and encompasses all six exterior surfaces. The name of the field variable `pressure_external_transfer` is chosen by the user and must match exactly (including case) the name specified in a `TRANSFER` block that appears in the procedure scope and defines the transfer of data from the BEM region to the Andante region.

```
##  surfaces for pressure transfer
    begin pressure
      surface = surface_100
      field variable = pressure_external_transfer
    end pressure
```

Next, the contact parameters are defined. This example uses frictionless contact.

```
####  Contact  ######
    begin contact definition
      enforcement = frictionless
```

In the next set of input lines, two contact surfaces are specified and named. The first, named `beam`, consists of sideset 2 in the exodusII mesh file. It is the bottom surface of the beam. The second, named `substrate` consists of sideset 7 which is the top surface of the substrate. Later `beam` is specified to be the slave surface and `substrate` to be the master surface for the contact enforcement. Other parameters defined in the `interaction` command block define the tolerances to be used for the search algorithm in determining whether a node of the slave surface has penetrated a face of the master surface. For complete details of the contact algorithm and syntax and interaction parameters, see the Adagio/Andante User Reference Manual [2].

```
      contact surface beam  contains surface_2
      contact surface substrate contains surface_7
       begin surface normal smoothing
         angle = 30.0
         distance = .25
         resolution = EDGE
       end
       begin interaction
         master = substrate
         slave = beam
         normal tolerance = .1e-6  #meters
         tangential tolerance = .05e-6  #meters
         capture tolerance = .05e-6  #meters
       end
     end contact definition
```

The final set of input lines in the Andante region defines the solver algorithm beginning with the loadstep predictor. Initially the deformation at the end of the current time step is predicted using the velocities at the end of the last step. The numerical factors at the end of this line command are multipliers on the predicted velocity. The first value, 1.0, tells Andante to use exactly the velocity at the current time. The second factor is a multiplier used only for the very first time step of a time period. In this case, the factor is zero, so the predicted state at the beginning of a new time period will be the same as that computed at the end of the last time period. For a full description of the options available for the predictor, the user is referred to the Adagio/Andante User Reference Manual [2].

```
  ####  predictor  ######
   loadstep predictor using scale factor = 1.0, 0.0
```

A multilevel solver must be used for problems involving contact. An Andante region uses the same multilevel and core solver as an Adagio region as denoted in the `BEGIN ADAGIO MULTILEVEL SOLVER` and `BEGIN ADAGIO SOLVER` lines. Level 0 of the solver is the core nonlinear conjugate gradient solver. Level 1 is the solver used for the contact constraint problem whose parameters are given within the `CONTROL SLIDING CONTACT` command block. Here a target relative residual and the values for maximum and minimum iterations are specified. If the contact residual does not converge to the target relative residual value (5.e-3), within 400 level 1 iterations, the code will abort. For a full description of the parameters on the multilevel solver, the user is referred to the Adagio/Andante User Reference Manual [2].

```
####  solver     ######
  Begin adagio multilevel solver
    level 1 predictor = none
    Begin control sliding contact controlled_slide
      Target Relative Residual   = 5.e-3
      Maximum Iterations = 400
      Minimum Iterations = 1
    end control sliding contact controlled_slide
```

The next block of input defines the parameters for the core solver. Once the contact constraints are established in a level 1 iteration, they are held fixed and the conjugate gradient solver is applied to the resulting model problem. The line commands that specify the parameters for the core solver are described in detail in the Adagio/Andante User Reference Manual [2]. For this example, the reference specified for convergence checking is the norm of the internal forces. The default is the norm of the external forces, but for an Andante analysis such as this one, there is a period of time when the external forces are identically zero. To force the convergence in a load step in which the reference norm may be zero, the `Acceptable Residual` line will allow convergence based on the actual residual value rather than the target relative residual, but only when the maximum iterations have been reached. In most cases, the parameters used for this example should work, although it is the responsibility of the user to determine a value for the target relative residual of that provides a sufficiently converged solution for the analysis.

```
    Begin adagio solver cg
       Reference =Internal
       Target Relative Residual = 1.e-4
       Acceptable Residual = 1.e-12
       Maximum Iterations = 800
       Minimum Iterations = 2
       Orthogonality measure for reset = .5
       Line Search type secant
```

The efficiency of any conjugate gradient solver is dependent on the quality of the preconditioner. In this example, a full tangent preconditioner obtained using the FETI linear solver is specified. In most cases, this is a very efficient preconditioner. For analyses run on multiple processors, a penalty method must be used for the contact constraints. Therefore, this example input includes the line command `constaint enforcement = penalty` along with a following line that sets the penalty factor at a value of 100. The line command `small number of iterations = 30` specifies that the preconditioner is only to be reformulated at the beginning of a loadstep if the contact constraints have changed or the previous loadstep took more than 30 iterations to converge. When conditions have not changed substantially from one step to the next, reusing the old preconditioner can save a significant amount of computation time.

```
      begin full tangent preconditioner
        linear solver = feti
        constraint enforcement = penalty
        penalty factor = 100
        balance probe = 1
        probe factor = .0001
        small number of iterations = 30
        maximum resets for modelproblem = 2
      end full tangent preconditioner
    end adagio solver cg

  end adagio multilevel solver
```

The next line specifies the end of the Andante region scope.

```
end andante region ANDANTE_REGION
```

The next major block of input is the electrostatic BEM region scope. The first command SHOWN here within the region scope specifies the `FINITE ELEMENT MODEL` command block in the domain scope that defines the name and type of file that contains the mesh data.

```
### BEM REGION INPUT  #############

  begin bem region BEM_REGION
    use finite element model electrical
```

The following line specifies use of the aztecoo solver from the Trilinos package. This is a suitable solver for the dense, nonsymmetric matrices that result from use of the boundary integral method. The name `az_params` must match the name of a solver definition block provided in the domain scope.

```
use dense solver = aztecoo with az_params
```

The following line specifies that a deformed geometry is to be used rather than the original model coordinates. The field to be used to determined the deformed shape is the `displacements` field that must be defined in a `TRANSFER` block in the procedure scope.

```
deform geometry using transfer displacements
```

The next block of input lines defines the boundary element zone. Following the `BEGIN BE ZONE` command, the type of partial differential equation to be solved is specified. For electrostatic this is Laplace's equation.

```
begin be zone bem
  PDE = Laplace
```

The next line specifies a `DIRECT FORMULATION` and the solution variables, in this case two scalars, `V` and `E_norm`. The first variable, `V` defines the name for the potential, and the second variable `E_norm` is the normal of the flux on the boundary. These names are chosen by the user. Any case may be used, but wherever they are used in the input file, they must always match the case of this original definition. This line also specifies the use of continuous linear elements, currently the only supported interpolation order in BEM.

```
DIRECT FORMULATION for scalar V and E_norm with continuous
  linear elements
```

The next line specifies that the solution domain is external to the MEMS part. That is, the equation will be integrated over a boundary the includes the surfaces of any MEMS parts and the boundary at infinity. Thus, specification of an `external` means that the solution domain is the air space surrounding the parts.

```
SOLUTION DOMAIN = external
```

The Laplace equation for electrostatics requires one physical parameter, permittivity, for which a value is specified in the next line. Because the solution domain is the surrounding airspace, the value to be given for the permittivity is that of the surrounding air.

```
material parameter for laplace equation permittivity
  = 8.85e-12
```

The next two input lines define the boundary conditions on the mesh. For a typical MEMS analysis, potential (voltage) is defined at every point on the surfaces of the MEMS parts.

In this example, `block_1` consists of all elements on the surface of the beam, an element block in the exodusII meshfile with id=1. The elements of `block_2` are those which enclose the electrode. Every element of `block_1` is given a voltage that corresponds to the current time as given by the `FUNCTION` command block with the name `voltage` in the domain scope. The second line specifies that the voltage on block number 2 is given by the `FUNCTION` named `zero`. Alternatively, the `voltage` function with a scale factor of 0.0 could have been used. This completes the input required for the `BE ZONE` command block.

```
      BC QUASISTATIC UNIFORM dirichlet at block_1 V
         using function voltage scaled by 1.0
      BC QUASISTATIC UNIFORM dirichlet at block_2 V
         using function zero scaled by 1.0

   end be zone bem
```

The previous input block defined the solution of the primary variables on the finite part of the boundary external to the MEMS parts. The next block of input specifies computation of additional secondary variables during a post processing step. Like other command blocks in the SIERRA syntax, `pressure` is a user-defined name on the `BEGIN POST PROCESS` line. In the `EVALUATE` line command, the string `electric_pressure` is a BEM-defined name and is therefore case˙sensitive. The string `e_press` is a user-defined name that must match the variable name (including case) used in a `TRANSFER` block in the procedure scope that defines the transfer of electric pressure from the BEM region to the Andante region. Because the Andante region requires a face variable for the transferred electrical pressure, the `COMPUTE ON` command line specifies that `e_press` is to be computed on the faces of `surface_101`.

```
   begin post process pressure
      process equation laplace on zone bem
      evaluate electric_pressure as e_press
      compute on surface_101 faces
   end post process pressure
```

The next block specifies the results file name and format, as well as which variables are to output. The string `displacements` must match the string used in the `DEFORM GEOMETRY USING TRANSFER` command line in the BEM region scope. The variable names `V` and `E-norm` must match the solution variable names defined in the `DIRECT FORMULATION` command line in the `BE ZONE` command block. Finally, the variable name `e_press` must match that in the `EVALUATE electric_pressure` line in the `POST PROCESS` block in the BEM region scope.

```
BEGIN RESULTS OUTPUT LABEL output_bem
   database Name = polybeam_bem.e
   database Type = exodusII
   at step 0, increment = 1
   title parallel plate
    nodal Variables = displacements as displ
    nodal Variables = physical_coordinates as pcoord
    nodal variables = V as volt
    nodal variables = E_norm as E_norm
    face variables = e_press as e_press
END RESULTS OUTPUT LABEL output_bem
```

The next line signifies the end of the BEM region scope.

```
end bem region BEM_REGION
```

The next major block of input in the procedure scope is the `SOLUTION CONTROL` command block. The first line signifies the beginning of the solution control scope. The subsequent line specifies that a `SYSTEM` named `coupled_em` will be used for this analysis. This `SYSTEM` is defined later in the solution control scope. The user may define multiple `SYSTEM's` in the file, but only one may be used for any particular analysis.

```
###       SOLUTION CONTROL       ###
    begin solution control solution_control
    use system coupled_em
```

The next block of input defines a set of code operations that initialize the analysis. For an electrostatic analysis, `ADVANCE` line commands are required for each of the regions. The names on the `ADVANCE` lines must match the names the user has assigned on the `BEGIN ANDANTE REGION andante_region_name` and `BEGIN BEM REGION bem_region_name` lines that define the structural and electrostatic region scopes.

```
begin initialize transient_init
  advance ANDANTE_REGION
  advance BEM_REGION
end initialize transient_init
```

The system scope specifies the sequence of region executions and the data transfer for a load step within each user-defined time period. For this example, the code is to be initialized using the sequence defined in the `INITIALIZE transient_init` block above. The other line command in the system scope defines the termination time for the complete simulation.

```
begin system coupled_em
  use initialize transient_init
  simulation termination time = 16.e-6
```

The total simulation time for this example is divided into three time periods, each delineated by `BEGIN TRANSIENT` and `END [TRANSIENT]` commands. The line commands within a nested `NONLINEAR` command block in each transient scope give the order of execution of regions and transfers within an iteration of an individual time step. This sequence of steps will be iterated in until a convergence criteria established by the user in a corresponding `BEGIN PARAMETERS FOR NONLINEAR converge_step_name` block elsewhere in the solution control scope with the same name. Command lines with the `TRANSIENT` command block but outside the `NONLINEAR` command block are executed only once per time step in the order given. All these lines must appear in the order they are to be executed. The input line order shown here should be the same for all time periods in which both regions are to be executed.

```
begin transient rampLoad
  begin nonlinear converge_step1
     transfer bem_to_bem_1
     advance BEM_REGION
     transfer bem_to_andante
     advance ANDANTE_REGION
    transfer andante_to_bem
   end nonlinear converge_step1
   transfer bem_to_bem2
 end transient rampLoad

 begin transient zeroLoad
   begin nonlinear converge_step2
     transfer bem_to_bem_1
     advance BEM_REGION
     transfer bem_to_bem_1
     transfer bem_to_andante
     advance ANDANTE_REGION
     transfer andante_to_bem
   end nonlinear converge_step2
   transfer bem_to_bem2
 end transient zeroLoad

 begin transient reload
   transfer bem_to_bem_1
   begin nonlinear converge_step3
     transfer bem_to_bem_1
     advance BEM_REGION
```

```
      transfer bem_to_andante
      advance ANDANTE_REGION
      transfer andante_to_bem
    end nonlinear converge_step3
    transfer bem_to_bem2
  end transient reload

end system coupled_em
```

The next block of input defines parameters to be used for each of the three time periods (TRANSIENT blocks) defined in the coupled_em system block above. Each PARAMETERS FOR TRANSIENT block specifies a start time and a termination time for this time period, and the time stepping within each region. The time periods defined by all the transient blocks must be contiguous without overlapping any time, and the final termination time equal the simulation termination time specified in the solution control. Within each TRANSIENT block are nested PARAMETERS FOR ANDANTE REGION and PARAMETERS FOR BEM REGION blocks which define the number of time steps each region will take for this time period.

```
begin parameters for transient rampLoad
  start time = 0.0
  termination time = 9.75e-6
  begin parameters for andante region ANDANTE_REGION
     time increment = .25e-6
     time integration rule = HHT
     HHT integration parameter alpha = 0.0
     HHT integration parameter beta = 0.25
     HHT integration parameter gamma = 0.5
  end parameters for andante region ANDANTE_REGION
  begin parameters for bem region BEM_REGION
     time increment = .25e-6
  end parameters for bem region BEM_REGION
end parameters for transient rampLoad

begin parameters for transient zeroLoad
  start time = 9.75e-6
  termination time = 12.0e-6
  begin parameters for andante region ANDANTE_REGION
     time increment = .25e-6
     time integration rule = HHT
     HHT integration parameter alpha = 0.0
     HHT integration parameter beta = 0.25
     HHT integration parameter gamma = 0.5
  end parameters for andante region ANDANTE_REGION
```

```
   begin parameters for bem region BEM_REGION
      time increment = .25e-6
   end parameters for bem region BEM_REGION
end parameters for transient zeroLoad

 begin parameters for transient reload
   start time = 12.0e-6
   termination time = 16.e-6
   begin parameters for andante region ANDANTE_REGION
      time increment = .25e-6
      time integration rule = HHT
      HHT integration parameter alpha = 0.0
      HHT integration parameter beta = 0.25
      HHT integration parameter gamma = 0.5
   end parameters for andante region ANDANTE_REGION
   begin parameters for bem region BEM_REGION
      time increment = .25e-6
   end parameters for bem region BEM_REGION
end parameters for transient reload
```

The final block of input lines in the solution control scope are those defining the convergence criteria to be used for each `nonlinear` iteration loop. In this example, the same convergence criteria is specified for all three `nonlinear` blocks.

The `CONVERGED WHEN "(...)"` command line contains C++ logical statements. For `NONLINEAR converge_step1` and `NONLINEAR converge_step3`, the first condition `1 < CURRENT_STEP`, specifies that this is not the initial pass through the iteration loop. This statement is necessary because convergence is checked using results of the current iteration with results of the previous iteration. The keyword `CURRENT_STEP` is syntax used in the code for solution control and is case sensitive. The second condition, `ANDANTE_REGION.norm(0.0) < .001`, states that the change in the norm of the Adagio region solution from the previous iteration to the current iteration must be less than .001 times the norm of the solution at the previous iteration to satisfy convergence. Likewise, the third condition, `BEM_REGION.norm(0.0) < .1`, sets a maximum allowed change in the norm of the BEM region solution from the previous to the current iteration. Finally, if all three of the first three conditions are not met, the last condition, `50 < CURRENT_STEP` puts a limit of 50 on the number of iterations allowed. Note that this last condition prevents an infinite loop, but the code will proceed to the next time step when the maximum number of iterations is reached, regardless whether the conditions for the convergence criteria are met.

In `NONLINEAR converge_step2`, the convergence criteria is different. During this time period, the voltage load is exactly 0.0. Therefore a single execution of the regions is all that is needed and this is obtained by specifying that convergence is met when `0 < CUR-`

```
RENT_STEP).

      begin parameters for nonlinear converge_step1
        converged when "(1 < CURRENT_STEP &&
          ANDANTE_REGION.norm(0.0) < .001 &&
          BEM_REGION.norm(0.0) < .1) ||
          (50 < CURRENT_STEP)"
      end parameters for nonlinear converge_step1


      begin parameters for nonlinear converge_step2
        converged when "(0 < CURRENT_STEP )"
      end parameters for nonlinear converge_step2

      begin parameters for nonlinear converge_step3
        converged when "(1 < CURRENT_STEP &&
          ANDANTE_REGION.norm(0.0) < .001 &&
          BEM_REGION.norm(0.0) < .1) ||
          (50 < CURRENT_STEP)"
      end parameters for nonlinear converge_step3
```

The following line signifies the end of the solution control scope.

```
    end solution control solution_control
```

The next several blocks of input lines each define a type of data transfer. Coupled electrostatic-mechanical analysis with Arpeggio requires four different types of transfer. The user-assigned name on the block is used in the NONLINEAR command blocks to execute a particular transfer. The transfers are discussed below in the order that they appear in the NONLINEAR command blocks

The first transfer that needs to be executed is one within the BEM region itself. This transfer simply copies the field e_press that was computed in the POST PROCESS block of the BEM region to a variable called previousSolution. Because e_press was computed as a face variable, this transfer uses a COPY SURFACE ELEMENTS command line. The name previousSolution is a variable used internally in solution control to test convergence and must always appear exactly as shown in this example. The states new and none will be the same for every analysis.

```
####    DEFINE TRANSFERS    #############

      begin transfer bem_to_bem_1
        copy surface elements from BEM_REGION to BEM_REGION
```

```
  send field e_press state new to previousSolution state none
end transfer bem_to_bem_1
```

The second transfer to be executed that of the electrostatic pressure from the BEM region to the Andante region. The first line in this transfer block specifies that this is a transfer of surface element (face) data from the bem region to the Andante region, and the second line specifies exactly which surfaces (sideset id's in the exodus mesh databases) are involved. In this example, the mesh for the bem region has a sideset with id 101 and the mesh for the Andante region has a sideset with id 100. The third line identifies the fields to be transferred. The user-defined name `e_press` must match the name of the field computed in the `POST PROCESS` command block in the BEM region, and the user-defined name `pressure_external_transfer` must match the name of the field in the `PRESSURE` command block in the Andante region. The state of these fields to be transferred (`new` for `e_press` and `none` for `pressure_external_transfer`) will be the same for every analysis. The last two lines define values for search tolerances as described in Section 8.3.

```
begin transfer bem_to_andante
  interpolate surface elements from BEM_REGION to ANDANTE_REGION
  send block surface_101 to surface_100
  send field e_press state new to pressure_external_transfer
    state none
  search surface gap tolerance = 1.e-10
  search geometric tolerance = 1.e-10
end transfer bem_to_andante
```

The third type of transfer is that of the displacement field computed in the Andante region to the BEM region so that the electrostatic computation can be computed on the updated, deformed geometry. Like the transfer of pressure, the transfer of displacements is a surface interpolation and the second line in the block defines the surfaces involved. For this transfer, the Andante name for the displacement field is always `displacement state new`. The BEM name for the displacement field, in this case `displacements` is chosen by the user but must match the name on the `DEFORM GEOMETRY USING TRANSFER` line command in the BEM region scope. This state of this field to be transferred is always `new`.

```
begin transfer andante_to_bem
  interpolate surface nodes from ANDANTE_REGION to BEM_REGION
  send block surface_100 to surface_101
  send field displacement state new to displacements
    state new
  search surface gap tolerance = 1.e-10
  search geometric tolerance = 1.e-10
end transfer andante_to_bem
```

The last transfer used is executed at the end of a transient time step after the nonlinear sequence has converged. It simply copies the BEM field with user-defined name `displacements` from state new to state old. Except for the user-define names, the syntax for this block will be the same for all BEM-Andante analyses. The name `displacements` must match the name given to this field on the `DEFORM GEOMETRY USING TRANSFER` command line in the BEM region scope.

```
      begin transfer bem_to_bem2
        copy volume nodes from BEM_REGION to BEM_REGION
        send field displacements state new to displacements
          state old
      end transfer bem_to_bem2
```

Finally, the last two lines signal the end of the procedure scope and then the end of the SIERRA domain scope.

```
  end procedure Argo_Procedure
end sierra dynamic_MEMS_beam
```

Below is the entire input file without interruption.

```
begin sierra dynamic_MEMS_beam
  title  Diffaction grating microbeam  \$
         in MKS units
#  Model one-half of beam for symmetry
#    Fixed at one end, x-symmetry at other end
$  Only load is electrostatic

  begin definition for function voltage
    type is piecewise linear
    begin values
      0.0      0.0
      2.e-6     100.
      8.e-6     100.
      10.e-6     0.0
      12.0e-6   0.0
      13.0e-6   60.
      15.e-6    60.0
      30.e-6   60.
    end values
  end definition for function voltage

  begin definition for function zero
```

```
    type is constant
    begin values
       0.0
    end values
  end definition for function zero

  begin property specification for material linear_elastic
    density         = 1.928e4  #kg/m^3
    begin parameters for model elastic
      youngs modulus  = 106.6e9 #Pa
      poissons ratio  = 0.4
    end parameters for model elastic
  end property specification for material linear_elastic

  begin finite element model structural
    Database Name = microbeam_struct.g  #meters
    Database Type = exodusII

    begin parameters for block block_1 block_2
      material linear_elastic
      solid mechanics use model Elastic
    end parameters for block block_1 block_2

  end finite element model structural

  begin finite element model electrical
    Database Name = microbeam_bem.g  #meters
    Database Type = exodusII
  end finite element model electrical

### BEM solver
   begin trilinos equation solver az_params
     solution method = bicgstab
     preconditioning method = none
     maximum iterations = 110
     residual norm tolerance = 1.e-10
   end

### Adagio preconditioner linear solver

  begin Aztec equation solver my_aztec
    solution method = cg
    restart iterations = 50
    preconditioning method = dd-ilut
    residual norm tolerance = 1.e-4
```

```
  END Aztec EQUATION SOLVER my_aztec

   begin feti equation solver feti
     maximum iterations = 200
     maximum orthogonalization = 500
     preconditioning method = dirichlet
     residual norm tolerance = 1.e-4
     corner algorithm = 3
     corner dimensionality = 3
     corner augmentation = none
     coarse solver = sparse
     local solver  = sparse $ iterative $ sparse
     num local subdomains = 16
   end


######  PROCEDURE INPUT  ######################

  begin procedure Argo_Procedure

###  ANDANTE REGION INPUT  ####################
     begin andante region ANDANTE_REGION
       use finite element model structural
       options = LumpedMass

       ### output description ###
       begin Results Output output_andante
         Database Name = microbeam_struct.e
         Database Type = exodusII
         At Step 0, Increment = 1
         nodal Variables = force_external as f_ext
         nodal Variables = velocity as vel
         nodal Variables = displacement as displ
         nodal Variables = residual as resid
         face Variables = pressure_external_transfer as extP
         nodal Variables = contact_normal_direction as fcnor_dir
        nodal Variables = contact_normal_traction_magnitude
            as fcnor
         nodal Variables = contact_status as cElem
         nodal Variables = contact_area as cArea
         element Variables = rotated_stress as stress
         global Variables = timestep as timestep
       end results output output_andante

       ### definition of BCs ###
```

```
### boundary conditions on beam

       begin fixed displacement
         node set = nodelist_1
         components = x y z
       end fixed displacement

       begin fixed displacement
         node set = nodelist_2
         components = x
       end fixed displacement


###  fixed ground nodes

       begin fixed displacement
         node set = nodelist_10
         components = x y z
       end fixed displacement

##  surfaces for pressure transfer
       begin pressure
         surface = surface_100
         field variable = pressure_external_transfer
       end pressure


####  Contact  ######
     begin contact definition
       enforcement = frictionless
       contact surface beam  contains surface_2
       contact surface substrate contains surface_7
       begin surface normal smoothing
         angle = 30.0
         distance = .25
         resolution = EDGE
       end
       begin interaction
         master = substrate
         slave = beam
         normal tolerance = .1e-6
         tangential tolerance = .05e-6
         capture tolerance = .05e-6
```

```
       end
     end contact definition


  ####  predictor  ######
     loadstep predictor using scale factor = 1.0, 0.0

  ####  solver     ######

  Begin adagio multilevel solver
      level 1 predictor = none
      Begin control sliding contact controlled_slide
        Target Relative Residual   = 5.e-5
        Target Residual             = 1.e-9
        Maximum Iterations = 400
        Minimum Iterations = 1
      end   control sliding contact controlled_slide

       Begin adagio solver cg
         Target Relative Residual = 1.e-6
         Reference = internal
#         Target Residual          = 5.0e-12
         Acceptable relative residual = 1.e-4
         Acceptable residual = 1.e-12
         Maximum Iterations = 800
         Minimum Iterations = 2
         Orthogonality measure for reset = .5

         Line Search type secant
         begin full tangent preconditioner
           linear solver = feti
           constraint enforcement = penalty
           penalty factor = 100
           balance probe = 1
           probe factor = .0001
           small number of iterations = 30
           maximum resets for modelproblem = 2
         end full tangent preconditioner
       end   adagio solver cg

    end   adagio multilevel solver

     end andante region ANDANTE_REGION

### BEM REGION INPUT  #############
```

```
    begin bem region BEM_REGION
      use finite element model electrical
      use dense solver = aztecoo with az_params
      deform geometry using transfer displacements

      begin be zone bem
        PDE = Laplace
        material parameter for laplace equation
             permittivity = 8.8595e-12  #N/V^2
        solution domain = external

        BC QUASISTATIC UNIFORM dirichlet at block_1 V
            using function voltage scaled by 1.0
        BC QUASISTATIC UNIFORM dirichlet at block_2 V
            using function zero scaled by 1.0

        DIRECT FORMULATION for scalar V and E_norm
            with continuous linear elements

      end be zone bem

      begin post process pressure
         process equation laplace on zone bem
         evaluate electric_pressure as e_press
         compute on surface_101 faces
      end post process pressure

      BEGIN RESULTS OUTPUT LABEL output_bem
         database Name = microbeam_bem.e
         database Type = exodusII
         at step 0, increment = 1
         title parallel plate
          nodal Variables = displacements as displ
          nodal variables = V as volt
          nodal variables = E_norm as E_norm
          face variables = e_press as e_press
      END RESULTS OUTPUT LABEL output_bem

      end bem region BEM_REGION


###       SOLUTION CONTROL       ###
    begin solution control description
```

```
use system main

begin initialize mytransient_init
  advance ANDANTE_REGION
  advance BEM_REGION
end initialize mytransient_init

begin system main

  use initialize mytransient_init

  simulation termination time = 16.e-6


end system main

begin parameters for transient rampLoad
  start time = 0.0
  termination time = 9.75e-6
  begin parameters for andante region ANDANTE_REGION
     time increment = .25e-6
     time integration rule = HHT
     HHT integration parameter alpha = 0.0
     HHT integration parameter beta = 0.25
     HHT integration parameter gamma = 0.5
  end parameters for andante region ANDANTE_REGION
  begin parameters for bem region BEM_REGION
     time increment = .25e-6
  end parameters for bem region BEM_REGION
end parameters for transient rampLoad

begin parameters for transient zeroLoad
  start time = 9.75e-6
  termination time = 12.0e-6
  begin parameters for andante region ANDANTE_REGION
     time increment = .25e-6
     time integration rule = HHT
     HHT integration parameter alpha = 0.0
     HHT integration parameter beta = 0.25
     HHT integration parameter gamma = 0.5
  end parameters for andante region ANDANTE_REGION
  begin parameters for bem region BEM_REGION
     time increment = .25e-6
  end parameters for bem region BEM_REGION
end parameters for transient zeroLoad
```

```
      begin parameters for transient reload
        start time = 12.0e-6
        termination time = 16.e-6
        begin parameters for andante region ANDANTE_REGION
           time increment = .25e-6
           time integration rule = HHT
           HHT integration parameter alpha = 0.0
           HHT integration parameter beta = 0.25
           HHT integration parameter gamma = 0.5
        end parameters for andante region ANDANTE_REGION
        begin parameters for bem region BEM_REGION
           time increment = .25e-6
        end parameters for bem region BEM_REGION
      end parameters for transient reload

      begin parameters for nonlinear converge_step1
        converged when "(1 < CURRENT_STEP
          && ANDANTE_REGION.norm(0.0) < .001
          && BEM_REGION.norm(0.0) < .1)
          ||  (50 < CURRENT_STEP)"
      end parameters for nonlinear converge_step1


      begin parameters for nonlinear converge_step2
        converged when "(0 < CURRENT_STEP )"
      end parameters for nonlinear converge_step2

      begin parameters for nonlinear converge_step3
        converged when "(1 < CURRENT_STEP
          && ANDANTE_REGION.norm(0.0) < .001
          && BEM_REGION.norm(0.0) < .1)
          ||  (50 < CURRENT_STEP)"
      end parameters for nonlinear converge_step3


    end solution control description



####    DEFINE TRANSFERS    #############


      begin transfer bem_to_andante
        interpolate surface elements from BEM_REGION to ANDANTE_REGION
```

```
        send block surface_101 to surface_100
        send field E_PRESS state new
           to pressure_external_transfer state none
        search surface gap tolerance = 1.e-10
        search geometric tolerance = 1.e-10
      end transfer bem_to_andante

      begin transfer andante_to_bem
        interpolate surface nodes from ANDANTE_REGION to BEM_REGION
        send block surface_100 to surface_101
        send field displacement state new
           to displacements state new
        search surface gap tolerance = 1.e-10
        search geometric tolerance = 1.e-10
      end transfer andante_to_bem

      begin transfer bem_to_bem_1
        copy surface elements from BEM_REGION to BEM_REGION
        send field E_PRESS state new
           to previousSolution state none
      end transfer bem_to_bem_1

      begin transfer bem_to_bem2
        copy volume nodes from BEM_REGION to BEM_REGION
        send field displacements state new
           to displacements state old
      end transfer bem_to_bem2

  end procedure Argo_Procedure

end sierra dynamic_MEMS_beam
```

## 10.4 Results

Figure 10.1 shows the displacement of the center of the beam toward the substrate as a function of time. The beam is actively loaded with a voltage pulse from 0 to 10 microseconds followed by no voltage from 10 to 12 microseconds, and then voltage re-applied starting at 12 microseconds. The flat part of the curve from 11.5 to 12.0 microseconds is where the center of the beam is in contact with the substrate located 1.6 microns below the beam.

Figue 10.2 shows the distribution of electrical pressure on the bottom of the microbeam at

8.0 microseconds. This is the time at which the applied voltage pulse peaks at 100 volts.

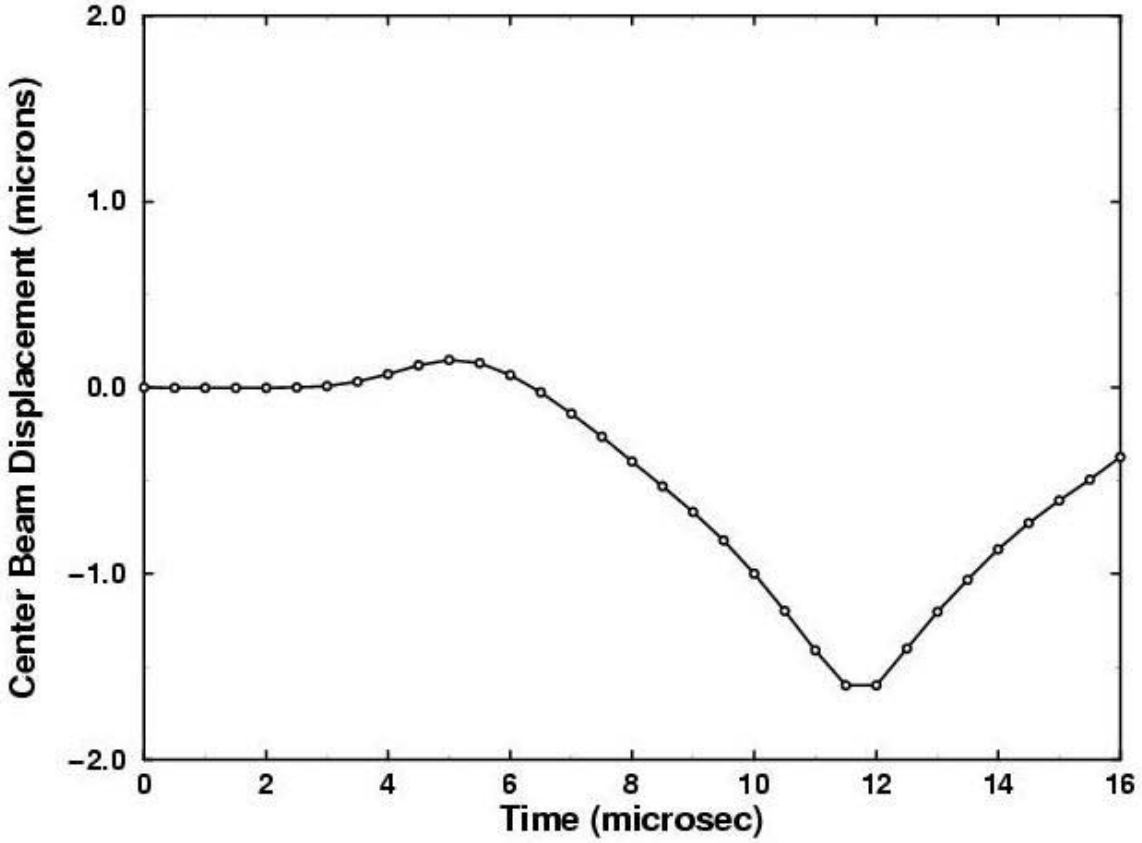


Figure 10.1: Displacement of center of microbeam.

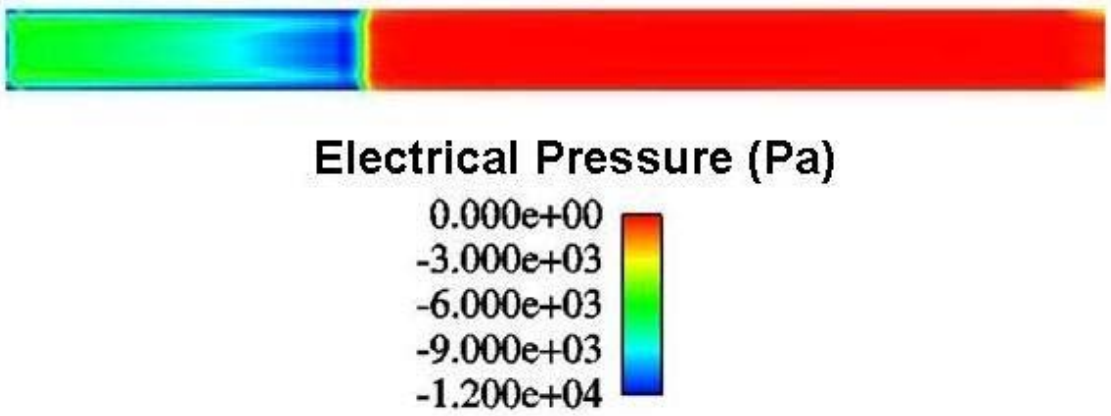


Figure 10.2: Distribution of electrical pressure at 8 microseconds.

# Appendix A

# Cubit Input Files for Mesh Generation

The first section of code below is the parameterized CUBIT journal file. The first part defines the structural volume mesh and the second part defines the electrical surface mesh. The included file, parameters.inc, gives values for the parameters whose names are enclosed within curly braces ({}). The contents of the file parameters.inc are shown below the journal file. This file will result in meshes with dimensions of micrometers.

```
{include("parameters.inc")}

create brick x  {xlen_bm} y {ylen} z {thick_bm}
move body 1 x {xlen_bm/2.} y {ylen/2.} z {thick_bm/2. + gap}

create brick x  {xlen_contact} y {ylen} z {pad_thick}
move body 2 x {-xlen_contact/2. + xlen_bm} y {ylen/2.}
     z {-pad_thick/2.+offset}

volume all size {mesh_size_st}
curve 9 10 11 12 size {.25*mesh_size_st}

volume all scheme auto

mesh volume all

nodeset 1 surface 4
nodeset 2 surface 6
nodeset 10 surface in volume 2

sideset 100 surface in volume 1
sideset 2 surface 2
```

```
sideset 7 surface 7

block 1 volume 1
block 2 volume 2

transform mesh output scale {scale}

Export Genesis 'microbeam_struct.g' overwrite

delete mesh

reset genesis

delete body 2
create brick x {xlen_el} y {ylen} z {pad_thick}
move body 3 x {xlen_el/2.} y {ylen/2.} z {-pad_thick/2.}
webcut body 1 with plane xplane offset {xlen_el+xlen_trans}

merge all

surface in vol 3 4 size {mesh_size_el}
surface in vol 1 size {4.*mesh_size_el}

curve 1 3 39 int 6
curve 9 10 38 33 34 int 1
curve 25 27 29 31  int 6

surface all scheme map
mesh surface in volume 4 except surface 19
mesh surface in volume 1 except surface 19
mesh surface in volume 3

# This line is necessary to get the element normals
# pointing inward to the beam and electrode.
surface all normal opposite

block 1 surface in volume 4 1
block 2 surface in volume 3

sideset 101 surface in volume 1 4

block 1 element type shell
block 2 element type shell

Export Genesis 'microbeam_bem.g' dimension 3 overwrite
```

```
quit
```

The contents of the included file ”parameters.inc” are shown below.

```
$ xlen_bm = {xlen_bm = 250.}
$ xlen_contact = {xlen_contact = 150.}
$ xlen_trans={xlen_trans = 20.}
$ ylen = {ylen = 10.}
$ thick_bm = {thick_bm = 2.}
$ offset = {offset = .4}
$ xlen_el = {xlen_el = 80.}

$ gap = {gap = 2.}

$ pad_thick = {pad_thick = 1.}

$ mesh_size_st = {mesh_size_st = 1.7}
  $ mesh_size_el = {mesh_size_el = 2.5}

$scale = {scale = 1.}
```

# Bibliography


[1] H. Carter Edwards. *SIERRA Framework Version 3: Core Services Theory and Design*, SAND2002-3616, Sandia National Laboratories, November 2002.

[2] Adagio Development Team, *Adagio/Andante User Reference Manual*, Sandia National Laboratories, August 2005.

[3] James Strickland and Samuel Subia, *BEM Users Manual*, Sandia National Laboratories, May 2005.

[4] Alan B. Williams. *SIERRA Framework Version 4: Solver Services*, SAND2004-6428, Sandia National Laboratories, February 2005.

[5] Alan B. Williams. *Finite Element Interface to Linear Solvers (FEI) Version 2.9: Guide and Reference Manual*, SAND2004-6430, Sandia National Laboratories, February 2005.

[6] Michael Allen Heroux, et. al. *An Overview of Trilinos*, SAND2003-2927, Sandia National Laboratories, August 2003.

[7] Hans M. Hilber, Thomas J. R. Hughes, and Robert L. Taylor, ”Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics”, *Earthquake Engineering and Structural Dynamics*, Vol. 5, 283-292, 1977.

[8] G. B. Hocker, et.al.,”The Polychromator: A Programmable MEMS Diffraction Grating for Synthetic Spectra”, Solid-State Sensor and Actuator Workshop, Hilton Head Island, South Carolina, June 4-8, 2000.

[9] C. Farhat, M. Lesoinne, P. Le Tallec, K. Pierson, and D.Rixen, ”FETI-DP: A dual-primal unified FETI method – part I: A faster alternative to the two-level FETI method”, *Int. J. Numer. Meth. Engrg.*, 50:1523–1544, 2001.

Distribution

| | | |
|---|---|---|
| 1 | 0372 | Bishop, Joseph E., 1525 |
| 1 | 0372 | Breivik, Nicole L., 1524 |
| 1 | 0372 | Dempsey, J. Franklin, 1524 |
| 1 | 0372 | Dion, Kristin, 1524 |
| 1 | 0372 | Gruda, Jeffrey D., 1524 |
| 1 | 0372 | Gwinn, Kenneth W., 1524 |
| 1 | 0372 | Hammerand, Daniel C., 1524 |
| 1 | 0372 | Harding, David C., 1524 |
| 1 | 0372 | Hinnerichs, Terry D., 1524 |
| 1 | 0372 | Holland, John F., 1524 |
| 1 | 0372 | Jones, Timothy C., 1524 |
| 1 | 0372 | Lo, Chi S., 1524 |
| 1 | 0372 | Metzinger, Kurt E., 1524 |
| 1 | 0372 | Neidigk, Matthew A., 1524 |
| 1 | 0372 | Pott, John, 1524 |
| 1 | 0372 | Rath, Jonathan S., 1524 |
| 1 | 0372 | Scherzinger, William M., 1524 |
| 1 | 0372 | Skousen, Troy J., 1524 |
| 1 | 0372 | Wellman, Gerald W., 1525 |
| 1 | 0372 | Worrel, Leah, 1524 |
| 1 | 0376 | Arguello Jr., Jose G., 1525 |
| 1 | 0376 | Stone, Charles M., 1525 |
| 1 | 0380 | Crane, Nathan K., 1542 |
| 1 | 0380 | Fortier, Harrison, 1542 |
| 1 | 0380 | Gullerud, Arne S., 1542 |
| 1 | 0380 | Hales, Jason D., 1542 |
| 1 | 0380 | Heinstein, Martin W., 1542 |
| 1 | 0380 | Jung, Joseph, 1542 |
| 1 | 0380 | Key, Samuel Witt, 1542 |
| 1 | 0380 | Koteras, James R., 1542 |
| 1 | 0380 | Morgan, Harold S., 1540 |
| 1 | 0380 | Pierson, Kendall H., 1542 |
| 5 | 0380 | Porter, Vicki L., 1542 |
| 1 | 0380 | Reese, Garth M., 1542 |
| 1 | 0380 | Spencer, Benjamin W., 1542 |
| 1 | 0380 | Walsh, Timothy F., 1542 |

1 0382 Baur, David Gregory, 1543
1 0382 Edwards, Harold C.
1 0382 Gianoulakis, Steven D., 1541
1 0382 Overfelt, James R., 1543
1 0382 Sjaardema, Gregory D., 1543
1 0382 Stewart, James R., 1543
1 0382 Subia, Samuel R., 1541
1 0382 Williams, Alan B., 1543
1 0557 Segalman, Daniel J., 1525
1 0557 Simmermacher, Todd W., 1523
1 0826 Piekos, Edward S., 1513
1 0826 Wong, Chungnin C., 1513
1 0847 Bitsie, Fernando, 1523
1 0847 Chaplya, Pavel M., 1526
1 0847 Field Jr., Richard V., 1526
1 0847 Fulcher, Clay W. G., 1526
1 0847 Redmond, James M., 1526
1 0847 Rouse, Jerry W., 1526
1 0847 Starr, Michael J., 1526
1 0886 Cox, James V., 1526
1 0886 Lavin, Colby A., 1524
1 0888 Reedy Jr., Earl David, 1526
1 1070 Massad, Jordan E., 1526
1 1070 Reu, Phillip L., 1526
1 1070 Sumali, Hartono, 1526
2 9018 Central Technical Files, 8944
2 0899 Technical Library, 4536