ANSYS Advanced Analysis Techniques Guide

ANSYS Release 8.1

001972
April 2004

ANSYS, Inc. is a
UL registered
ISO 9001: 2000
Company

ANSYS Advanced Analysis Techniques Guide

ANSYS Release 8.1

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Revision History

Number	Release	Date
001618	ANSYS 6.1	April 2002
001695*	ANSYS 7.0	October 2002
001788*	ANSYS 7.1	May 2003
001901*	ANSYS 8.0	October 2003
001972*	ANSYS 8.1	April 2004

* ANSYS Documentation on CD.

Trademark Information

ANSYS, DesignSpace, DesignModeler, ANSYS DesignXplorer VT, ANSYS DesignXplorer, ANSYS Emax, ANSYS Workbench environment, CFX, AI*Environment, CADOE and any and all ANSYS, Inc. product names referenced on any media, manual or the like, are registered trademarks or trademarks of subsidiaries of ANSYS, Inc. located in the United States or other countries.

Copyright © 2004 SAS IP, Inc. All rights reserved. Unpublished rights reserved under the Copyright Laws of the United States.

ANSYS, Inc. is a UL registered ISO 9001: 2000 Company

ANSYS Inc. products may contain U.S. Patent No. 6,055,541

Microsoft, Windows, Windows 2000 and Windows XP are registered trademarks of Microsoft Corporation.
Inventor and Mechanical Desktop are registered trademarks of Autodesk, Inc.
SolidWorks is a registered trademark of SolidWorks Corporation.
Pro/ENGINEER is a registered trademark of Parametric Technology Corporation.
Unigraphics, Solid Edge and Parasolid are registered trademarks of Electronic Data Systems Corporation (EDS).
ACIS and ACIS Geometric Modeler are registered trademarks of Spatial Technology, Inc.

"FLEXlm License Manager" is a trademark of Macrovision Corporation.

Other product and company names mentioned herein are the trademarks or registered trademarks of their respective owners.

This ANSYS, Inc. software product and program documentation is ANSYS Confidential Information and are furnished by ANSYS, Inc. under an ANSYS software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, warranties, disclaimers and remedies, and other provisions. The Program and Documentation may be used or copied only in accordance with the terms of that license agreement.

See the ANSYS, Inc. online documentation or the ANSYS, Inc. documentation CD for the complete Legal Notice.

If this is a copy of a document published by and reproduced with the permission of ANSYS, Inc., it might not reflect the organization or physical appearance of the original. ANSYS, Inc. is not liable for any errors or omissions introduced by the copying process. Such errors are the responsibility of the party providing the copy.

Table of Contents

List of Figures

List of Tables

Chapter 1: Design Optimization

Design optimization is a technique that seeks to determine an optimum design. By "optimum design," we mean one that meets all specified requirements but with a minimum expense of certain factors such as weight, surface area, volume, stress, cost, etc. In other words, the optimum design is usually one that is as effective as possible.

Virtually any aspect of your design can be optimized: dimensions (such as thickness), shape (such as fillet radii), placement of supports, cost of fabrication, natural frequency, material property, and so on. Actually, any ANSYS item that can be expressed in terms of parameters can be subjected to design optimization. (See Section 5.10.6.2: Use ANSYS Parameters in the *ANSYS Modeling and Meshing Guide* for a description of ANSYS parameters.)

The ANSYS program offers two optimization methods to accommodate a wide range of optimization problems. The *subproblem approximation* method is an advanced zero-order method that can be efficiently applied to most engineering problems. The *first order* method is based on design sensitivities and is more suitable for problems that require high accuracy.

For both the subproblem approximation and first order methods, the program performs a series of analysis-evaluation-modification cycles. That is, an analysis of the initial design is performed, the results are evaluated against specified design criteria, and the design is modified as necessary. This process is repeated until all specified criteria are met.

In addition to the two optimization techniques available, the ANSYS program offers a set of strategic tools that can be used to enhance the efficiency of the design process. For example, a number of random design iterations can be performed. The initial data points from the random design calculations can serve as starting points to feed the optimization methods mentioned above.

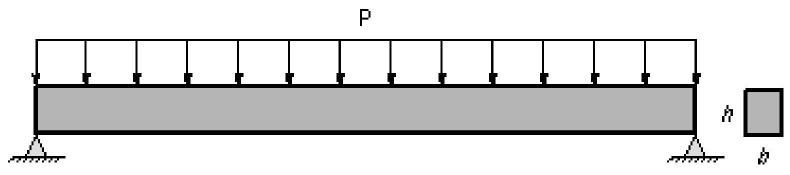
1.1. Design Optimization Terminology

To understand the terminology involved in probabilistic design, consider the following problem:

Find the minimum-weight design of a beam of rectangular cross section subject to the following constraints:

- Total stress σ should not exceed σ_{max} [$\sigma < \sigma_{max}$]
- Beam deflection Δ should not exceed Δ_{max} [$\Delta < \Delta_{max}$]
- Beam height h should not exceed h_{max} [$h < h_{max}$]

Figure 1.1 Example of a Beam for Design Optimization

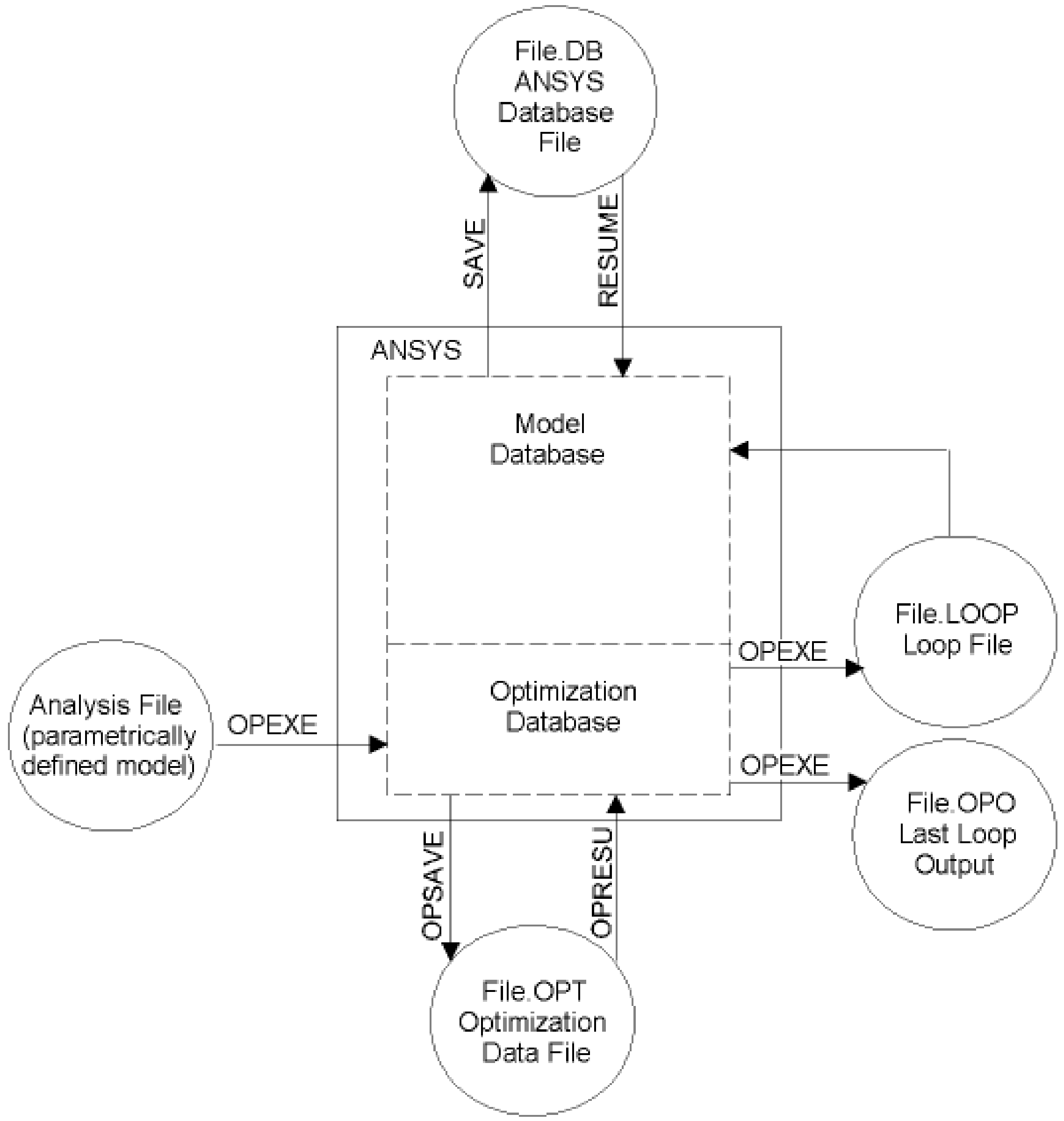


Design Optimization Term	Description
design variables (DVs)	Independent quantities, varied to achieve the optimum design. Upper and lower limits are specified to serve as "constraints" on the design variables. These limits define the range of variation for the DV. In the above beam example, width *b* and height *h* are obvious candidates for DVs. Both *b* and *h* cannot be zero or negative, so their lower limit would be *b,h* > 0.0. Also, *h* has an upper limit of h_{max}. Up to 60 DVs may be defined in an ANSYS design optimization problem.
state variables (SVs)	Quantities that constrain the design. Also known as "dependent variables," they are typically response quantities that are functions of the design variables. A state variable may have a maximum and minimum limit, or it may be "single sided," having only one limit . Our beam example has two SVs: σ (the total stress) and Δ (the beam deflection). You can define up to 100 SVs in an ANSYS design optimization problem.
objective function	The dependent variable that you are attempting to minimize. It should be a function of the DVs (that is, changing the values of the DVs should change the value of the objective function). In the beam example, the total weight of the beam could be the objective function (to be minimized). You may define only one objective function in an ANSYS design optimization problem.
optimization variables	Collectively, the design variables, state variables, and the objective function. In an ANSYS optimization, these variables are represented by user-named variables called parameters. You must identify which parameters in the model are DVs, which are SVs, and which is the objective function.
design set or *design*	A unique set of parameter values representing a given model configuration. Typically, a design set is characterized by the optimization variable values; however, all model parameters (including those not identified as optimization variables) are included in the set.
feasible design	A design that satisfies all specified constraints (those on the SVs as well as on the DVs. If *any one* of the constraints is not satisfied, the design is considered *infeasible*. The *best design* is the one which satisfies all constraints and produces the minimum objective function value. (If all design sets are infeasible, the best design set is the one closest to being feasible, irrespective of its objective function value.)
analysis file	An ANSYS input file containing a complete analysis sequence (preprocessing, solution, and postprocessing). The file must contain a parametrically defined model, using parameters to represent all inputs and outputs to be used as DVs, SVs, and the objective function.
loop file	An optimization file (named **Jobname.LOOP**), created automatically via the analysis file. The design optimizer uses the loop file to perform analysis loops.
loop	A single pass through the analysis file. Output for the last loop performed is saved in file **Jobname.OPO**. An (or simply *iteration*) is

Design Optimization Term	Description
optimization iteration	One or more analysis loops which result in a new design set. Typically, an iteration equates to one loop; however, for the first order method, one iteration represents more than one loop.
optimization database	contains the current optimization environment, which includes optimization variable definitions, parameters, all optimization specifications, and accumulated design sets. This database can be saved (to **Jobname.OPT**) or resumed at any time in the optimizer.

The following figure shows the flow of information during an optimization analysis. Note that the analysis file must exist as a separate entity, and that the optimization database is not part of the ANSYS model database.

Figure 1.2 Optimization Data Flow



1.2. Understanding GUI Paths and Command Syntax

Throughout this document, you will see references to ANSYS commands and their equivalent GUI paths. Such references use only the command name, because you do not always need to specify all of a command's arguments, and specific combinations of command arguments perform different functions. For complete syntax descriptions of ANSYS commands, consult the *ANSYS Commands Reference*.

The GUI paths shown are as complete as possible. In many cases, choosing the GUI path as shown will perform the function you want. In other cases, choosing the GUI path given in this document takes you to a menu or dialog box; from there, you must choose additional options that are appropriate for the specific task being performed.

For all types of analyses described in this guide, specify the material you will be simulating using an intuitive material model interface. This interface uses a hierarchical tree structure of material categories, which is intended to assist you in choosing the appropriate model for your analysis. See Section 1.2.4.4: Material Model Interface in the *ANSYS Basic Analysis Guide* for details on the material model interface.

1.3. Employing Design Optimization

You can approach an ANSYS optimization in two ways: as a batch run or interactively through the Graphical User Interface (GUI). The approach you take will depend on your ANSYS expertise and your preference for interacting with the ANSYS program.

If you are very familiar with ANSYS commands, you may choose to perform the entire optimization by creating an ANSYS command input file and submitting it as a batch job. This may be a more efficient method for complex analyses (for example, nonlinear) that require extensive run time.

On the other hand, the interactive features of optimization offer greater flexibility and immediate feedback for review of loop results. When performing optimization through the GUI, it is important to first establish the analysis file for your model. Then all operations within the optimizer can be performed interactively, allowing the freedom to probe your design space before the actual optimization is done. The insights you gain from your initial investigations can help to narrow your design space and achieve greater efficiency during the optimization process. (The interactive features can also be used to process batch optimization results.)

The process involved in design optimization consists of the following general steps. The steps may vary slightly, depending on whether you are performing optimization interactively (through the GUI), in batch mode, or across multiple machines.

1. Create an analysis file to be used during looping. This file should represent a complete analysis sequence and must do the following:
   - Build the model parametrically (PREP7).
   - Obtain the solution(s) (SOLUTION).
   - Retrieve and assign to parameters the response quantities that will be used as state variables and objective functions (POST1/POST26).
2. Establish parameters in the ANSYS database which correspond to those used in the analysis file; this step is typical, but not required (Begin or OPT).
3. Enter OPT and specify the analysis file (OPT).
4. Declare optimization variables (OPT).
5. Choose optimization tool or method (OPT).

6. Specify optimization looping controls (OPT).
7. Initiate optimization analysis (OPT).
8. Review the resulting design sets data (OPT) and postprocess results (POST1/POST26).

Details of the optimization process are presented below. Differences in the procedure for a "batch" versus "interactive" approach are indicated, where appropriate.

1.3.1. Create the Analysis File

The analysis file is a key component and crucial to ANSYS optimization. The program uses the analysis file to form the loop file, which is used to perform analysis loops. *Any type* of ANSYS analysis (structural, thermal, magnetic, etc.; linear or nonlinear) can be incorporated in the analysis file.

Note — An explicit dynamics analysis using ANSYS LS-DYNA cannot be optimized.

In this file, the model must be defined in terms of parameters (which are usually the DVs), and results data must be retrieved in terms of parameters (for SVs and the objective function). Only numerical scalar parameters are used by the design optimizer. See Section 5.10.6.2: Use ANSYS Parameters in the *ANSYS Modeling and Meshing Guide* for a discussion of parameters. See the *ANSYS APDL Programmer's Guide* for a discussion of the ANSYS Parametric Design Language (APDL).

It is your responsibility to create the analysis file and to verify that it is correct and complete. It must represent a clean analysis that will run from start to finish. Most nonessential commands (such as those that perform graphic displays, listings, status requests, etc.) should be stripped off or commented out of the file. Maintain only those display commands which you would like to see during an interactive session (**EPLOT**, etc.), or direct desired displays to a graphics file (**/SHOW**). Keep in mind that the analysis file will be used over and over again during optimization looping. Any commands that do not have direct bearing on the analysis will produce wasted action and therefore decrease looping efficiency.

There are two ways to create an analysis file:

- Input commands line by line with a system editor.
- Create the analysis interactively through ANSYS and use the ANSYS command log as the basis for the analysis file.

Both methods have advantages and disadvantages. Creating the file with a system editor is the same as creating a batch input file for the analysis. (If you are performing the entire optimization in batch mode, the analysis file will usually be the first portion of the complete batch input stream.) This method allows you full control of parametric definitions through exact command inputs. It also eliminates the need to clean out unnecessary commands later. However, if you are not moderately familiar with ANSYS commands, this method may be inconvenient.

You may find it easier to perform the initial analysis interactively, and then use the resulting command log as the basis for the analysis file. In this case, final editing of the log file may be required in order to make it suitable for optimization looping. (See Section 1.3.1.4: Preparing the Analysis File.)

No matter how you intend to create the analysis file, the basic information that it must contain is the same. The steps it must include are explained next.

1.3.1.1. Build the Model Parametrically

PREP7 is used to build the model in terms of the DV parameters. For our beam example, the DV parameters are *B* (width) and *H* (height), so the element real constants are expressed in terms of *B* and *H*:

```
...
/PREP7
!  Initialize DV parameters:
B=2.0                        ! Initialize width
H=3.0                        ! Initialize height
!
ET,1,BEAM3                   ! 2-D beam element
AREA=B*H                     ! Beam cross-sectional area
IZZ=(B*(H**3))/12            ! Moment of inertia about Z
R,1,AREA,IZZ,H               ! Real constants in terms of DV parameters
!
! Rest of the model:
MP,EX,1,30E6                 ! Young's modulus
N,1                          ! Nodes
N,11,120
FILL
E,1,2                        ! Elements
EGEN,10,1,-1
FINISH                       ! Leave PREP7
...
```

As mentioned earlier, you can vary virtually any aspect of the design: dimensions, shape, material property, support placement, applied loads, etc. The only requirement is that the design must be defined in terms of parameters.

The DV parameters (*B* and *H* in this example) may be initialized anywhere, but are typically defined in PREP7. The initial values assigned to these parameters represent a starting design, which is later modified by the optimizer.

> **Caution:** If you build your model interactively (through the GUI), you will encounter many situations where data can be input through graphical picking (such as when defining geometric entities). However, some picking operations do not allow parametric input. Therefore, you should avoid these picking operations when defining items that will be used as DVs, SVs, or an objective function. Instead, use menu options that allow direct input of parameters.

1.3.1.2. Obtain the Solution

The SOLUTION processor is used to define the analysis type and analysis options, apply loads, specify load step options, and initiate the finite element solution. All data required for the analysis should be specified: master degrees of freedom in a reduced analysis, appropriate convergence criteria for nonlinear analyses, frequency range for harmonic response analysis, and so on. Loads and boundary conditions may also be DVs.

The SOLUTION input for our beam example could look like this:

```
...
/SOLU
ANTYPE,STATIC             ! Static analysis (default)
D,1,UX,0,,11,10,UY        ! UX=UY=0 at the two ends of the beam
SFBEAM,ALL,1,PRES,100     ! Transverse pressure (load per unit
                          !  length) = 100
SOLVE
FINISH                    ! Leave SOLUTION
```

This step is not limited to just one analysis. You can, for instance, obtain a thermal solution and then obtain a stress solution (for thermal stress calculations).

If your solution uses the multiframe restart feature, all changes to the parameter set that are made after the first load step will be lost in a multiframe restart. To ensure that the correct parameters are used in a multiframe restart, you must explicitly save (**PARSAV**) and resume (**PARESU**) the parameters for use in the restart. See *ANSYS Basic Analysis Guide* for more information on multiframe restarts.

1.3.1.3. Retrieve Results Parametrically

This is where you retrieve results data and assign them to parameters. These parameters usually represent SVs and the objective function. The ***GET** command (**Utility Menu> Parameters> Get Scalar Data**), which assigns ANSYS calculated values to parameters, is used to retrieve the data. POST1 is typically used for this step, especially if the data are to be stored, summed, or otherwise manipulated.

In our beam example, the weight of the beam is the objective function (to be minimized). Because weight is directly proportional to volume, and assuming uniform density, minimizing the total volume of the beam is the same as minimizing its weight. Therefore, we can use volume as the objective function. The SVs for this example are the total stress and deflection. The parameters for these data may be defined as follows:

```
...
/POST1
SET,...
NSORT,U,Y               ! Sorts nodes based on UY deflection
*GET,DMAX,SORT,,MAX     ! Parameter DMAX = maximum deflection
!
! Derived data for line elements are accessed through ETABLE:
ETABLE,VOLU,VOLU        ! VOLU = volume of each element
ETABLE,SMAX_I,NMISC,1   ! SMAX_I = max. stress at end I of each
                        !  element
ETABLE,SMAX_J,NMISC,3   ! SMAX_J = max. stress at end J of each
                        !  element
!
SSUM                    ! Sums the data in each column of the element
                        !  table*GET,VOLUME,SSUM,,ITEM,VOLU! Parameter
VOLUME = total volume
ESORT,ETAB,SMAX_I,,1    ! Sorts elements based on absolute value
                        !  of SMAX_I
*GET,SMAXI,SORT,,MAX    ! Parameter SMAXI = max. value of SMAX_I
ESORT,ETAB,SMAX_J,,1    ! Sorts elements based on absolute value
                        !  of SMAX_J
*GET,SMAXJ,SORT,,MAX    ! Parameter SMAXJ = max. value of SMAX_J
SMAX=SMAXI>SMAXJ        ! Parameter SMAX = greater of SMAXI and
                        !  SMAXJ, that is, SMAX is the max. stress
FINISH
...
```

Please see the ***GET** and **ETABLE** command descriptions for more information.

1.3.1.4. Preparing the Analysis File

So far, we have described what needs to go into the analysis file. If you create this file as you would a batch input file (entering commands with a system editor), then you simply save the file and begin the optimization procedure (see Section 1.3.2: Establish Parameters for Optimization). However, if you choose to create your model interactively in ANSYS, you must derive the analysis file from the interactive session. This can be done one of two ways, using the database command log or the session log file.

Database Command Log - You can create a command log file that represents the model database by using the **LGWRITE** command (**Utility Menu> File> Write DB Log File**). **LGWRITE** writes the internal database command log to a file (**Jobname.LGW**). The internal log contains all commands that were used to create the current database.

Session Log File - **Jobname.LOG** contains all commands that are issued during an interactive session. To use **Jobname.LOG** as the optimization analysis file, you should edit out all nonessential commands with a system editor. Because all commands issued are saved to this file, extensive editing may be needed. Also, if your analysis was performed in several ANSYS sessions, you should piece together the log file segments for each session to create a complete analysis file. (See Chapter 8, “Using the ANSYS Command Log” in the *ANSYS Operations Guide* for more information on the session log file and database command log.)

Note — With either method, you may have to exit ANSYS or use the **/SYS** command in order to edit the analysis file. For more details on creating this file, see Section 1.7.1: Generating the Analysis File.

1.3.2. Establish Parameters for Optimization

At this point, having completed the analysis file, you are ready to begin optimization. (You may have to reenter ANSYS if you edited the analysis file at the system level.) When performing optimization interactively, it is advantageous (but optional) to first establish the parameters from the analysis file in the ANSYS database. (This step is not necessary for optimization performed in batch mode.)

There are two possible reasons for performing this step. The initial parameter values may be required as a starting point for a first order optimization. Also, for any type of optimization run, having the parameters in the database makes them easy to access through the GUI when defining optimization variables. To establish the parameters in the database do *one* of the following:

- Resume the database file (**Jobname.DB**) associated with the analysis file. This establishes your entire model database in ANSYS, including the parameters. To resume the database file, use one of these methods:
  Command(s): RESUME
  GUI: Utility Menu> File> Resume Jobname.db
  Utility Menu> File> Resume from
- Read the analysis file into ANSYS to perform the complete analysis. This establishes your entire model database in ANSYS, but may be time-consuming for a large model. To read the analysis file, use one of these methods:
  Command(s): /INPUT
  GUI: Utility Menu> File> Read Input from
- Resume only the parameters into ANSYS from a previously saved parameter file; that is, a parameter file that you saved using either the **PARSAV** command or the **Utility Menu> Parameters> Save Parameters** menu path. To resume the parameters, use either of these methods:
  Command(s): PARRES
  GUI: Utility Menu> Parameters> Restore Parameters
- Recreate the parameter definitions as they exist in the analysis file. Doing this requires that you know which parameters were defined in the analysis file. Use one of these methods:
  Command(s): *SET or =
  GUI: Utility Menu> Parameters> Scalar Parameters

You may choose to do none of the above, and instead depend on the **OPVAR** command (**Main Menu> Design Opt> Design Variables**) to define the parameters which you declare as optimization variables (see Section 1.3.4: Declare Optimization Variables).

Note — The ANSYS database does not need to contain model information corresponding to the analysis file to perform optimization. The model input will be read from the analysis file automatically during optimization looping.

1.3.3. Enter OPT and Specify the Analysis File

The remaining steps are performed within the OPT processor. When you first enter the optimizer, any parameters that exist in the ANSYS database are automatically established as design set number 1. It is assumed that these parameter values represent a potential design solution. To enter the optimizer, use one of these methods:
Command(s): /OPT
GUI: Main Menu> Design Opt

In interactive mode, you must specify the analysis file name. The file is used to derive the optimization loop file **Jobname.LOOP**. There is no default for the analysis file name, therefore it must be input. To specify the analysis file name, use one of these methods:

Command(s): OPANL
GUI: Main Menu> Design Opt> Analysis File> Assign

For an optimization run in batch mode, the analysis file is usually the first portion of the batch input stream, from the first line down to the first occurrence of **/OPT**. In batch mode the analysis file name defaults to **Jobname.BAT** (a temporary file containing input copied from the batch input file). Therefore, you normally do not need to specify an analysis file name in batch mode. However, if for some reason you have separated the batch optimization input into two files (one representing the analysis and the other containing all optimization operations), then you will need to specify the analysis file (**OPANL**) after entering the optimizer (**/OPT**).

Note — In the analysis file, the **/PREP7** and **/OPT** commands must occur as the first nonblank characters on a line (that is, do not use the $ delimiter on a line containing either of these commands). This is required for proper loop file construction.

1.3.4. Declare Optimization Variables

The next step is to declare optimization variables, that is, specify which parameters are DVs, which ones are SVs, and which one is the objective function. As mentioned earlier, up to 60 DVs and up to 100 SVs are allowed, but only one objective function is allowed. To declare optimization variables, use one of these methods:

Command(s): OPVAR
GUI: Main Menu> Design Opt> Design Variables
Main Menu> Design Opt> State Variables
Main Menu> Design Opt> Objective

Minimum and maximum constraints can be specified for SVs and DVs. No constraints are needed for the objective function. Each variable has a tolerance value associated with it, which you may input or let default to a program calculated value.

If the parameter name which you specify on the **OPVAR** command is not an existing parameter, the parameter is automatically defined in the ANSYS database with an initial value of zero.

You may change a previously declared optimization variable at any time by simply redefining it. You may also delete an optimization variable (**OPVAR**,*Name*,DEL). The delete option does not delete the parameter; it simply deactivates the parameter as an optimization variable (see Section 1.7.3: Modifying the Optimization Variables After Execution).

1.3.5. Choose Optimization Tool or Method

In the ANSYS program, several different optimization tools and methods are available. Single loop is the default. To specify a tool or method to be used for subsequent optimization looping, use one of these methods:

Command(s): OPTYPE
GUI: Main Menu> Design Opt> Method/Tool

Optimization methods are traditional techniques that strive for minimization of a single function (the objective function) subject to constraints. Two methods are available: the subproblem approximation method and the first order method. In addition, you can supply an external optimization algorithm, in which case the ANSYS algorithm will be bypassed. To use one of these methods, you must have an objective function defined.

- *Subproblem Approximation Method*: This is an advanced zero-order method which uses approximations (curve fitting) to all dependent variables (SVs and the objective function). It is a general method that can be applied efficiently to a wide range of engineering problems.
- *First Order Method*: This method uses derivative information, that is, gradients of the dependent variables with respect to the design variables. It is highly accurate and works well for problems having dependent variables that vary widely over a large range of design space. However, this method can be computationally intense.
- *User-supplied Method*: An external optimization routine (USEROP) can be used instead of the ANSYS optimizer logic.

Optimization tools are techniques used to measure and understand the design space of your problem. Since minimization may or may not be a goal, an objective function is not required for use of the tools. However, design variables must be defined. The following tools are available.

- *Single Loop Run*: This tool performs one loop and produces one FEA solution at a time. You can do "what if" studies by using a series of single loop runs, setting different design variable values before each loop.
- *Random Design Generation*: Multiple loops are performed, with random design variable values at each loop. A maximum number of loops and a desired number of feasible loops can be specified. This tool is useful for studying the overall design space, and for establishing feasible design sets for subsequent optimization analysis.
- *Sweep Generation*: Starting from a reference design set, this tool generates several sequences of design sets. Specifically, it varies one design variable at a time over its full range using uniform design variable increments. This tool makes global variational evaluations of the objective function and of the state variables possible.
- *Factorial Evaluation*: This is a statistical tool that is used to generate design sets at all extreme combinations of design variable values. This technique is related to the technology known as *design of experiment* that uses a 2-level, full and fractional factorial analysis. The primary aim is to compute main and interaction effects for the objective function and the state variables.
- *Gradient Evaluation*: At a user-specified reference design set, this tool calculates the gradients of the objective function and the state variables with respect to the design variables. Using this tool, you can investigate local design sensitivities.
- *User-supplied Design Tool*: An external routine (USEROP) can be used to bypass the ANSYS logic.

As noted above, you can implement your own method or tool by invoking the USEROP routine. The *Guide to ANSYS User Programmable Features* contains more details on this user-supplied routine.

1.3.6. Specify Optimization Looping Controls

Each method and tool has certain looping controls associated with it, such as maximum number of iterations, etc. All of the commands that you use to set these controls are accessed by the menu path **Main Menu> Design Opt> Method/Tool**

The commands for setting controls are as follows:

- To set controls for the subproblem approximation method, use **OPSUBP** and **OPEQN**.
- To set controls for the first order method, use **OPFRST**.
- To set controls for the random design generation tool, use **OPRAND**.
- To set controls for the sweep generation tool, use **OPSWEEP**.
- To set controls for the factorial evaluation tool, use **OPFACT**.

- To set controls for the gradient evaluation tool, use **OPGRAD**.
- To set controls for the user optimization tool, use **OPUSER**.

There are also a number of general controls which affect how data is saved during optimization. They are as follows:

- To specify the file where optimization data is to be saved (defaults to **Jobname.OPT**):
  Command(s): OPDATA
  GUI: Main Menu> Design Opt> Controls
- To activate a detailed summary printout:
  Command(s): OPPRNT
  GUI: Main Menu> Design Opt> Controls
- To determine whether information from the best design set is saved (by default, the database and results files are saved only for the last design set):
  Command(s): OPKEEP
  GUI: Main Menu> Design Opt> Controls

You can also control various looping characteristics, including how the analysis file is to be read for looping. The file can be read starting from the first line (default) or from the first occurrence of **/PREP7**, and parameters assigned as DVs can be ignored (default) or processed during looping. In addition, you can specify which type of parameters are to be saved during looping: scalar parameters only (default), or scalar and array parameters. This capability allows for control of parameter values (DV and non-DV) during looping. To control these looping characteristics, use one of these methods:
Command(s): OPLOOP
GUI: Main Menu> Design Opt> Controls

Note — The `Parms` argument on the **OPLOOP** command controls which parameters are saved during looping. The option to save both scalar and array parameters (`Parms` = ALL) should typically not be used, except for the case when array parameters defined outside of the analysis file (***DIM**) need to be preserved during looping.

1.3.7. Initiate Optimization Analysis

After all appropriate controls have been specified, you can initiate looping:
Command(s): OPEXE
GUI: Main Menu> Design Opt> Run

Upon execution of **OPEXE**, an optimization loop file (**Jobname.LOOP**) is written from the analysis file. This loop file, which is transparent to the user, is used by the optimizer to perform analysis loops. Looping will continue until convergence, termination (not converged, but maximum loop limit or maximum sequential infeasible solutions encountered), or completion (for example, requested number of loops for random design generation) has been reached.

If a loop is interrupted due to a problem within the model (for example, a meshing failure, a non-converged nonlinear solution, conflicting design variable values, etc.), the optimizer aborts that loop, but can continue looping. In interactive mode, a warning will be issued, and you may choose to continue or terminate looping. In batch mode, looping will automatically continue. (The **NCNV** command (menu path **Main Menu> Solution> Analysis Type> Sol'n Control**:Advanced NL Tab, **Main Menu> Solution> Unabridged Menu> Nonlinear> Criteria to Stop**, or **Main Menu> Solution> Load Step Opts> Nonlinear> Criteria to Stop**), which specifies program termination controls for nonlinear analyses, is ignored during optimization looping.) The design set for the aborted loop will be saved, but the response quantity parameters for that set will have very large, irrelevant values.

The values of all optimization variables and other parameters at the end of each iteration are stored on the optimization data file (**Jobname.OPT**). Up to 130 such sets are stored. When the 130th set is encountered, the data associated with the "worst" design are discarded.

Continuing with our beam example, the optimization portion of the input would look like this:

```
/OPT                        ! Enter OPT
OPANL,...                   ! Analysis file name (not needed for batch)
!
! Declare optimization variables:
OPVAR,B,DV,.5,16.5          ! Parameters B and H are DVs
OPVAR,H,DV,.5,8
OPVAR,DMAX,SV,-0.1,0        ! Parameters DMAX and SMAX are SVs
OPVAR,SMAX,SV,0,20000
OPVAR,VOLUME,OBJ            ! Parameter VOLUME is the obj. function
!
! Specify optimization type and controls
OPTYPE,SUBP                 ! Subproblem approximation method
OPSUBP,30                   ! Maximum number of iterations
OPEXE                       ! Initiate optimization looping
```

Several optimization executions may occur in series. For example, we could perform a sweep generation after the subproblem approximation execution is completed. The following series of commands executes a sweep with respect to the *best* design set:

```
OPTYPE,SWEEP                       ! Sweep evaluation tool
OPSWEEP,BEST,5                     ! 5 evaluations per DV at best design set
OPEXE                              ! Initiate optimization looping
```

See the **/OPT**, **OPANL**, **OPVAR**, **OPTYPE**, **OPSUBP**, **OPSWEEP**, and **OPEXE** command descriptions for more information.

1.3.8. Review Design Sets Data

After optimization looping is complete, you can review the resulting design sets in a variety of ways using the commands described in this section. These commands can be applied to the results from any optimization method or tool.

To list the values of parameters for specified set numbers:

Command(s): OPLIST
GUI: Main Menu> Design Opt> Design Sets> List

You can choose to list all scalar parameters, or only those used as optimization variables.

To graph specified parameters versus set number so you can track how a variable changed from iteration to iteration:

Command(s): PLVAROPT
GUI: Main Menu> Design Opt> Graphs/Tables

To change the abscissa of the graph from set number to any other parameter:

Command(s): XVAROPT
GUI: Main Menu> Design Opt> Graphs/Tables

To print the values of specified parameters in tabular form (versus the **XVAROPT** parameter, which defaults to set number):

Command(s): PRVAROPT
GUI: Main Menu> Design Opt> Graphs/Tables

For the **PLVAROPT** and **PRVAROPT** operations, the design sets are automatically sorted in a sequence corresponding to an ascending order of the **XVAROPT** parameter.

There are several specialized ways to review results from the sweep, factorial, and gradient tools. For sweep tools, use the **OPRSW** command to list results and the **OPLSW** command to plot results. For factorial tools, use the **OPRFA** command to list results and the **OPLFA** command to plot results. For gradient tools, use the **OPRGR** command to list results and the **OPLGR** command to plot results. (Menu paths appear in detailed discussions of these commands later in this chapter.)

Another way to access optimization data is with the **STAT** command (**Main Menu> Design Opt> Opt Database> Status**). When issued within the optimizer, this command lists other current optimization information such as the analysis file name, specified optimization technique, number of existing design sets, optimization variables, etc. Using the **STAT** command is a good way to check the current optimization environment (at any point in the optimizer) and to verify that all desired settings have been input.

In addition to reviewing the optimization data, you may wish to postprocess the analysis results using POST1 or POST26. By default, results are saved for the last design set in file **Jobname.RST** (or **.RTH**, etc., depending on the type of analysis). The results and the database for the best design set will also be available if **OPKEEP**,ON was issued before looping. The "best results" will be in file **Jobname.BRST** (**.BRTH**, etc.), and the "best database" will be in **Jobname.BDB**.

1.3.8.1. Manipulating Designs Sets

After reviewing the design sets, it may be desirable to manipulate them in some way. For example, after performing a random design execution, you may wish to discard all non-feasible designs, keeping the feasible sets as data points for subsequent optimization. There are several ways in which you can change the design sets.

Two commands are available for discarding unwanted sets.

- To select a number of best design sets, or all feasible sets:
  Command(s): OPSEL
  GUI: Main Menu> Design Opt> Design Sets> Select/Delete

(All design sets not selected with **OPSEL** are permanently removed from the optimization database.)

- To delete the design sets in a specified range, use one of these methods:
  Command(s): OPDEL
  GUI: Main Menu> Design Opt> Design Sets> Select/Delete

For both of these commands, the original set numbers will be retained for the remaining design sets. (Up to 130 design sets can be stored in the optimization database.)

There are other commands that can affect design sets.

- To form a new design set by adding two existing sets (with scaling factors if desired):
  Command(s): OPADD
  GUI: Main Menu> Design Opt> Design Sets> Combine
- To create a new design set using the active scalar parameter values (without running an analysis loop):
  Command(s): OPMAKE
  GUI: Main Menu> Design Opt> Analysis File> Create

1.4. Multiple Optimization Executions

There are various reasons why you might wish to do more than one optimization execution. For example, your initial optimization run may not find the desired optimum. Or, you may start by using a design tool and then perform a subsequent optimization (for example, random design generation followed by a subproblem approximation run). The knowledge you gain from the first few runs may prompt you to change your design space and optimize yet again.

If you perform all executions within the same ANSYS session (or within the same batch input stream), the procedure is very straightforward. After an execution, simply redefine all optimization input as desired and initiate the next execution. To initiate the next execution:

Command(s): OPEXE
GUI: Main Menu> Design Opt> Run

If you have left the ANSYS program after performing optimization, and would like to continue that optimization analysis at some later time, you can do a restart as described next.

1.4.1. Restarting an Optimization Analysis

To restart an optimization analysis, simply resume the optimization database file (**Jobname.OPT**):

Command(s): OPRESU
GUI: Main Menu> Design Opt> Opt Database> Resume

Once the data is read in, you can respecify optimization type, controls, etc., and initiate looping. (The analysis file corresponding to the resumed database must be available in order to perform optimization.) To initiate looping:

Command(s): OPEXE
GUI: Main Menu> Design Opt> Run

A typical restart might look like this:

```
....
/OPT
OPRESU,....          ! Read named file (defaults to Jobname.OPT)
OPSEL,10             ! Select 10 best designs
OPTYPE,....          ! Specify optimization tool or method
....                 ! Specify other optimization input
....
OPEXE                ! Initiate optimization looping
```

See the **/OPT**, **OPRESU**, **OPSEL**, **OPTYPE**, and **OPEXE** command descriptions for more details.

Note — In addition to optimization data, the ANSYS jobname is saved to the optimization database file (**Jobname.OPT**). Therefore, when an optimization data file is resumed (**OPRESU**), the jobname saved in that file overwrites the current jobname (**/FILNAME**).

You can use the **OPRESU** command (**Main Menu> Design Opt> Opt Database> Resume**) in an interactive session to resume optimization data that was created through a batch run, thus allowing convenient interactive viewing of batch optimization results.

If there is data in the optimization database at the time you want to resume, you should first clear the optimization database. When you do this, all settings are reset to their default values, and all design sets are deleted. To clear the optimization database:

Command(s): OPCLR
GUI: Main Menu> Design Opt> Opt Database> Clear & Reset

Because the ANSYS database is not affected by the **OPCLR** command, it may also be necessary to clear the ANSYS database if the resumed optimization problem is totally independent of the previous one. To clear the ANSYS database:

Command(s): /CLEAR
GUI: Utility Menu> File> Clear & Start New

A counterpart to the **OPRESU** command is the **OPSAVE** command (**Main Menu> Design Opt> Opt Database> Save**), which writes optimization data to a named file (defaults to **Jobname.OPT**). Although optimization data is automatically saved at the end of each optimization loop (see the **OPDATA** command), you can save the optimization data at any time by using the **OPSAVE** command.

1.5. Optimization Techniques

Understanding the algorithm used by a computer program is always helpful, and is particularly true in the case of design optimization. In this section, details on the following techniques are presented: subproblem approximation, first order, random design generation, sweep generation, factorial evaluation, and gradient evaluation. See the *ANSYS, Inc. Theory Reference* for more information.

1.5.1. Subproblem Approximation Method

The subproblem approximation method can be described as an advanced zero-order method in that it requires only the values of the dependent variables, and not their derivatives. There are two concepts that play a key role in the subproblem approximation method: the use of *approximations* for the objective function and state variables, and the *conversion* of the constrained optimization problem *to an unconstrained problem.*

1.5.1.1. Approximations

For this method, the program establishes the relationship between the objective function and the DVs by curve fitting. This is done by calculating the objective function for several sets of DV values (that is, for several designs) and performing a least squares fit between the data points. The resulting curve (or surface) is called an *approximation*. Each optimization loop generates a new data point, and the objective function approximation is updated. It is this approximation that is minimized instead of the actual objective function.

State variables are handled in the same manner. An approximation is generated for each state variable and updated at the end of each loop.

You can control curve fitting for the optimization approximations. You can request a linear fit, quadratic fit, or quadratic plus cross terms fit. By default, a quadratic plus cross terms fit is used for the objective function, and a quadratic fit is used for the SVs. To control curve fitting:

Command(s): OPEQN
GUI: Main Menu> Design Opt> Method/Tool

OPEQN also gives you control over how the available design data points are weighted in forming the approximations. See the *ANSYS, Inc. Theory Reference* for details.

1.5.1.2. Conversion to an Unconstrained Problem

State variables and limits on design variables are used to constrain the design and make the optimization problem a *constrained* one. The ANSYS program converts this problem to an *unconstrained* optimization problem because minimization techniques for the latter are more efficient. The conversion is done by adding *penalties* to the objective function approximation to account for the imposed constraints.

The search for a minimum of the unconstrained objective function approximation is then carried out by applying a *Sequential Unconstrained Minimization Technique* (SUMT) at each iteration.

1.5.1.3. Convergence Checking

At the end of each loop, a check for convergence (or termination) is made. The problem is said to be *converged* if the current, previous, or best design is feasible and *any* of the following conditions are satisfied:

- The change in objective function from the best feasible design to the current design is less than the objective function tolerance.
- The change in objective function between the last two designs is less than the objective function tolerance.
- The changes in *all* design variables from the current design to the best feasible design are less then their respective tolerances.
- The changes in *all* design variables between the last two designs are less than their respective tolerances.

You specify the objective function and design variable tolerances using one of these methods:
Command(s): OPVAR
GUI: Main Menu> Design Opt> Design Variables
Main Menu> Design Opt> Objective

Convergence does not necessarily indicate that a true global minimum has been obtained. It only means that one of the four criteria mentioned above has been satisfied. Therefore, it is your responsibility to determine if the design has been sufficiently optimized. If not, you can perform additional optimization analyses.

Sometimes the solution may *terminate* before convergence is reached. This happens if one of the following conditions is true:

- The number of loops specified (*NITR* on the **OPSUBP** command) has been performed.
- The number of consecutive infeasible designs has reached the specified limit (`NINFS` on the **OPSUBP** command). The default number is 7.

1.5.1.4. Special Considerations for Subproblem Approximation

Because approximations are used for the objective function and SVs, the optimum design will be only as good as the approximations. Guidelines to help establish "good" approximations are presented below.

For subproblem approximation, the optimizer initially generates random designs to establish the state variable and objective function approximations. Because these are random designs, convergence may be slow. You can sometimes speed up convergence by providing more than one feasible starting design. Simply run a number of random designs and discard all infeasible designs. To run a number of random designs:
Command(s): OPTYPE,RAND
GUI: Main Menu> Design Opt> Method/Tool

To discard all infeasible designs, use one of these methods:
Command(s): OPSEL
GUI: Main Menu> Design Opt> Design Sets> Select/Delete

Alternatively, you could create the initial design sets by using multiple single loop runs, specifying a new set of acceptable design variables before each run:
Command(s): OPTYPE,RUN
GUI: Main Menu> Design Opt> Method/Tool

(This latter method works best if you have some insight into the nature of your problem.)

> *Note* — Generating many trial designs may be good for the rate of convergence, but if the designs are very similar to each other, that is, if the design data points are "clustered" together, you may be forcing the optimizer to follow a specific path, thereby missing potentially good designs.

If many infeasible designs are being generated by the subproblem approximation method, it may mean that the state variable approximation does not adequately represent the actual state variable function. In that case, you can do the following:

- Increase the allowable number of consecutive infeasible designs and perform an additional subproblem approximation execution (if it appears likely that a feasible design will be obtained):
  Command(s): OPSUBP,,`NINFS`
  GUI: Main Menu> Design Opt> Method/Tool
- Periodically select only the best designs between sequential subproblem approximation runs to force better curve fit approximations:
  Command(s): OPSEL
  GUI: Main Menu> Design Opt> Design Sets> Select/Delete
- Choose cross terms for the state variable approximations:
  Command(s): OPEQN,,`KFSV`
  GUI: Main Menu> Design Opt> Method/Tool

1.5.2. First Order Method

Like the subproblem approximation method, the first order method converts the problem to an unconstrained one by adding penalty functions to the objective function. However, unlike the subproblem approximation method, the actual finite element representation is minimized and not an approximation.

The first order method uses gradients of the dependent variables with respect to the design variables. For each iteration, gradient calculations (which may employ a steepest descent or conjugate direction method) are performed in order to determine a search direction, and a line search strategy is adopted to minimize the unconstrained problem.

Thus, each iteration is composed of a number of subiterations that include search direction and gradient computations. That is why one optimization iteration for the first order method performs several analysis loops.

The **OPFRST** command (**Main Menu> Design Opt> Method/Tool**) has two input fields which may be used to enhance convergence of the first order solution. You can specify the forward difference applied to the design variable range used to compute the gradient (`DELTA`), and also the limit on the line search step size (`SIZE`). Typically, the default values for these two inputs are sufficient. See the *ANSYS, Inc. Theory Reference* for details.

1.5.2.1. Convergence Checking

First order iterations continue until either convergence is achieved or termination occurs. The problem is said to be converged if, when comparing the current iteration design set to the previous and best sets, one of the following conditions is satisfied:

- The change in objective function from the best design to the current design is less than the objective function tolerance.
- The change in objective function from the previous design to the current design is less than the objective function tolerance.

It is also a requirement that the final iteration used a steepest descent search, otherwise additional iterations are performed.

To specify the objective function tolerance:

Command(s): OPVAR
GUI: Main Menu> Design Opt> Objective

The problem may terminate before convergence is reached. This occurs if the number of iterations specified by *NITR* on the **OPFRST** command has been performed.

1.5.2.2. Special Considerations for the First Order Method

Compared to the subproblem approximation method, the first order method is seen to be more computationally demanding and more accurate. However, high accuracy does not always guarantee the best solution. Here are some situations to watch out for:

- It is possible for the first order method to converge with an infeasible design. In this case, it has probably found a local minimum, or there is no feasible design space. If this occurs, it may be useful to run a subproblem approximation analysis (**OPTYPE**,SUBP), which is a better measure of full design space. Also, you may try generating random designs (**OPTYPE**,RAND) to locate feasible design space (if any exists), then rerun the first order method using a feasible design set as a starting point.
- The first order method is more likely to hit a local minimum (see Section 1.7.4: Local Versus Global Minimum). This is because first order starts from one existing point in design space and works its way to the minimum. If the starting point is too near a local minimum, it may find that point instead of the global minimum. If you suspect that a local minimum has been found, you may try using the subproblem approximation method or random design generation, as described above.
- An objective function tolerance that is too tight may cause a high number of iterations to be performed. Because this method solves the actual finite element representation (not an approximation), it will strive to find an exact solution based on the given tolerance.

1.5.3. Random Design Generation

For random design generation (**OPTYPE**,RAND), the program performs a specified number of analysis loops using random design variable values for each loop. You can use the **OPRAND** command (**Main Menu> Design Opt> Method/Tool**) to specify a maximum number of iterations and (if desired) a maximum number of feasible designs. If a number of feasible design sets is specified, looping will terminate when that number is reached, even if the maximum number of iterations has not been reached.

Random design generation is often used as a precursor to a subproblem approximation optimization (as explained earlier). It can also be used to develop trial designs for a variety of purposes. For example, a number of random designs can be generated, then reviewed in order to judge the validity of the current design space.

1.5.4. Using the Sweep Tool

The sweep tool (**OPTYPE**,SWEEP) is used to perform a global sweep of design space. Exactly *n***NSPS* design sets are generated, where *n* is the number of design variables and *NSPS* is the number of evaluation points per sweep (specified on the **OPSWEEP** command). For each design variable, the range of the variable is divided into *NSPS*-1 equal increments, and *NSPS* loops are performed. The DV in question is incremented for each loop, while the remaining design variables are held fixed at their reference values. The DV reference values correspond to the design set specified by *Dset* on the **OPSWEEP** command (**Main Menu> Design Opt> Method/Tool**).

To graph response variable values versus design variable values, use one of these methods:

Command(s): OPLSW
GUI: Main Menu> Design Opt> Tool Results> Graph> Sweeps

The vertical axis shows actual values for the objective function or state variable. The horizontal axis shows normalized values (0 to 1) for the design variables, where the normalization is with respect to the DV maximum and minimum values (**OPVAR**).

To generate tabular results, use one of these methods:
Command(s): OPRSW
GUI: Main Menu> Design Opt> Tool Results> Print

Normalized response values are tabulated against normalized (0 to 1) design variables. The objective function and state variable results are normalized to the values associated with the reference design set (**OPSWEEP**,`Dset`). For the design variables, 0 corresponds to its minimum value and 1 to its maximum.

1.5.5. Using the Factorial Tool

The factorial tool (**OPTYPE**,FACT) employs a 2-level technique to generate design set results at all extreme points (or corners) of design space. (The 2-level technique samples the 2 extreme points of each DV.) Either a full or fractional evaluation will be performed, as specified by the **OPFACT** command (**Main Menu> Design Opt> Method/Tool**). For a full evaluation, the program performs 2^n loops, where *n* is the number of design variables. A 1/2 fractional factorial evaluation will perform $2^n/2$ loops; a 1/4 fractional factorial evaluation will perform $2^n/4$ loops; etc.

You can display graphics in the form of bar charts and generate tables that show certain effects for either the objective function or any state variable. For example, you may request a graph of the main effect that each design variable has on the objective function. You can also see the effects associated with 2- and 3-variable interactions.

To display graphics in the form of bar charts, use one of these methods:
Command(s): OPLFA
GUI: Main Menu> Design Opt> Tool Results> Graph> Factorial

To generate tables that show effects for the objective function or any state variable, use one of these methods:
Command(s): OPRFA
GUI: Main Menu> Design Opt> Tool Results> Print

See the *ANSYS, Inc. Theory Reference* for more information.

1.5.6. Using the Gradient Evaluation Tool

The gradient tool (**OPTYPE**,GRAD) computes a gradient at a point in design space. Gradient results are useful for studying the sensitivities of the objective function or state variables. To identify the design set where the gradient is to be computed:
Command(s): OPGRAD
GUI: Main Menu> Design Opt> Method/Tool

The number of loops performed for this tool equals the number of design variables.

You can graph response variables with respect to design variable values. The vertical axis shows actual values for the objective function or state variable graphed. The horizontal axis shows a plus or minus 1% change in the DVs. To graph response variables:
Command(s): OPLGR
GUI: Main Menu> Design Opt> Tool Results> Graph> Gradient

You can also generate tabular results for the objective function and the state variables. Changes in objective function and state variable values are tabulated against plus or minus 1% changes in the design variables. To generate these tabular results:

Command(s): OPRGR
GUI: Main Menu> Design Opt> Tool Results> Print

Note — The 1% change in the DVs is with respect to the range of the DV (MAX- MIN value from the **OPVAR** command) and, therefore, is *not* based on the current DV values.

1.6. Guidelines on Choosing Optimization Variables

There are many useful guidelines you can follow in defining your DVs, SVs, and objective function. Some of these are presented below.

1.6.1. Choosing Design Variables

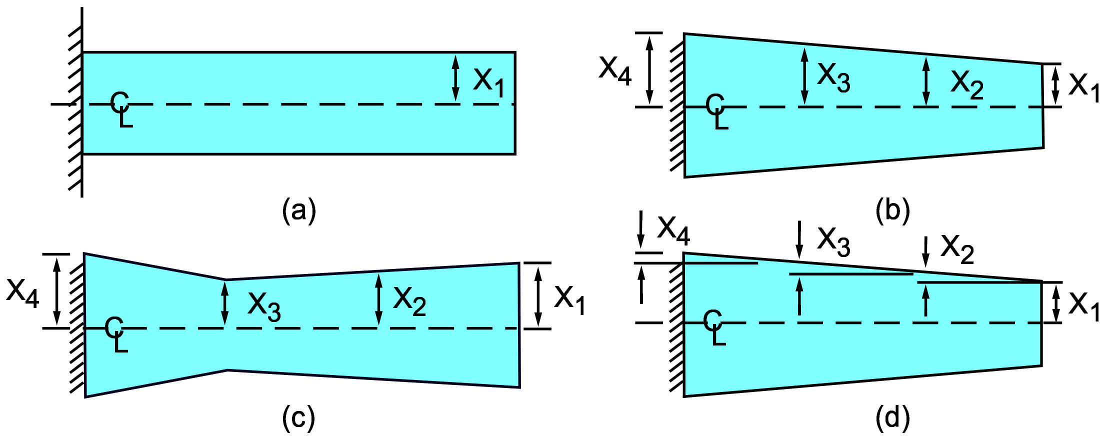
DVs are usually *geometric* parameters such as length, thickness, diameter, or model coordinates. They are restricted to *positive values*. Some points to remember about DVs are:

- Use as few DVs as possible. Having too many design variables increases the chance of converging to a local minimum rather than the true global minimum, or even diverging if your problem is highly nonlinear. Obviously, more DVs demand more iterations and, therefore, more computer time. One way to reduce the number of design variables is to eliminate some DVs by expressing them in terms of others, commonly referred to as design variable linking.

DV linking may not be practical if the DVs are truly independent. However, it may be possible to make judgements about your structure's behavior which allow a logical link between some DVs. For example, if it is thought that an optimum shape will be symmetric, use one DV for symmetric members.

- Specify a reasonable range of values for the design variables (`MIN` and `MAX` on the **OPVAR** command). Too wide a range may result in poor representation of design space, whereas too narrow a range may exclude "good" designs. Remember that only positive values are allowed, and that an upper limit *must* be specified.
- Choose DVs such that they permit practical optimum designs. For example, you can perform weight optimization of a cantilever beam with just one design variable, X1, as shown in Figure 1.3: "Choosing DVs for a Tapered Cantilever Beam" (a). However, this excludes a tapered or curved design that may offer a lower weight. To allow for such a design, you may choose four design variables X1 to X4 (Figure b), but that may result in an undesirable local minimum (Figure c). A different scheme for such a situation would be to relate the DVs as shown in Figure d. Also, avoid choosing DVs that may result in unrealistic or undesirable designs.

Figure 1.3 Choosing DVs for a Tapered Cantilever Beam



1.6.2. Choosing State Variables

SVs are usually response quantities that constrain the design. Examples of SVs are stresses, temperatures, heat flow rates, frequencies, deflections, absorbed energy, elapsed time, and so on. A state variable need not be an ANSYS-calculated quantity; virtually any parameter can be defined as a state variable. Some points to keep in mind while choosing state variables are:

- When defining SVs (**OPVAR** command), a blank input in the *MIN* field is interpreted as "no lower limit." Similarly, a blank in the *MAX* field is interpreted as "no upper limit." A zero input in either of these fields is interpreted as a zero limit. Example:

```
OPVAR,SIG,SV,,1000      ! SIG<=1000
OPVAR,SIG,SV,0,1000     ! 0<=SIG<=1000
```

- Choose enough SVs to sufficiently constrain the design. In a stress analysis, for example, choosing the maximum stress as the *only* SV may not be a good idea because the location of the maximum stress may change from loop to loop. Also avoid the other extreme which would be to choose the stress in *every* element as a state variable. The preferred method is to define the stresses at a few key locations as state variables.
- For the subproblem approximation method, if possible, choose SVs that have a linear or quadratic relationship with the DVs. For example, the state variable G = Z1/Z2 subject to G < C (where Z1 and Z2 are design variables and C is a constant) may not lead to a good approximation for G because of its inverse relationship with Z2. By restating the state variable to be G = Z1 - (C x Z2) subject to G < 0, the state variable approximation will be exact.
- If a state variable has both an upper and lower limit, specify a reasonable range of limit values (*MIN* and *MAX* on the **OPVAR** command). Avoid very small ranges, because feasible designs may not exist. A stress range of 500 to 1000 psi, for example, is better than 900 to 1000 psi.
- If an *equality constraint* is to be specified (such as frequency = 386.4 Hz), define two state variables for the same quantity and bracket the desired value, as illustrated below:

```
...
*GET,FREQ,ACTIVE,,SET,FREQ       ! Parameter FREQ = calculated frequency
FREQ1=FREQ
FREQ2=FREQ
```

```
...
/OPT
OPVAR,FREQ1,SV,,387             ! Upper limit on FREQ1 = 387
OPVAR,FREQ2,SV,386              ! Lower limit on FREQ2 = 386
```

The effective feasible region is now between 386 and 387, but each state variable has a wide enough range for smooth approximations. (Please see the **OPVAR** command description for more information.)

- Avoid choosing SVs near singularities (such as concentrated loads) by using selecting before defining the parameters.

1.6.3. Choosing the Objective Function

The objective function is the quantity that you are trying to minimize or maximize. Some points to remember about choosing the objective function are:

- The ANSYS program always tries to *minimize* the objective function. If you need to *maximize* a quantity *x*, restate the problem and minimize the quantity *x1 = C-x* or *x1 = 1/x*, where *C* is a number much larger than the expected value of *x*. *C-x* is generally a better way to define the objective function than *1/x* because the latter, being an inverse relationship, cannot be as accurately represented by the approximations used in the subproblem approximation method.
- The objective function should remain positive throughout the optimization, because negative values may cause numerical problems. To prevent negative values from occurring, simply add a sufficiently large positive number to the objective function (larger than the highest expected objective function value).

1.7. Hints for Performing Design Optimization

This section offers some hints for that you can employ to enhance design optimization at your site:

- Generating the Analysis File
- Fixing Design Variable Values After Execution
- Modifying the Optimization Variables After Execution
- Local Versus Global Minimum
- Minimum Weight Versus Minimum Volume
- Mesh Density
- Using Substructures

Design optimization involves a *series of analyses* (that is, several loops of the preprocessing-solution-postprocessing-optimization cycle). ANSYS recommends, therefore, that you start with a simple problem first and understand fully the optimization procedure. After you understand the various steps involved in the design optimization process, you will find it easier to optimize your own design.

1.7.1. Generating the Analysis File

There are two ways to derive the design optimization analysis file if you built model interactively: from the internal database command log (**LGWRITE**) (**Utility Menu> File> Write DB Log File**), or from the session log file (**Jobname.LOG**). Using the internal database log instead of **Jobname.LOG** has several advantages.

The **LGWRITE** command has an option (`Kedit` field) to exclude nonessential commands, or to write them to the file as comment lines. This option does some automatic cleanup of the log file for you, although you should still do a final review to ensure that the file is appropriate for optimization. Also, the internal database log will represent the complete database so no piecing together of files is necessary. Because the database log is saved in the database file (**Jobname.DB**), a resumed database will contain its own complete database log. (See Section 8.3: Using the Database Command Log in the *ANSYS Operations Guide* for more information on using the database command log.)

> **Caution:** Use `Kedit` = COMMENT (not `Kedit` = REMOVE) on the **LGWRITE** command. Some commands that are filtered out by the `Kedit` option may be required for subsequent ***GET** operations (for example, **EXTREM** and **PLNSOL**). These commands should be uncommented during the final edit of **Jobname.LGW**.
>
> *Note* — The **/CLEAR** command clears the database from memory, and therefore also clears the database log. A **/CLEAR** is issued automatically at the beginning of each optimization loop. If **LGWRITE** is issued after optimization looping, the resulting file will not contain a complete command history. Typically, the database log should be written before performing optimization looping.

As stated earlier, you should avoid picking operations when defining items which will be used as optimization variables. If you did use picking because it is more convenient, be aware that picking operations cause special GUI-generated commands (such as **FLST** and **FITEM**) to be written to the command log. These commands are documented in the *ANSYS Commands Reference*. However, it may be tedious to convert them to parametric input during the final cleanup of the command log file. You should avoid editing such commands on the log file. Any data change within the **FITEM** command, for example, could render the data to be invalid, and could cause unpredictable results.

1.7.2. Fixing Design Variable Values After Execution

After performing one or more optimization executions (**OPEXE**), you may decide to eliminate certain design variables (**OPVAR**,*Name*,DEL) for subsequent optimization. Typically, you would want these parameters to maintain current values (the values assigned during the last loop, or new values you assign to them), and not to revert back to the original values assigned in the analysis file. Assuming that no reassignment of the parameter value occurs within the loop file, the value of a "deleted" design variable can be fixed as follows:

1. In the analysis file, initialize the design variable parameters before the **/PREP7** command. (Only the parameters which you wish to later fix in value must occur before **/PREP7**.)
2. Before the next optimization execution, issue **OPLOOP**,PREP (**Main Menu> Design Opt> Controls**) to begin reading the analysis file from the first occurrence of **/PREP7**.

If you do not perform both steps, each parameter that was previously a design variable will be reset to its initial value during subsequent optimization analyses.

In the following example, we start with two design variables, AREA1 and AREA2, and perform optimization. Then AREA2 is "deleted" (no longer a design variable) and held at its current value (fixed) by following the steps above.

```
AREA1=5.00           ! AREA1 is first area
AREA2=5.00           ! AREA2 is second area
/PREP7               ! Enter PREP7 preprocessor
! Use AREA1 and AREA2 to build a parametric model
....
FINISH
/SOLVE
! Apply loads, etc. and solve
....
FINISH
/POST1
SET,...
....
*GET,SIG1,....        ! Define parameters which will be SVs and OBJ
*GET,SIG2,....
*GET,TVOL,....
....
FINISH
```

```
/OPT                    ! Enter optimization module
OPVAR,AREA1,DV,....     ! Assign parameters AREA1 and AREA2 as DVs
OPVAR,AREA2,DV,....
OPVAR,SIG1,SV,....      ! Assign state variables
OPVAR,SIG2,SV,....
OPVAR,TVOL,OBJ          ! Assign objective function
OPTYPE,SUBP             ! Use subproblem approximation method
OPEXE                   ! Execute
OPVAR,AREA2,DEL         ! Delete design variable AREA2
STATUS                  ! Verify current optimization variables
OPLOOP,PREP             ! Read analysis file from first /PREP7
OPTYPE,....             ! Specify desired optimization type
....                    ! Specify other OPT controls
OPEXE                   ! Execute optimization
FINISH
```

Please see the **OPVAR**, **OPTYPE**, **OPEXE**, and **OPLOOP** command descriptions for more information.

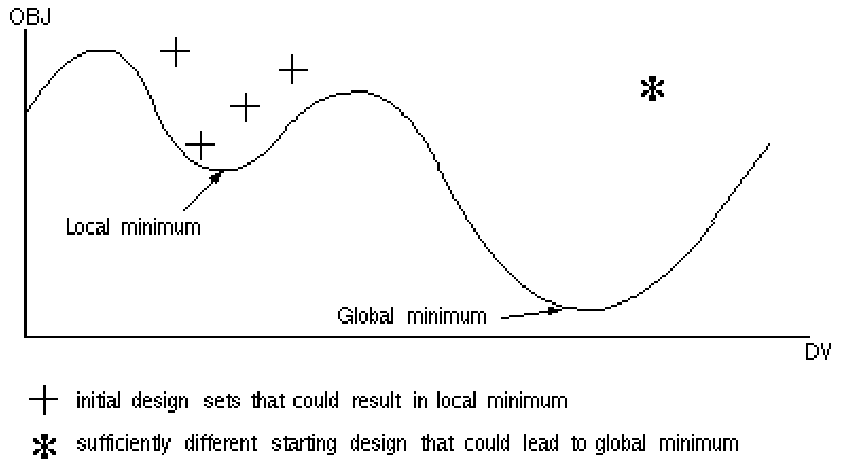
1.7.3. Modifying the Optimization Variables After Execution

Optimization variables can be modified between executions (**OPEXE**) by use of the **OPVAR** command (**Main Menu> Design Opt> Design Variables**). For example, you may wish to change the tolerance of the objective function, the upper or lower limit of a state variable, or you may decide to delete a design variable or define a new one. Whatever the reason, if optimization variables are modified (or new ones are added) after an optimization analysis, a partial clearing of the optimization database is automatically performed. This does not affect the existing design sets and optimization settings you have defined. Only information relating to the optimization calculations (transparent to the user) is cleared. This is done to eliminate data which might be invalid for the modified optimization variable set.

1.7.4. Local Versus Global Minimum

Sometimes the solution might terminate at a local minimum instead of at the global minimum (see Figure 1.4: "Local and Global Minima"). To verify that this has not happened, you could rerun the problem with a different starting design (that is, different initial DV values). Additional strategies for avoiding local minima are found in Section 1.5.2.2: Special Considerations for the First Order Method.

Figure 1.4 Local and Global Minima


OBJ
Local minimum
Global minimum
DV
+ initial design sets that could result in local minimum
* sufficiently different starting design that could lead to global minimum

1.7.5. Minimum Weight Versus Minimum Volume

Avoid specifying material density if it is not necessary for the analysis. This will save some computer time because the mass matrix is not calculated. You can still calculate weight parametrically as weight = density x volume, where volume is the item to be minimized (assuming density is uniform throughout the model).

1.7.6. Mesh Density

In shape optimization problems, where the finite element mesh may change from loop to loop, it is important to verify the adequacy of the mesh. By specifying the mesh divisions in terms of parameters or absolute size, you can vary them appropriately for each loop.

Also, for a linear stress or thermal analysis, you can list the percent error in energy norm (see Section 5.3.6: Estimating Solution Error in the *ANSYS Basic Analysis Guide*) in each loop. An interesting extension to this is to run an adaptive meshing loop within a design optimization loop to ensure that the meshing error does not exceed a certain value. Details of adaptive meshing are described in Section 5.4: Adaptive Meshing Hints and Comments. To list the percent error, use one of these methods:

Command(s): PRERR
GUI: Main Menu> General Postproc> List Results> Percent Error
Utility Menu> List> Results> Percent Error

1.7.7. Using Substructures

If only portions of a model change during design optimization, consider substructuring the unchanging portions. The optimization run would then loop through just the use pass (and expansion pass if necessary), resulting in a significant savings in computer time.

1.8. Sample Optimization Analysis

In the following example, you will perform an optimization analysis of a hexagonal steel plate.

1.8.1. Problem Description

You will build a parametric model of a hexagonal steel plate, using thickness t1 and fillet radius fil as the parameters. All other dimensions are fixed.

This example uses a 2-D model and takes advantage of symmetry.

1.8.2. Problem Specifications

The loading for this example is tensile pressure (traction) of 100 MPa at the three flat faces.

The following material properties are used for this analysis:

Thickness = 10 mm
Young's modulus (E) = 2.07e5 MPa
Poisson's ratio (υ) = 0.3

1.8.3. Using a Batch File for the Analysis

You can perform the example optimization analysis of a hexagonal steel plate using the ANSYS commands shown below. Items prefaced with an exclamation point (!) are comments.

The analysis happens in essentially two passes. In the first pass, you create the analysis file. In the second pass, you create the optimization input. If you prefer, you can perform the second pass of the example analysis using the GUI method rather than ANSYS commands. See Section 1.8.4: Using the GUI for the Analysis for details.

```
!  *******************************************************
!  First Pass:  Create analysis file.
!  *******************************************************
```

```
*create,hexplate
!
!  GEOMETRY (in mm)
!-----------------
*afun,deg             ! Degree units for trig. functions
inrad=200*cos(30)-20  ! Inner radius
t1=30                 ! Thickness
fil=10                ! Fillet radius

/prep7
!  Create the three bounding annuli
cyl4,-200,,inrad,-30,inrad+t1,30
cyl4,200*cos(60),200*sin(60),inrad,-90,inrad+t1,-150
cyl4,200*cos(60),200*sin(-60),inrad,90,inrad+t1,150
aplot
aadd,all
adele,all                  ! Delete area, keep lines
lplot
!  Fillets on inner slot
lsel,,radius,,inrad+t1     ! Select inner arcs
l1 = lsnext(0)             ! Get their line numbers
l2 = lsnext(l1)
l3 = lsnext(l2)
lfillet,l1,l2,fil          ! Fillets
lfillet,l2,l3,fil
lfillet,l3,l1,fil
lsel,all
lplot
!  Keep only symmetric portion
wprot,,90
lsbw,all
wprot,,,60
lsbw,all
csys,1
lsel,u,loc,y,0,60
ldele,all,,,1
lsel,all
ksll
ksel,inve
kdele,all                  ! Delete unnecessary keypoints
ksel,all
lplot
!  Create missing lines and combine right edge lines
csys,0
ksel,,loc,y,0
lstr,kpnext(0),kpnext(kpnext(0))  ! Bottom symmetry edge
ksel,all
csys,1
ksel,,loc,y,60
lstr,kpnext(0),kpnext(kpnext(0))  ! 60-deg. symm. edge
ksel,all
csys,0
lsel,,loc,x,100
lcomb,all           ! Add lines at the right edge
lsel,all
!  Create the area
al,all
aplot

! MESHING
! -------
et,1,82,,,3         ! Plane stress with thickness
r,1,10              ! Thickness
mp,ex,1,2.07e5      ! Young's modulus of steel, MPa
mp,nuxy,1,0.3       ! Poisson's ratio
smrt,3
amesh,all
eplot
finish

! LOADING
! -------
```

```
/solu
csys,1
lsel,u,loc,y,1,59
dl,all,,symm        !  Symmetry b.c.
csys,0
lsel,,loc,x,100
sfl,all,pres,-50    !  Pressure load (MPa)
lsel,all
lplot

! SOLUTION
! --------
eqslv,pcg
solve

! POSTPROCESSING
! --------------
/post1
plnsol,s,eqv                ! Equivalent stress contours
/dscale,,off                ! Displacement scaling off
/expand,6,polar,half,,60    ! Symmetry expansion
/replot
/expand
!  Retrieve maximum equivalent stress and volume
nsort,s,eqv
*get,smax,sort,,max         ! smax = max. equivalent stress
etable,evol,volu
ssum
*get,vtot,ssum,,item,evol   ! vtot = total volume
finish
*end
!
*use,hexplate ! RUN INITIAL ANALYSIS
!
!  ********************************************************
!  Second Pass:  Create optimization input.
!  ********************************************************

! ENTER OPT AND IDENTIFY ANALYSIS FILE
/opt
opanl,hexplate  !Assign oploop file
! IDENTIFY OPTIMIZATION VARIABLES
opvar,t1,dv,20.5,40   ! DVs: Thickness
opvar,fil,dv,5,15     !      Fillet radius
opvar,smax,sv,,150    ! SV:  Maximum equivalent stress
opvar,vtot,obj,,,1    ! OBJ: Total volume, tolerance = 1.0

! RUN THE OPTIMIZATION
opkeep,on             ! Save best design
optype,subp           ! Subproblem approximation method
opsave,anfile,opt0    ! Save the current opt database
opexe

! REVIEW RESULTS
oplist,all,,,1        ! List all design sets
plvaropt,t1,fil       ! DVs t1 & fil vs. set number
plvaropt,smax         ! SV smax vs. set number
plvaropt,vtot         ! OBJ vtot vs. set number
finish
```

1.8.4. Using the GUI for the Analysis

Section 1.8.3: Using a Batch File for the Analysis describes this example optimization analysis as consisting of two passes. In the first you create an analysis file, and in the second you create the optimization input. As discussed earlier in this chapter, you should avoid graphical picking operations when defining a parametric model. Thus, the GUI method is not recommended for performing the first pass of the example analysis and will not be presented here. However, it is acceptable to perform the optimization pass of the hexagonal steel plate example using the GUI method instead of the ANSYS commands shown earlier. The GUI procedure for performing the optimization pass follows.

Step 1: Test Analysis File

To test the analysis file, you clear the database and then read input from the hexplate.lgw file.

1. Choose menu path **Utility Menu> File> Clear & Start New**. Click on OK.
2. When the Verify dialog box appears, click Yes.
3. Change the jobname. Choose menu path **Utility Menu> File> Change Jobname**. The Change Jobname dialog box appears.
4. Change the jobname to hexplate and click on OK.
5. Choose menu path **Utility Menu> File> Read Input from**. In the Files list, click on hexplate.lgw. Then click on OK. You see a replay of the entire analysis. Click on Close when the “Solution is done!” message appears.

Step 2: Enter the Optimizer and Identify Analysis File

In the next several steps of this problem, you optimize the design. The overdesigned steel plate under tension loading of 100 MPa needs to be optimized for minimum weight subject to a maximum von Mises stress limit of 150 MPa. You are allowed to vary the thickness t1 and fillet radius fil.

First, enter the optimizer and identify the analysis file.

1. Choose menu path **Main Menu> Design Opt> Analysis File> Assign**. The Assign Analysis File dialog box appears.
2. In the Files list, click once on hexplate.lgw and then click on OK.

Step 3: Identify the Optimization Variables

1. Choose menu path **Main Menu> Design Opt> Design Variables**. The Design Variables dialog box appears.
2. Click on Add. The Define a Design Variable dialog box appears.
3. In the list of parameter names, click on T1. Type 20.5 in the MIN field and 40 in the MAX field. Click on Apply.
4. In the list of parameter names, click on FIL. Type 5 in the MIN field and 15 in the MAX field. Click on OK.
5. Click on Close to close the Design Variables dialog box.
6. Choose menu path **Main Menu> Design Opt> State Variables**. The State Variables dialog box appears.
7. Click on Add. The Define a State Variable dialog box appears.
8. In the list of parameter names, click on SMAX. Type 150 in the MAX field. Click on OK.
9. Click on Close to close the State Variables dialog box.
10. Choose menu path **Main Menu> Design Opt> Objective**. The Define Objective Function dialog box appears.
11. In the list of parameter names, click on VTOT. Set the TOLER field to 1.0. Click on OK.

Step 4: Run the Optimization

This step involves specifying run time controls and the optimization method, saving the optimization database, and executing the run.

1. Choose menu path **Main Menu> Design Opt> Controls**. The Specify Run Time Controls dialog box appears.
2. Change the OPKEEP setting from “Do not save” to “Save.” Click on OK.
3. Specify an optimization method. Choose menu path **Main Menu> Design Opt> Method/Tool**. The Specify Optimization Method dialog box appears.
4. Choose Sub-Problem. Click on OK. Click OK again.
5. Save the optimization database. Choose menu path **Main Menu> Design Opt> Opt Database> Save**. In the Selection field, type hexplate.opt0. Click on OK.
6. Execute the run. Choose menu path **Main Menu> Design Opt> Run**. Review the settings and click on OK. (If you receive any warning messages during the run, close them.)
7. Notes will appear to let you know which design set ANSYS is currently running. When the run converges, review the Execution Summary. Click on OK.

Step 5: Review the Results

In this step, you start by listing design sets, then graph the objective function and state variables versus set number.

1. Choose menu path **Main Menu> Design Opt> Design Sets> List**. The List Design Sets dialog box appears.
2. Verify that the ALL Sets option is selected. Click on OK.
3. Review the information that appears in the window. Click on Close.
4. Choose menu path **Main Menu> Design Opt> Graphs/Tables**. The Graph/List Tables of Design Set Parameters dialog box appears.
5. For X-variable parameter, select Set number. For Y-variable parameters, select VTOT. Click on OK. Review the graph.
6. Choose menu path **Main Menu> Design Opt> Graphs/Tables**. The Graph/List Tables of Design Set Parameters dialog box appears.
7. For X-variable parameter, select Set number. For Y-variable parameters, select SMAX. Unselect VTOT by clicking on it. Click on OK. Review the graph.

Step 6: Restore the Best Design

In this step, you restore the best design. First, however, save the optimization database to a file.

1. Choose menu path **Main Menu> Design Opt> Opt Database> Save**. The Save Optimization Data dialog box appears.
2. In the Selection field, type hexplate.opt1. Then click on OK.
3. Choose menu path **Main Menu> Finish**
4. Issue the following commands in the ANSYS Input window. After you type each command in the window, press ENTER.

```
resume,hexplate,bdb
```

```
/post1
file,hexplate,brst
lplot
```

5. Choose menu path **Main Menu> General Postproc> Read Results> First Set**.
6. Choose menu path **Main Menu> General Postproc> Plot Results> Contour Plot> Nodal Solu**. The Contour Nodal Solution Data dialog box appears.
7. Choose Stress from the list on the left. Choose von Mises SEQV from the list on the right. Click on OK. Review the plot.
8. Choose menu path **Utility Menu> PlotCtrls> Style> Displacement Scaling**. For DMULT, select 0.0 (off). Click on OK.
9. Choose menu path **Utility Menu> PlotCtrls> Style> Symmetry Expansion> User-Specified Expansion**. The Expansion by Values dialog box appears.
10. Fill in the 1st Expansion of Symmetry section. For NREPEAT, type 6. For TYPE, choose Polar. For PATTERN, choose Alternate Symm. Type 0, 60, and 0 in the DX, DY, and DZ fields, respectively. Click on OK.

Step 7: Exit ANSYS

Click on Quit in the ANSYS Toolbar. Select an option to save, then click on OK.

1.8.5. Where to Find Other Examples

Several ANSYS publications, particularly the *ANSYS Verification Manual* and the *Design Optimization Seminar* course notes, describe additional optimization analyses.

The *ANSYS Verification Manual* consists of test case analyses demonstrating the analysis capabilities of the ANSYS program. While these test cases demonstrate solutions to realistic analysis problems, the *ANSYS Verification Manual* does not present them as step-by-step examples with lengthy data input instructions and printouts. However, most ANSYS users who have at least limited finite element experience should be able to fill in the missing details by reviewing each test case's finite element model and input data with accompanying comments.

The *ANSYS Verification Manual* contains the following optimization analysis test cases:

VM155 - Shape Optimization of a Cantilever Beam
VM157 - Optimization of a Frame Structure

Chapter 2: Topological Optimization

Topological optimization is a form of "shape" optimization, sometimes referred to as "layout" optimization. The purpose of topological optimization is to find the best use of material for a body such that an objective criterion (such as global stiffness or natural frequency) takes on a maximum/minimum value subject to given constraints (such as volume reduction).

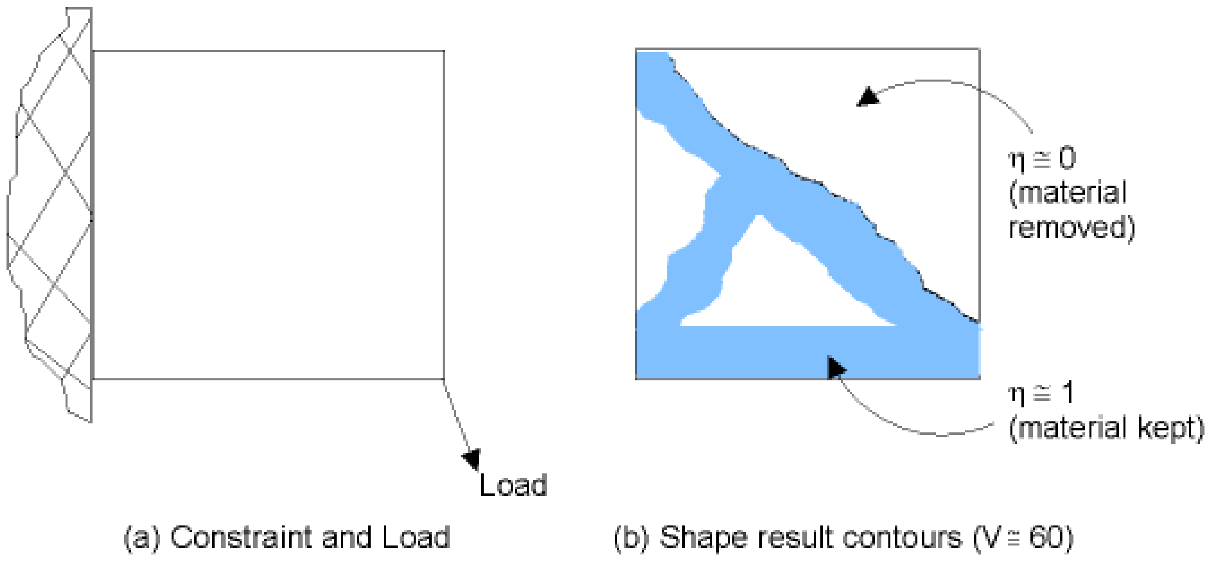
2.1. Understanding Topological Optimization

Unlike traditional optimization, topological optimization does not require you to explicitly define optimization parameters (that is, independent variables to be optimized). In topological optimization, the material distribution function over a body serves as the optimization parameter. You define the structural problem (material properties, FE model, loads, etc.) and the objective function (the function to be minimized/maximized), then select the state variables (the constrained dependent variables) from among a set of predefined criteria.

The goal of topological optimization--the objective function--is to minimize/maximize the criteria selected (minimize the energy of structural compliance, maximize the fundamental natural frequency, etc.) while satisfying the constraints specified (volume reduction, etc.). This technique uses design variables (η_i) that are internal pseudo-densities assigned to each finite element. The densities are plotted via the **PLNSOL**,TOPO and **PLESOL**,TOPO commands (as described in Section 2.2.6: Review the Results).

The standard formulation of topological optimization defines the problem as minimizing the structural compliance while satisfying a constraint on the volume (V) of the structure. Minimizing the compliance is equivalent to maximizing the global structural stiffness. For example, specifying V = 60 means that 60 percent of the material is to be removed in a manner that maximizes the stiffness, with the given load configuration. Figure 2.1: "An Optimization Sample with 60 Percent Volume Reduction" shows a constrained and loaded rectangular area that is to be subjected to topological optimization. Figure (a) shows the loads and boundary conditions and Figure (b) shows the "shape" results in terms of density contours.

Figure 2.1 An Optimization Sample with 60 Percent Volume Reduction



2.2. Employing Topological Optimization

The process for performing a topological optimization consists of the following general steps.

1. Define the structural problem.
2. Select the element types.
3. Specify optimized and non-optimized regions.
4. Define and control the load cases or frequency extraction.
5. Define and control the optimization process.
6. Review the results.

Details of the optimization procedure are presented below. Where appropriate, differences in the procedure for a "batch" versus "interactive" approach are indicated.

2.2.1. Define the Structural Problem

Define the problem as you would for any linear elastic analysis. You need to define material properties (Young's modulus, Poisson's ratio, and possibly the material density). Poisson's ratio must be between 0.1 and 0.4. You then select the proper element types for topological optimization, generate a finite element model, and depending on the criteria you need for your particular topological optimization problem, you will need to apply either:

- Load and boundary conditions for a single or multiple load case linear structural static analysis, or
- Boundary conditions for a modal frequency analysis.

See Chapter 1, "Getting Started with ANSYS" and Chapter 2, "Loading" in the *ANSYS Basic Analysis Guide* for more information about defining the problem.

2.2.2. Select the Element Types

Topological optimization supports 2-D planar, 3-D solid, and shell elements. To use this technique, your model must contain only the following element types:

2-D Solids: PLANE2 or PLANE82
3-D Solids: SOLID92 or SOLID95
Shells: SHELL93

The 2-D elements should be used for plane stress or axisymmetric applications.

2.2.3. Specify Optimized and Non-Optimized Regions

Only those elements identified as type 1 (via the **TYPE** command) undergo topological optimization. Use this rule to control which regions of your model to optimize. For example, if you want to keep material near a hole or a support, you should identify elements in those areas as type 2 or higher:

```
...
ET,1,SOLID92
ET,2,SOLID92
...
TYPE,1
VSEL,S,NUM,,1,2    ! The volume modeled by these elements will be
VMESH,ALL          ! optimized
TYPE,2
VSEL,S,NUM,,3      ! The volume modeled by these elements will not
VMESH,ALL          ! be optimized
...
```

You can use any appropriate ANSYS select and modification command to control the type definitions for various elements.

2.2.4. Define and Control Your Load Cases or Frequency Extraction

You can perform topological optimization based on either linear structural static or modal analysis.

2.2.4.1. Linear Structural Static Analysis

When defining the structural compliance as either the objective or constraint for your topological optimization (**TOCOMP**, **TOVAR**), you must perform a linear structural static analysis during optimization looping. You can perform topological optimization for a single load case or collectively for several load cases. The single-load case is the simplest.

However, to obtain a single optimization solution from several independent load cases, you must use load case write and solve capabilities. After each load case is defined, you must write that data to a file (use the **LSWRITE** command). Finally, you need to solve the collection of load cases by using the **LSSOLVE** command. The **TOLOOP** command performs this last step for you.

For example, the following input fragment shows how three load cases can be combined for a single topological optimization analysis.

```
...
...
D,10,ALL,0,,20,1         ! Define Loads and constraints for 1st load case
NSEL,S,LOC,Y,0
SF,
ALLSEL
LSWRITE,1                ! Write 1st load case
DDEL,                    ! Clear and then define 2nd load case
SFDEL,
NSEL,S,LOC,X,0,1
D,ALL,ALL,0
F,212,FX
LSWRITE,2                ! Write 2nd load case
...                      ! Etc.
LSWRITE,3                ! Write 3rd load case
...                      ! Etc.
FINISH
TOCOMP,MCOMP,MULTIPLE,3  ! Define weighted multiple compliance function
                           "MCOMP"
                         ! considering all three load cases
TOVAR,MCOMP,OBJ          ! Define "MCOMP" as topological objective
TOVAR,VOLUME,CON,,10     ! Define "VOLUME" as topological constraint
TODEF                    ! Initialize topo opt.
TOLOOP,20                ! Solve and perform topological optimization
```

2.2.4.2. Modal Analysis

When defining a natural frequency formulation as an objective for topological optimization (**TOFREQ**, **TOVAR**) a modal analysis must be performed during optimization looping. Depending on the frequency model you need to specify the number of frequencies to solve for during modal analysis by using the **MODOPT** command. You also must specify the number of modes to expand and write by means of **MXPAND**. Note that the element calculation key must be set to 'YES'. The following input fragment shows the modal analysis setup for topological optimization. When using the **TOLOOP** command macro, you need only the **ANTYPE** command because the **MODOPT** and **MXPAND** commands are done by the **TOLOOP** command. Only the Block Lanczos eigensolver is available with the **TOLOOP** command.

```
/SOLUTION
ANTYPE,MODAL               ! Switch to modal analysis
FINISH

TOFREQ,MFREQ,RECIPROCAL,3  ! Define reciprocal frequency function "MFREQ"
                           ! for topological optimization
TOVAR,MFREQ,OBJ            ! Define function "MFREQ" as objective for
```

```
                                topological
                              ! optimization
    TOVAR,VOLUME,CON,,50      ! Define "VOLUME" as constraint for topological
                              ! optimization
    TODEF,1.0D-4              ! Initialize topological optimization
              (accuracy=1.0d-4)
    TOLOOP,20                 ! Solve for first 3 natural frequencies and
                                corresponding
                              ! mode shapes and then perform topological
                                optimization
    ...
```

2.2.5. Define and Control the Optimization Process

The topological optimization process consists of four parts: defining optimization functions, defining objective and constraints, initializing optimization, and executing topological optimization. You can run the fourth part, executing topological optimization, in two ways; carefully controlling and executing each iteration, or automatically performing many iterations. ANSYS recommends the latter approach.

Seven ANSYS commands define and execute topological optimization: **TOFREQ**, **TOCOMP**, **TOVAR**, **TOTYPE**, **TODEF**, **TOEXE**, and **TOLOOP**. The commands **TOCOMP** and **TOFREQ** are used to define topological optimization functions. Command **TOVAR** defines the objective and constraint functions for the optimization problem. The **TOTYPE** command defines the solution approach employed to solve the optimization problem. The **TODEF** command defines a tolerance for convergence and initializes the topological optimization process. **TOEXE** executes a single iteration of optimization. **TOLOOP** executes several iterations.

> *Note* — To correctly perform topological optimization, after you have defined the optimization parameters, you must solve the problem (**SOLVE**) before executing one or more optimization iterations (**TOEXE**). If you do not solve the problem, you will receive an error message when you try to execute an optimization step. The **TOLOOP** command macro performs the solve steps for you.

2.2.5.1. Defining Optimization Functions

First, you define the optimization functions involved in your topological optimization problem. Use **TOCOMP** to define a compliance function for single or multiple load case conditions in a linear static structural analysis. In contrast, **TOFREQ** is intended to define natural frequency type functions. For every topological function, a unique reference name must be specified (note that the reference name "VOLUME" is a predefined name for the total volume function). You may also delete a topological optimization function using **TOCOMP**,RefName or **TOFREQ**,RefName (with remaining fields left blank). The delete option also deactivates the function as a topological objective or constraint if the **TOVAR** command has already been used.

To define a function for topological optimization, use one of these methods:

Command(s): TOCOMP, **TOFREQ**
GUI: Main Menu> Topological Opt> Advanced Opt> Topo Function
Main Menu> Topological Opt> Set Up> Basic Opt

To list all topological optimization functions currently defined use:

Command(s): TOLIST
GUI: Main Menu> Topological Opt> Advanced Opt> List Functions

2.2.5.2. Defining Objective and Constraints

The next step is to assign the objective and the constraint(s) for the topological optimization problem, that is, specify which functions, previously defined by means of **TOCOMP** and **TOFREQ**, are constraints, and which one is the objective function. A predefined topological function "VOLUME" is provided (representing the total volume function), which can be used for either objective or constraint. Note that only the following combinations of objective and constraints are allowed:

Objective	**Valid Constraints**
Single Compliance (**TOCOMP**)	VOLUME
Multiple Compliance (**TOCOMP**)	VOLUME
Single Frequency (**TOFREQ**)	VOLUME
Weighted mean frequency (**TOFREQ**)	VOLUME
Reciprocal mean frequency (**TOFREQ**)	VOLUME
Euclidean norm frequency (**TOFREQ**)	VOLUME
VOLUME	Single Compliance (**TOCOMP**), multiple constraint definition allowed
VOLUME	Multiple Compliance (**TOCOMP**)

To assign the objective and constraint(s) for topological optimization, use one of these methods:

Command(s): TOVAR
GUI: Main Menu> Topological Opt> Advanced Opt> Topo Objective
Main Menu> Topological Opt> Set Up> Basic Opt

The objective function must be defined before you define constraints. Minimum and maximum bounds can be specified for constraint definition. No constraints are needed for the objective function.

You may change a previously declared objective or constraint at any time by simply redefining it. You may also delete an objective or constraint (**TOVAR**,*RefName*,DEL). The delete option does not delete the topological function; it simply deactivates the function as a topological objective or constraint.

For example, the following input fragment shows how a single compliance minimization subject to a volume reduction of 25 percent is defined

```
...
TOCOMP,COMP,SINGLE,1  ! Define single compliance (load case 1) as
                        topological
                      ! optimization function
                      ! "COMP"
TOVAR,COMP,OBJ        ! Define the compliance function "COMP" as
                        objective for
                      ! topological optimization
TOVAR,VOLUME,CON,,25  ! Define predefined total volume function
                        "VOLUME" as constraint
                      ! for topological optimization; Specify a volume
                        reduction of 25 percent
TODEF,1.0d-4          ! Initialize topological optimization
TOLOOP,10,1           ! Do 10 topological optimization iterations
                        automatically
...
```

At any time you can query the current active status from **TOVAR**, **TODEF**, and **TOTYPE** by using the command **TOSTAT**.

Command(s): TOSTAT
GUI: Main Menu> Topological Opt> Advanced Opt> Status
Main Menu> Topological Opt> Status

2.2.5.3. Solving and Initializing Optimization

After defining your optimization parameters, solve the problem (**SOLVE**). You *must* solve the problem before you perform topological optimizations.

Command(s): SOLVE
GUI: Main Menu> Solution> Solve> Current LS

After specifying the optimization problem (see Section 2.2.5.1: Defining Optimization Functions and Section 2.2.5.2: Defining Objective and Constraints) you may select the solution approach you want employed to solve the optimization problem; **TOTYPE** allows you to choose Optimality Criteria (OC) or Sequential Convex Programming (SCP). The OC approach is applicable to problems with volume as constraint only. The SCP approach is applicable to all valid combinations of objective and constraints (see Section 2.2.5.2: Defining Objective and Constraints for a list of valid combinations of objective and constraints).

Command(s): TOTYPE
GUI: Main Menu> Topological Opt> Run

As a last preparation step you must initialize the topological optimization process. Here you also define the termination/convergence accuracy.

Command(s): TODEF
GUI: Main Menu> Topological Opt> Run

The specification details generated at this point are not saved in the ANSYS database. Therefore, if you want to perform another topological optimization after a **RESUME**, you need to reissue all of the commands you used to set up the topological optimization problem (**TOCOMP**, **TOFREQ**, **TOVAR**, **TOTYPE**, and **TODEF**).

2.2.5.4. Executing a Single Iteration

After defining your optimization parameters, you can launch a single iteration. After execution, you can check convergence and display and/or list your current topological results. You may continue to solve and execute additional iterations until you achieve the desired result. The **TOEXE** command is not available in the GUI.

Command(s):
TOEXE

The following example demonstrates how you would execute topological optimization one iteration at a time:

```
...
TOCOMP,COMP,SINGLE,1  ! Define single compliance (load case 1) as topological
                      ! optimization function
                      ! "COMP"
TOVAR,COMP,OBJ        ! Define the compliance function "COMP" as objective for
                      ! topological optimization
TOVAR,VOLUME,CON,,25  ! Define predefined total volume function "VOLUME" as
                        constraint
                      ! for topological optimization; Specify a volume
                        reduction of 25 percent
TOTYPE,OC             ! Use OC approach for optimization problem
TODEF,1.0d-4          ! Initialize topological optimization
/SOLUTION
SOLVE                 ! Perform 1st stress analysis
TOEXE                 ! Perform 1st topological iteration
FINISH
/POST1                ! Enter post processing
PLNSOL,TOPO           ! Plot topological results
*GET,TOPSTAT,TOPO,,CONV   ! Get the topological convergence status
*STAT,TOPSTAT         ! List convergence status
/SOLUTION
SOLVE                 ! Perform 2nd stress analysis
TOEXE                 ! Perform 2nd topological optimization
FINISH
/POST1                    ! Etc.
...
```

The following is an input fragment showing how you could perform a frequency maximization formulation one iteration at a time.

```
...
TOFREQ,FREQ1,SINGLE,1  ! Define single frequency as topological optimization
                       ! function "FREQ1"
TOVAR,FREQ1,OBJ        ! Define the frequency function "FREQ1" as objective for
                       ! topological optimization
TOVAR,VOLUME,CON,,25   ! Define predefined total volume function "VOLUME" as
                       ! constraint for topological optimization; Specify a
                         volume
                       ! reduction of 25 percent
TOTYPE,SCP             ! Use SCP approach for optimization problem
TODEF,1.0d-4           ! Initialize topological optimization
/SOLUTION
ANTYPE,MODAL           ! Switch to modal analysis
MODOPT,LANB,1          ! Specify modal analysis options: Choose Block Lanczos
                       ! solver (default); extract 1 eigenmode.
MXPAND,1,,,YES         ! Expand fundamental mode, and set element
                         calculation key to YES.
SOLVE                  ! Perform modal analysis
TOEXE                  ! Perform 1st topological iteration
FINISH
TOPLOT,0               ! Plot topological results
*GET,TOPSTAT,TOPO,,CONV   ! Get the topological convergence status
*STAT,TOPSTAT          ! List convergence status
/SOLUTION
SOLVE                  ! Perform 2nd modal analysis
TOEXE                  ! Perform 2nd topological optimization
FINISH
TOPLOT,0
...                    ! Etc.
```

One of the main advantages of **TOEXE** is that you can use it to devise your own iterative macros for automatic optimization looping and plotting. The **TOLOOP** command is an ANSYS macro that can perform several optimization iterations.

2.2.5.5. Executing Several Iterations Automatically

After defining your optimization parameters, you can launch several iterations to be executed automatically. After all of the iterations have run, you can check convergence and display and/or list your current topology. You may continue to solve and execute additional iterations if you want. The **TOLOOP** command is an ANSYS macro and, as such, can be copied and customized (see the *ANSYS APDL Programmer's Guide*).

Command(s): TOLOOP
GUI: Main Menu> Topological Opt> Run

The following example demonstrates how you would use the **TOLOOP** macro to execute multiple iterations (in this case, 20) automatically:

```
...                   ! Setup, define and
LSWRITE               !   write 1st load case
...                   ! Setup, define and
LSWRITE               !   write 2nd load case
...                   ! Setup, define and
LSWRITE               !   write 3rd load case
...
TOCOMP,MCOMP,MULTIPLE,3    ! Define multiple compliance (3 load cases) as
                             topological
                           ! optimization function "MCOMP"
TOVAR,MCOMP,OBJ       ! Define the compliance function "MCOMP" as objective
                        for
                      ! topological optimization
TOVAR,VOLUME,CON,,80  ! Define predefined total volume function "VOLUME" as
                        constraint for
                      ! topological optimization; Specify a volume reduction
                        of 80 percent
```

```
TODEF,0.001          ! Initialize topological optimization with .001
                       convergence tolerance
...
/DSCALE,,OFF          ! Remove distortion
/CONTOUR,,3           ! Request 3 contours for each display
TOLOOP,20,1           ! Perform 20 (max.) iterations. Each iteration solves and
                      ! plots results
...
```

Each topological iteration executes an **LSSOLVE** command, a **TOEXE** command, and a **PLNSOL**,TOPO display (optional) command. The optimization iteration process terminates once convergence is attained (defined with **TODEF**) or when the maximum iteration number is achieved (defined with **TOLOOP**)

2.2.6. Review the Results

After your topological optimization solutions are complete, pertinent results are stored in the ANSYS results file (**Jobname.RST**) and are available for additional processing. You can use the following postprocessing options. For more information on any of these options, see the *ANSYS Commands Reference* for the particular command description, or see The General Postprocessor (**/POST1**) in the *ANSYS Basic Analysis Guide*.

For a nodal listing or plot of the pseudo-densities, use the *TOPO* argument of the **PRNSOL** and **PLNSOL** commands. Or you can use the command **TOPLOT**,0.

For an element-based listing or plot of pseudo-densities, use the *TOPO* argument of the **PLESOL** or **PRESOL** commands. Or you can use the command **TOPLOT**,1.

Additionally you may graph or print the history of the topological optimization course over iterations by using the commands **TOGRAPH** or **TOPRINT**.

```
...
/POST1             ! Enter post processor
TOPLOT,1           ! Plot nonaveraged element pseudo-densities
PLNS,TOPO          ! Plot averaged nodal pseudo-densities
TOGRAPH,OBJ        ! Plot iteration history of topological objective
TOGRAPH,CON,VOLUME ! Plot iteration history of topological constraint
                     "VOLUME"
TOPRINT,OBJ        ! Print iteration history of topological objective
TOPRINT,CON        ! Plot iteration history of topological constraint
                     "VOLUME"
...
```

You can also view the results via ANSYS' tabular capabilities:

```
...
ETABLE,EDENS,TOPO
PLETAB,EDENS
PRETAB,EDENS
ESEL,S,ETAB,EDENS,0.9,1.0
EPLOT
...
```

To check the most recent convergence status (that is, the last iteration) and the objective or constraint values, use ***GET**:

```
...
*GET,TOPCV,TOPO,,CONV          ! If TOPCV = 1 (converged)
*GET,TITER,TOPO,,ITER          ! TITER = Iteration counter
*GET,TOBJ,TOPO,ITER-1,TOHO     ! TOBJ = objective function value of last
                                 iteration
*GET,TCON,TOPO,ITER-1,TOHC,1   ! TCON = constraint function value of last
                                 iteration
*STAT
```

For a frequency solution, particularly when using the weighted, reciprocal, or Euclidean formulations, you should look at the true frequencies of the structure at the converged solution by issuing the ***GET** command:

```
...
*GET,FREQ1,MODE,1,FREQ    ! First fundamental frequency
*GET,FREQ2,MODE,2,FREQ    ! Second fundamental frequency
```

You should also look at the mode shapes in **/POST1** by using the **PLDISP** command, and you should review the **ifreq.out** file. For more information on reviewing results of a modal analysis, see Chapter 3, “Modal Analysis” in the *ANSYS Structural Analysis Guide*.

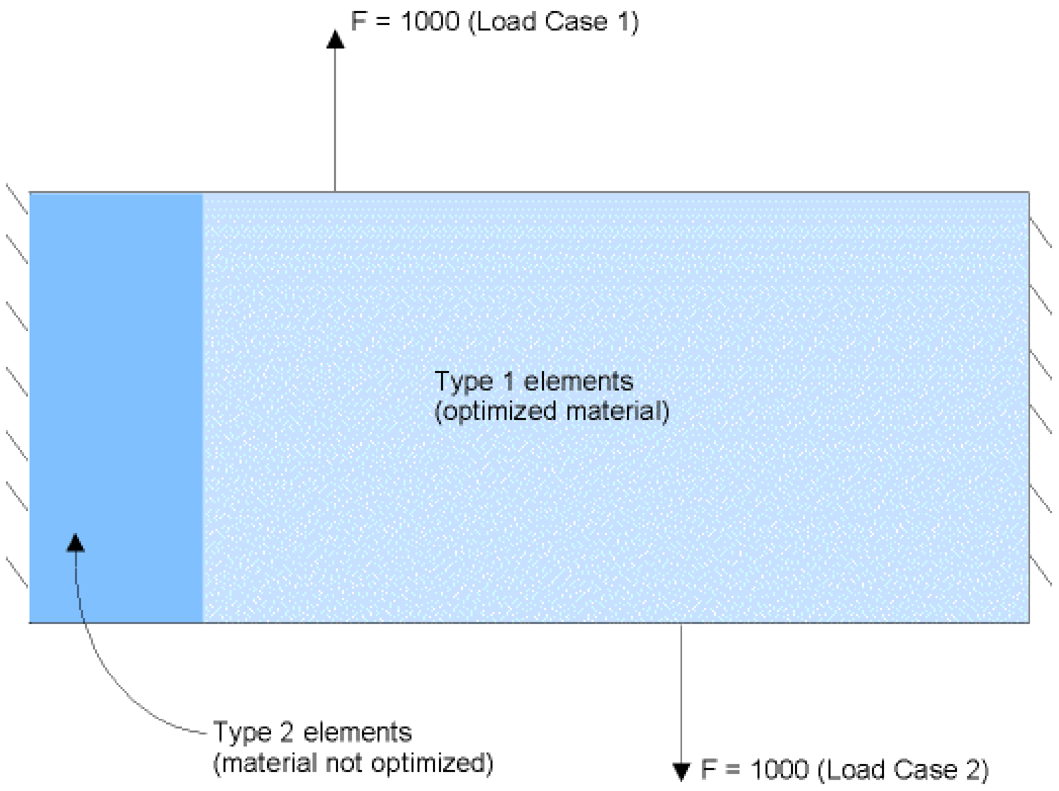
2.3. A 2-D Multiple-Load Case Optimization Example

In the following sample analysis, you will run topological optimization on a beam subjected to two load cases.

2.3.1. Problem Description - First Scenario

A loaded, elastic beam is shown in Figure 2.2: “Beam With Two Load Cases”. The beam is fixed along both ends and subjected to two load cases. Notice that one area of the beam is modeled by type 1 (**TYPE**) finite elements and can therefore be subjected to topological optimization. The other area, identified and modeled by type 2 elements, is not optimized. The goal is to minimize the energy of structural compliances (for each load case independently) subject to a 50 percent reduction in volume of type 1 material.

Figure 2.2 Beam With Two Load Cases



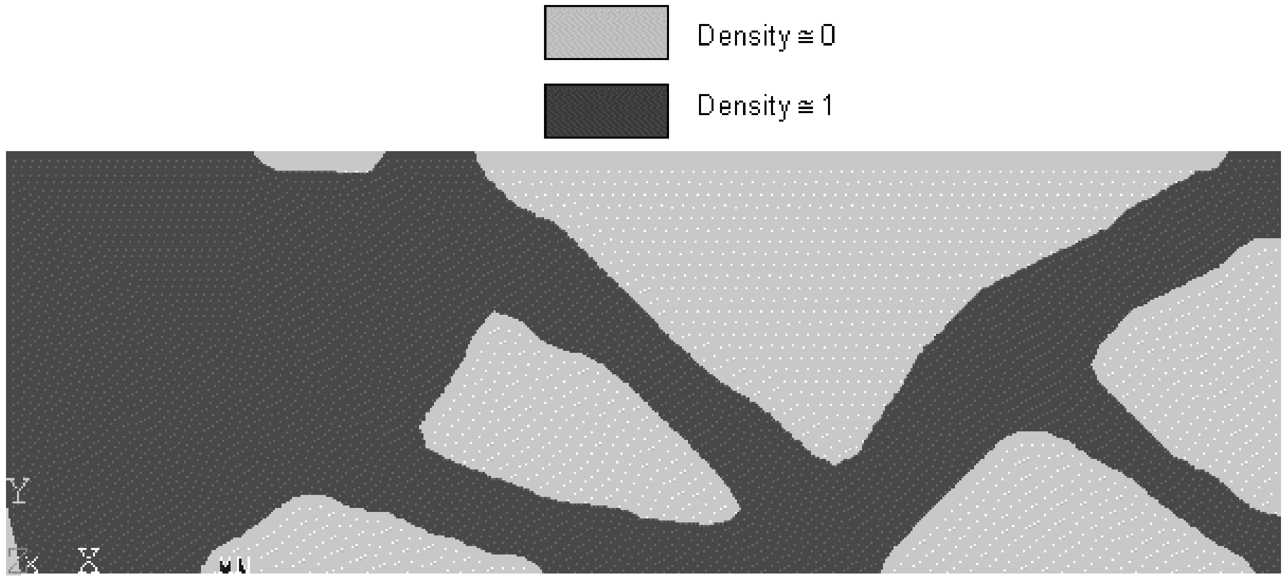
This problem is solved using the ANSYS commands below. Notice that the two load cases are defined and written by the **LSWRITE** command. Using ANSYS selection commands and logic, type 1 and type 2 PLANE82 elements are used to represent optimized and non-optimized areas, respectively. The **TOCOMP** command defines a 2 load case compliance function with the reference name MCOMP. **TOVAR** defines MCOMP as the objective and calls for a 50 percent volume reduction (**TOVAR**,VOLUME,CON,,50). The **TOEXE** command is not explicitly used in this example. Instead, up to 12 iterations of topological optimization are specified via the **TOLOOP** command macro. After the optimization execution, the final objective (compliance) and constraint (volume) histories are graphed and printed, and the optimum weighted compliance value is retrieved from the ANSYS database (***GET**).

```
/TITLE, A 2-D, multiple compliance minimization problem subjected
                        to volume constraint
/PREP7
BLC4,0,0,3,1          ! Create solid model (3 x 1 rectangle)
ET,1,82               ! Use 2-D solids. Type 1 is optimized
ET,2,82               ! Type 2 is not optimized.
MP,EX,1,118E9         ! Linear isotropic, material
MP,NUXY,1,0.3
ESIZE,0.05            ! Use a relatively fine mesh density
TYPE,1
AMESH,ALL             ! Free, rectangular-element meshing
NSEL,S,LOC,X,0,0.4    ! Select region not to be optimized
ESLN
TYPE,2
EMODIF,ALL             ! Define type 2 elements
ALLSEL
NSEL,S,LOC,X,0
D,ALL,ALL,0           ! Fixed at X = 0
NSEL,S,LOC,X,3
D,ALL,ALL,0           ! Fixed at X = 3
FORCE = 1000          ! Value for applied load
NSEL,S,LOC,X,1
NSEL,R,LOC,Y,1
F,ALL,FY,FORCE        ! Define first load case
ALLSEL
LSWRITE,1             ! Write first load case
FDEL,ALL
NSEL,S,LOC,X,2
NSEL,R,LOC,Y,0
F,ALL,FY,-FORCE       ! Define second load case
ALLSEL
LSWRITE,2             ! Write second load case
FDEL,ALL
TOCOMP,MCOMP,MULTIPLE,2  ! Define multiple compliance function
                         ! "MCOMP" for topological optimization
TOVAR,MCOMP,OBJ          ! Define "MCOMP" as topological objective
TOVAR,VOLUME,CON,,50     ! Define "VOLUME" as topological constraint; 50 percent
                           volume reduction
TOTYPE,OC                ! Specify solution approach
TODEF                    ! Initialize topological opt.
/SHOW,topo,grph          ! Put graphics in a file (remove if interactive)
/DSCALE,,OFF
/CONTOUR,,2
TOLOOP,12,1              ! Perform no more than 12 iterations
FINISH
TOGRAPH,OBJ              ! Graph final objective (compliance) history
TOGRAPH,CON              ! Graph final constraint (volume) history
TOPRINT,OBJ              ! Print final objective (compliance) history
TOPRINT,CON              ! Print final constraint (volume) history
*GET,TITER,TOPO,,ITER        ! Get iteration counter
*GET,OCMP,TOPO,TITER-1,TOHO  ! Get final compliance value
```

2.3.2. Problem Results -- First Scenario

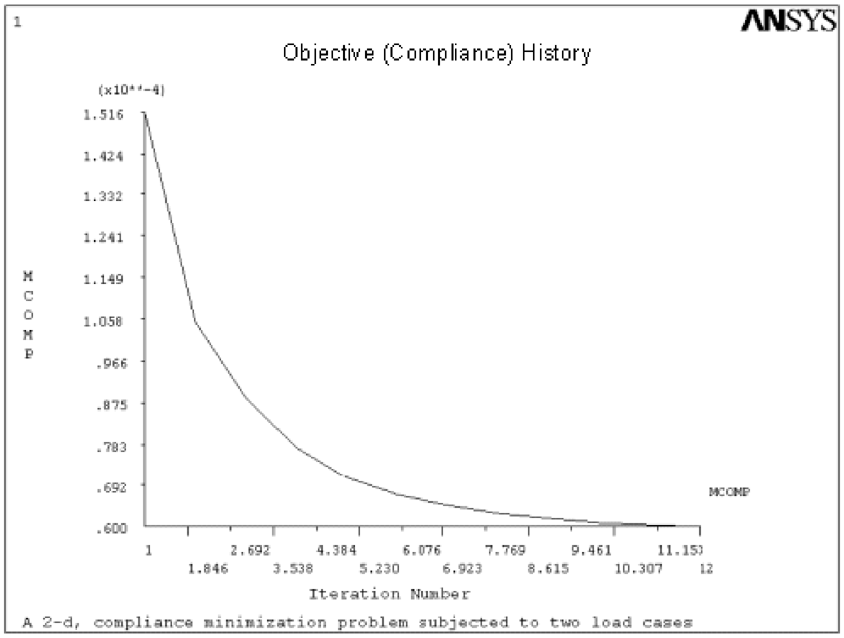
The final optimal shape obtained from the above input stream is shown in Figure 2.3: "Final Topological Shape -- 50 Percent Volume Reduction". Notice that these results were diverted to the topo.grph file for subsequent display. If running ANSYS in an interactive mode, you should remove the **/SHOW** command so you can view graphical results every iteration.

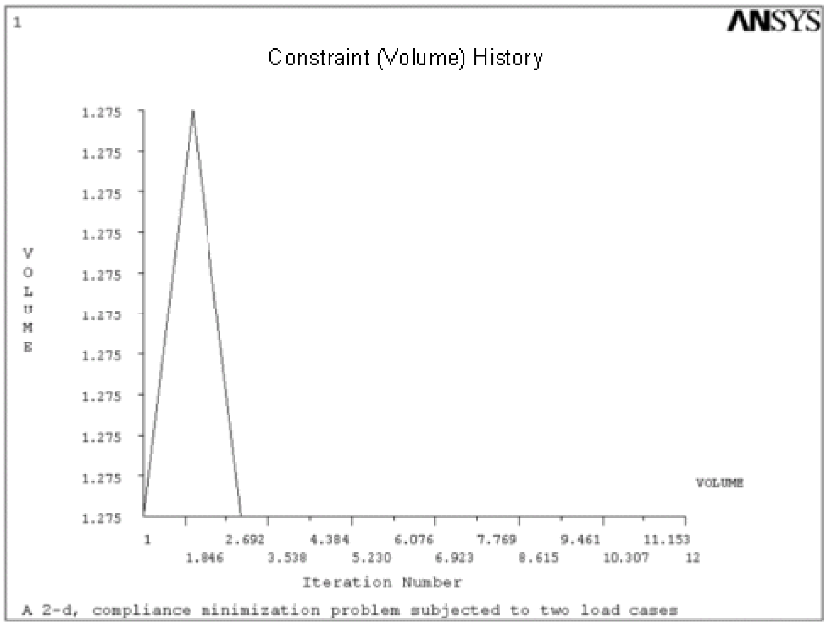
Figure 2.3 Final Topological Shape -- 50 Percent Volume Reduction



A graph of the objective (compliance) and the constraint (volume) history is shown in Figure 2.4: "History of Objective and Constraint Functions". The final optimal weighted compliance value is 0.6E-04.

Figure 2.4 History of Objective and Constraint Functions





2.3.3. Problem Description -- Second Scenario

In contrast to the first scenario, where we have designed for minimum two-load case compliance subject to 50 percent volume reduction, in this second scenario we will optimize for minimum volume design subject to two compliance constraints. The analysis is based on the same finite element model, boundary and load conditions as in the first scenario. (See Figure 2.3: "Final Topological Shape -- 50 Percent Volume Reduction" and the input fragment in Section 2.3.1: Problem Description - First Scenario.) Here, the topological optimization problem is modeled as follows. We first define the volume of type 1 material as objective function by using command **TOVAR**. We then specify two single compliance functions, "COMP1" for the first load case and "COMP2" for the second load case by means of command **TOCOMP** and define them as topological constraints (**TOVAR**) with an upper bound value of 0.6E-4 (the optimum compliance value for the first scenario). Notice that SCP was selected as the solution approach for the topological optimization problem because this problem cannot be solved with the OC approach.

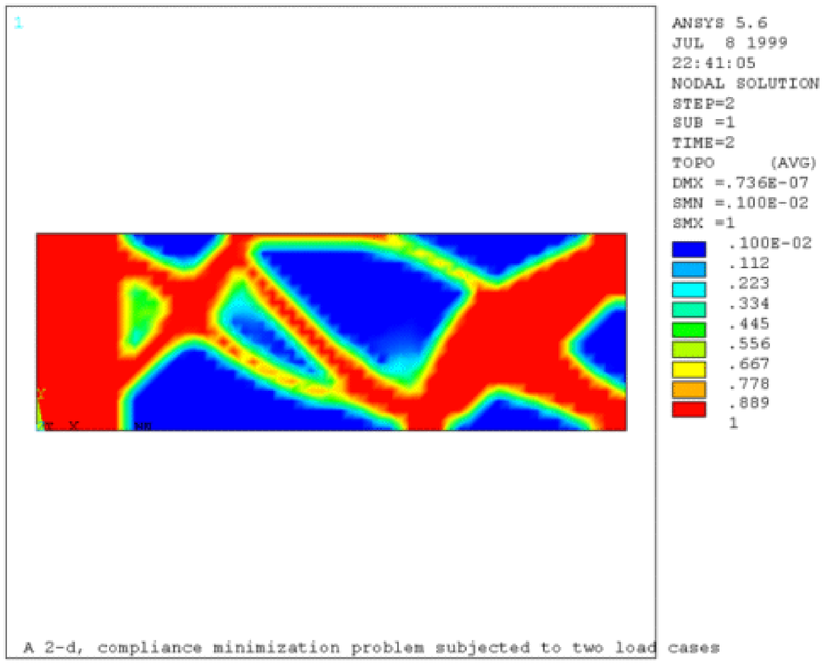
```
/TITLE, A 2-D, volume minimization problem subjected to 2 compliance
             constraints
...        ! Same modeling, boundary conditions and load cases as in
             previous listing

TOCOMP,COMP1,SINGLE,1           ! Define single compliance function "COMP1"
                                 (1st load case)
TOCOMP,COMP2,SINGLE,2           ! Define single compliance function "COMP2"
                                 (2nd load case)
TOVAR,VOLUME,OBJ                ! Define "VOLUME" of type 1 material as
                                  topological objective
TOVAR,COMP1,CON,,0.6E-4,ACTUAL  ! Define first constraint "COMP1" with an
                                  actual upper bound of
                                ! 0.6E-4
TOVAR,COMP2,CON,,0.6E-4,ACTUAL  ! Define second constraint "COMP2" with an
                                  actual upper bound of
                                ! 0.6E-4

TOTYPE,SCP                  ! Specify SCP solution approach
TODEF,1.0d-5                ! Initialize topological opt; set accuracy to
                              1.0d-5
/SHOW,topo2,grph            ! Put graphics in a file (remove if interactive)
TOLOOP,25,1                 ! Perform 25 iterations
FINISH
TOPLOT,1                    ! Plot final densities unaveraged
TOGRAPH,OBJ                 ! Graph final objective (compliance) history
```

```
TOGRAPH,CON,COMP1          ! Graph final constraint (volume) history
TOGRAPH,CON,COMP2          ! Graph final constraint (volume) history
```

2.3.4. Problem Results - Second Scenario

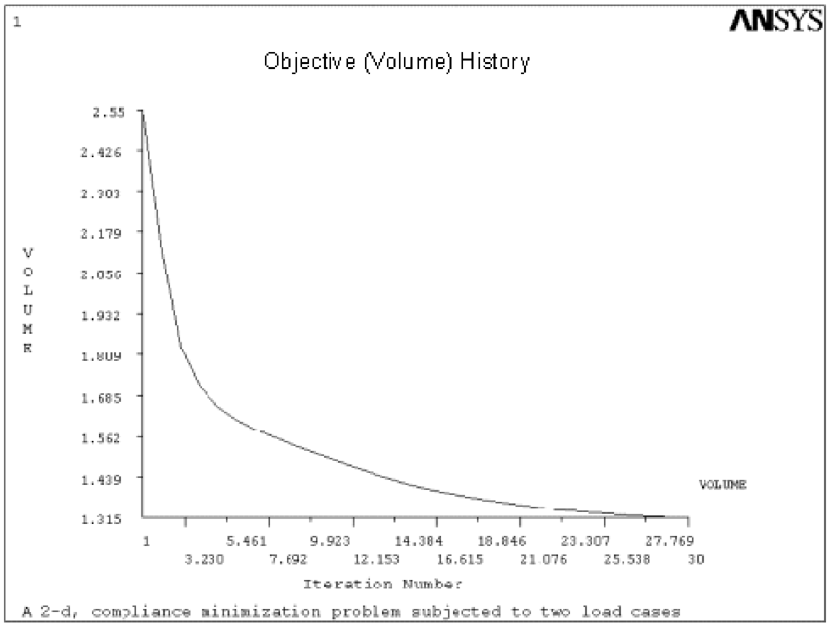
The final optimal shape obtained from the above commands are shown in Figure 2.5: “Final Topological Shape for Second Scenario”. Notice that these results were diverted to the **topo2.grph** file for subsequent display. If running ANSYS in an interactive mode, you should remove the **/SHOW** command in order to view graphical results every iteration.

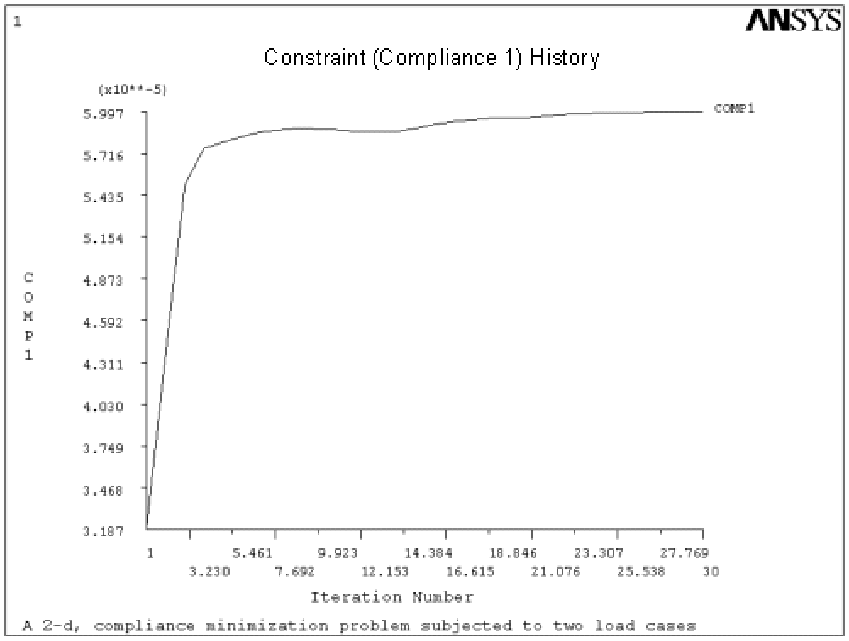
Figure 2.5 Final Topological Shape for Second Scenario

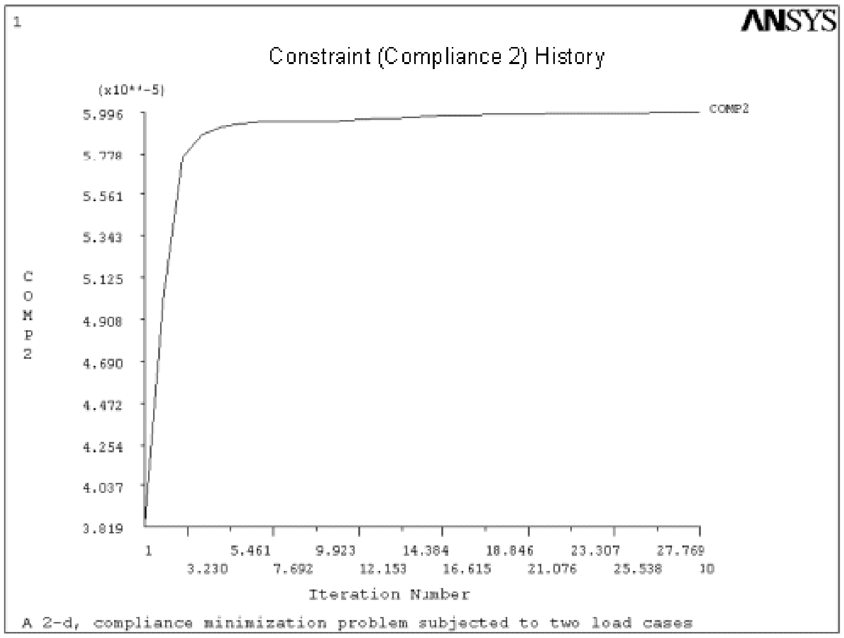


A graph of the objective (volume) and both constraint (compliances) histories is shown in Figure 2.6: “History of Objective and Constraint Functions for Second Scenario”.

Figure 2.6 History of Objective and Constraint Functions for Second Scenario







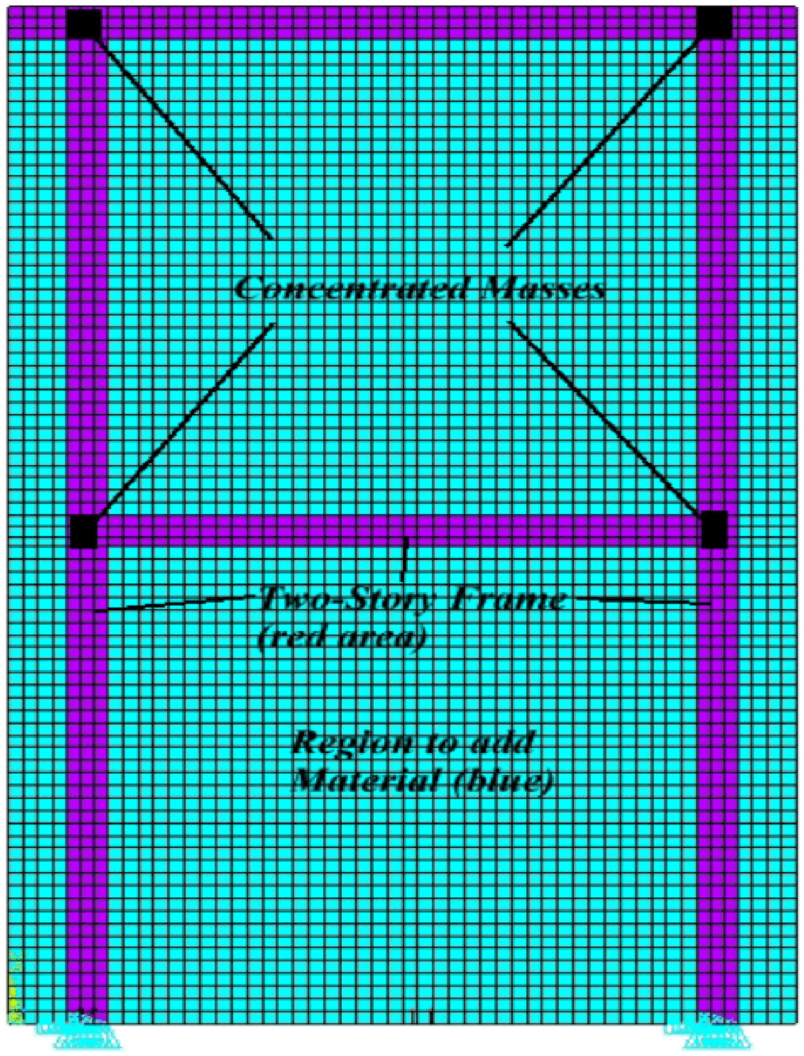
2.4. A 2-D Natural Frequency Maximization Example

In this example analysis, you will run an optimal reinforcement problem of a two-story planar frame as shown in Figure 2.7: “Two-Story Planar Frame”.

2.4.1. Problem Description

This is an initial core-structure (type 2 material) with four concentrated masses specified and assumed to be unchanged during the optimization process. As shown in Figure 2.7: “Two-Story Planar Frame”, the design domain is specified as a rectangle, 5.0 in horizontal length and 8.0 in vertical height with two fixed supported boundaries at the bottom of the domain. Within the optimization problem, material is added (type 1 area) to reinforce the core-structure subjected to dynamic stiffness. The design domain is filled by a material with a Young's Modulus of E = 100, Poisson's Ratio = 0.3, and density = 1.0D-06. Each concentrated mass is 5.0D-06.

Figure 2.7 Two-Story Planar Frame



In this scenario, we maximize the fundamental frequency, whereas the constraint of the total type 1 volume was specified as V = 14. Thus, we define a single frequency function "FREQ1" using **TOFREQ** and specify this function to be the objective for topological optimization (**TOVAR**). We also define the volume constraint with an actual upper bound of 14 (**TOVAR**,VOLUME,,14,ACTUAL). Again, the **TOEXE** command is not explicitly used in this example. Instead, a maximum of 40 iterations of topological optimization are selected via the **TOLOOP** command macro.

```
/TITLE, 2-D Two-Story reinforcement problem - Maximize fundamental frequency
A=0.25                 ! Prepare Model
B=5
C=0.375
D=8
E=3.75
/PREP7
K,1
K,2,C
K,3,C+A
K,4,B/2
K,5,,E
K,6,C,E
K,7,C+A,E
K,8,B/2,E
KSEL,S,,,5,8
KGEN,2,ALL,,,,A
KSEL,S,,,9,12
```

```
KGEN,2,ALL,,,,E
KSEL,S,,,13,16
KGEN,2,ALL,,,,A
ALLSEL
A,1,2,6,5
A,5,6,10,9
A,9,10,14,13
A,13,14,18,17
A,2,3,7,6
A,6,7,11,10
A,10,11,15,14
A,14,15,19,18
A,3,4,8,7
A,7,8,12,11
A,11,12,16,15
A,15,16,20,19

ET,1,82                  ! Define two element type regions
ET,2,82                  ! 1 - optimized region
ASEL,S,,,4,8             ! 2 - non-optimized region
ASEL,A,,,10,12,2
TYPE,2
ESIZE,0.1
AMESH,ALL
ASEL,INVE
TYPE,1
AMESH,ALL
ALLSEL
MP,EX,1,100              ! Material of structure
MP,NUXY,1,0.3
MP,DENS,1,1.0D-6
MP,EX,2,100              ! Material of concentrated masses
MP,NUXY,2,0.3
MP,DENS,2,5.0D-6

ASEL,S,,,6,8,2
ESLA,S,1
EMODIF,ALL,MAT,2         ! Define concentrated masses
ALLSEL
LOCAL,11,0,2.5
ARSYM,X,ALL              ! Full model
NUMM,KP
NUMM,ELEM
NUMM,NODE
LSEL,S,,,14
LSEL,A,,,45
NSLL,S,1
D,ALL,ALL
ALLSEL
FINISH
TOFREQ,FREQ1,SING,1      ! Define single frequency function (1st)
TOVAR,FREQ1,OBJ          ! Define objective for topological optimization
TOVAR,VOLUME,CON, ,14,ACTUAL ! Define volume constraint (upper bound = 14)
TOTYPE,SCP                   ! Select SCP solution approach
TODEF,0.00001,               ! Initialize topological optimization process,
                               accuracy = 0.00001
TOLOOP,40,1                  ! Perform up to 40 iterations
FINISH
TOPLOT,1                     ! Plot final pseudo-densities
TOGRAPH,OBJ                  ! Graph final objective (1st frequency) history
TOGRAPH,CON                  ! Graph final constraint (volume) history
```

2.4.2. Problem Results

Figure 2.8: "Final Topological Shape for Maximum Fundamental Frequency" shows the optimal shape of the reinforcement optimization problem and Figure 2.9: "History of Fundamental Frequency" the history of the first frequency and volume, respectively (**TOGRAPH**) over optimization iteration.

Figure 2.8 Final Topological Shape for Maximum Fundamental Frequency

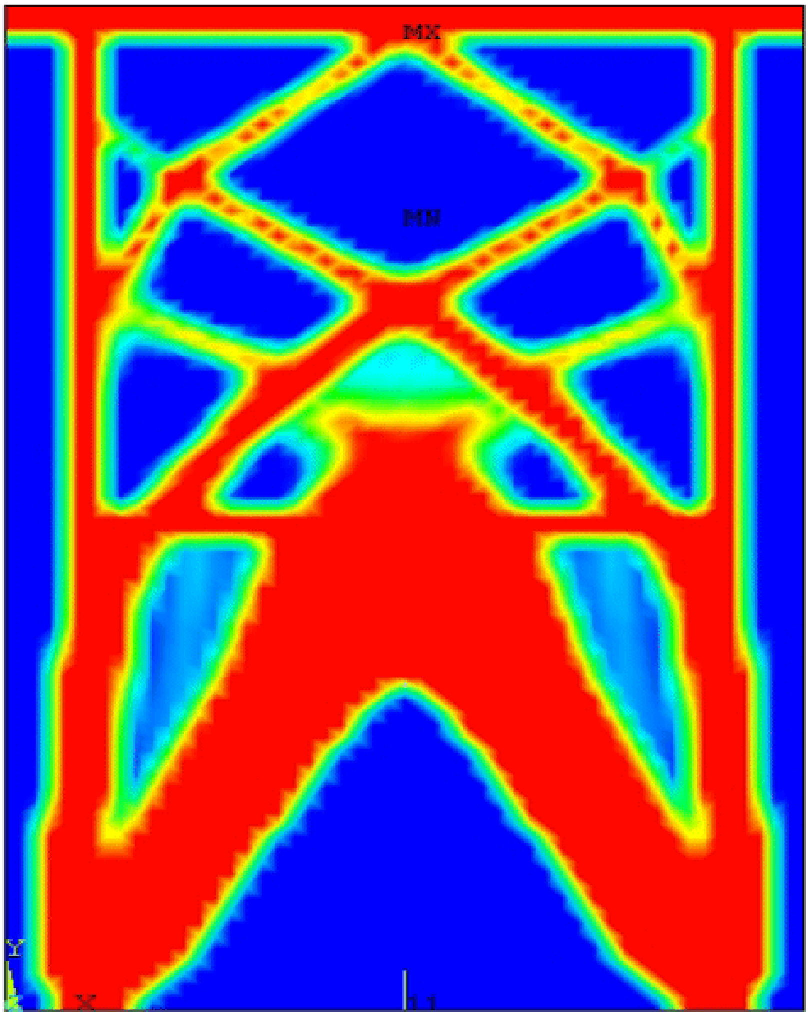
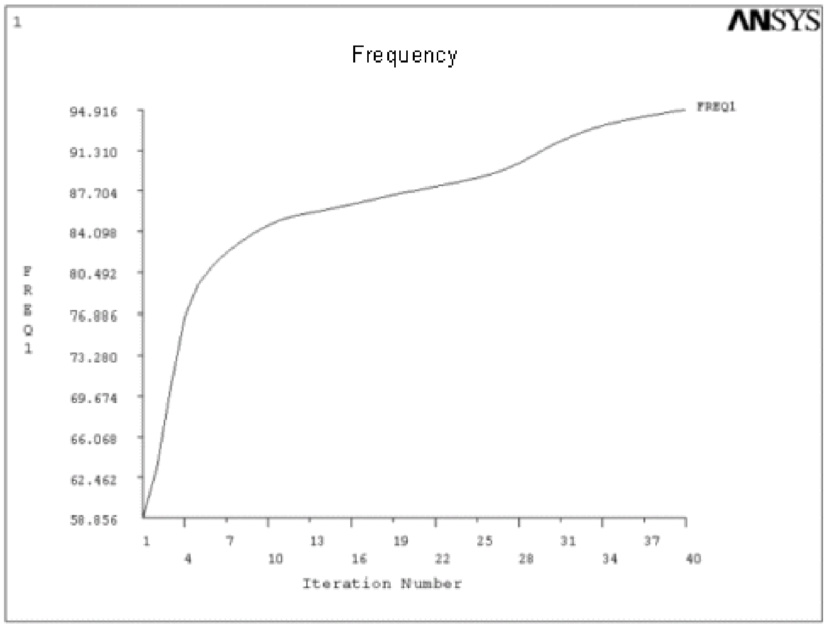
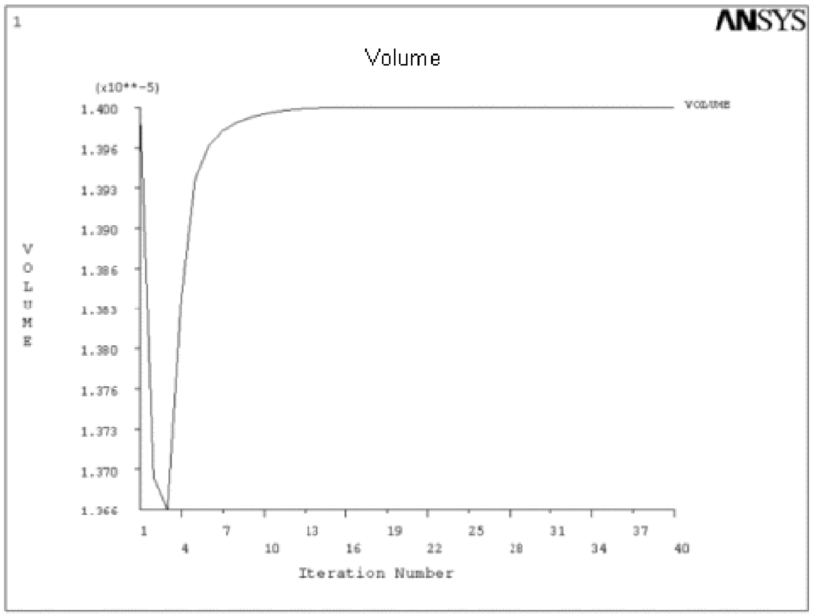


Figure 2.9 History of Fundamental Frequency





2.5. Hints and Comments

Use the following hints and comments to enhance your topological optimization:

- Results are sensitive to your load configuration. Small changes to your applied loads or load distributions can lead to significant differences in results.
- Results are sensitive to the density of the finite element mesh. In general, a very fine mesh will produce "clear" topological results. A coarse mesh will lead to "fuzzier" results. However, a model with a large number of elements will take more time to reach convergence.
- Under certain circumstance, a truss-like solution will result. This will happen when you request a high volume reduction and a very fine finite element mesh. For example, a large reduction could be 80 percent or more volume removed (set in **TOVAR**).
- If you have several load cases, there are many ways to combine your loads and to generate topological results. Consider, for example, what you could do with five different load cases. You may choose to perform five separate topological optimization analyses, or you could perform a single optimization analysis for all five load independent cases. Also, you could combine all five loads into one load case and perform a single topological analysis. In summary, you could produce seven different topological optimization solutions:

  – 5 independent topological solutions (1 for each load case)

  – 1 topological solution for 5 independent load cases

  – 1 topological solution for the combined load cases

Additional results and combination of results are also possible.

- Results are sensitive to Poisson's ratio but not Young's modulus. However, the effects of the dependence to Poisson's ratio are usually not significant.
- Maximizing a chosen natural frequency is usually used as the objective in a frequency topological optimization problem. However, in the frequency optimization problem, when one maximizes a lower frequency, higher eigenvalues may fall down to the lower values. This means that if the optimization process is to follow a specified mode of the structure, then the order number of this mode may be changed during

the optimization process. For example, at the beginning we may wish to optimize the *k*th eigenfrequency, but the optimal solution obtained may correspond to the *k*+*p*'th mode, where *p* > 0. Thus the problem can have an unexpected solution. In contrast, if you follow the number of modal order (for example, to optimize the *k*th eigenfrequency), then the mode being optimized may change to another one. In this case, the sensitivities of the objective function become discontinuous, and may cause oscillation and divergence in the iterative optimization process. To overcome this problem, several mean-frequency functions (see **TOFREQ**) can be used to smooth out the frequency objective.

- The specifications set using **TOCOMP**, **TOFREQ**, **TOVAR**, **TODEF**, **TOTYPE**, and **TOLOOP** are not saved in the ANSYS database; therefore, you will need to specify your optimization goals and definitions each time you use topological optimization.

Chapter 3: Probabilistic Design

Probabilistic design is an analysis technique for assessing the effect of uncertain input parameters and assumptions on your model.

A probabilistic analysis allows you to determine the extent to which uncertainties in the model affect the results of a finite element analysis. An *uncertainty* (or random quantity) is a parameter whose value is impossible to determine at a given point in time (if it is time-dependent) or at a given location (if it is location-dependent). An example is ambient temperature; you cannot know precisely what the temperature will be one week from now in a given city.

In a probabilistic analysis, statistical distribution functions (such as the Gaussian or normal distribution, the uniform distribution, etc.) describe uncertain parameters. (See the *ANSYS Modeling and Meshing Guide* for a description of ANSYS parameters.)

This section describes the ANSYS Probabilistic Design System (PDS) and how to use it to perform a probabilistic design analysis.

3.1. Understanding Probabilistic Design

Computer models are expressed and described with specific numerical and deterministic values; material properties are entered using certain values, the geometry of the component is assigned a certain length or width, etc. An analysis based on a given set of specific numbers and values is called a deterministic analysis. Naturally, the results of a deterministic analysis are only as good as the assumptions and input values used for the analysis. The validity of those results depend on how correct the values were for the component under real life conditions.

In reality, every aspect of an analysis model is subjected to scatter (in other words, is uncertain in some way). Material property values are different if one specimen is compared to the next. This kind of scatter is inherent for materials and varies among different material types and material properties. For example, the scatter of the Young's modulus for many materials can often be described as a Gaussian distribution with standard deviation of ±3 - 5%. Likewise, the geometric properties of components can only be reproduced within certain manufacturing tolerances. The same variation holds true for the loads that are applied to a finite element model. However, in this case the uncertainty is often due to a lack of engineering knowledge. For example, at elevated temperatures the heat transfer coefficients are very important in a thermal analysis, yet it is almost impossible to measure the heat transfer coefficients. This means that almost all input parameters used in a finite element analysis are inexact, each associated with some degree of uncertainty.

It is neither physically possible nor financially feasible to eliminate the scatter of input parameters completely. The reduction of scatter is typically associated with higher costs either through better and more precise manufacturing methods and processes or increased efforts in quality control; hence, accepting the existence of scatter and dealing with it rather than trying to eliminate it makes products more affordable and production of those products more cost-effective.

To deal with uncertainties and scatter, you can use the ANSYS Probabilistic Design System (PDS) to answer the following questions:

- If the input variables of a finite element model are subjected to scatter, how large is the scatter of the output parameters? How robust are the output parameters? Here, output parameters can be any parameter that ANSYS can calculate. Examples are the temperature, stress, strain, or deflection at a node, the maximum temperature, stress, strain, or deflection of the model, etc.

- If the output is subjected to scatter due to the variation of the input variables, then what is the probability that a design criterion given for the output parameters is no longer met? How large is the probability that an unexpected and unwanted event takes place (what is the failure probability)?
- Which input variables contribute the most to the scatter of an output parameter and to the failure probability? What are the sensitivities of the output parameter with respect to the input variables?

Probabilistic design can be used to determine the effect of one or more variables on the outcome of the analysis. In addition to the probabilistic design techniques available, the ANSYS program offers a set of strategic tools that can be used to enhance the efficiency of the probabilistic design process. For example, you can graph the effects of one input variable versus an output parameter, and you can easily add more samples and additional analysis loops to refine your analysis.

3.1.1. Traditional (Deterministic) vs. Probabilistic Design Analysis Methods

In traditional deterministic analyses, uncertainties are either ignored or accounted for by applying conservative assumptions.

Uncertainties are typically ignored if the analyst knows for certain that the input parameter has no effect on the behavior of the component under investigation. In this case, only the mean values or some nominal values are used in the analysis. However, in some situations the influence of uncertainties exists but is still neglected; for example, the Young's modulus mentioned above or the thermal expansion coefficient, for which the scatter is usually ignored. Let's assume you are performing a thermal analysis and you want to evaluate the thermal stresses (thermal stresses are directly proportional to the Young's modulus as well as to the thermal expansion coefficient of the material). The equation is:

$$\sigma_{therm} = E \alpha \Delta T$$

If the Young's modulus alone has a Gaussian distribution with a 5% standard deviation, then there is almost a 16% chance that the stresses are more than 5% higher than what you would think they are in a deterministic case. This figure increases if you also take into account that, typically, the thermal expansion coefficient also follows a Gaussian distribution.

Random Input Variables taken into account	**Probability that the thermal stresses are more than 5% higher than expected**	**Probability that the thermal stresses are more than 10% higher than expected**
Young's modulus (Gaussian distribution with 5% standard deviation)	~16%	~2.3%
Young's modulus and thermal expansion coefficient (each with Gaussian distribution with 5% standard deviation)	~22%	~8%

When a conservative assumption is used, this actually tells you that uncertainty or randomness is involved. Conservative assumptions are usually expressed in terms of safety factors. Sometimes regulatory bodies demand safety factors in certain procedural codes. If you are not faced with such restrictions or demands, then using conservative assumptions and safety factors can lead to inefficient and costly over-design. You can avoid over-design by using probabilistic methods while still ensuring the safety of the component.

Probabilistic methods even enable you to quantify the safety of the component by providing a probability that the component will survive operating conditions. Quantifying a goal is the necessary first step toward achieving it. Probabilistic methods can tell you how to achieve your goal.

3.1.2. Reliability and Quality Issues

Use probabilistic design when issues of reliability and quality are paramount.

Reliability is usually always a concern because product or component failures have significant financial consequences (costs of repair, replacement, warranty, or penalties); worse, a failure can result in injury or loss of life. Although perfection is neither physically possible nor financially feasible, probabilistic design helps you to design safe and reliable products while avoiding costly over-design and conserve manufacturing resources (machining accuracy, efforts in quality control, and so on).

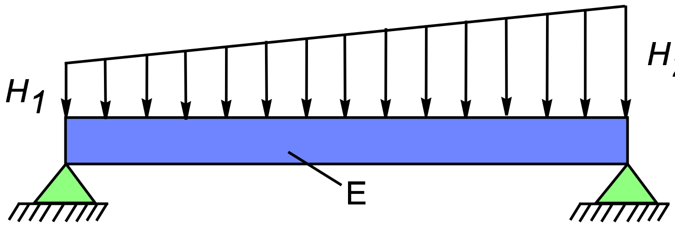
Quality is the perception by a customer that the product performs as expected or better. In a quality product, the customer rarely receives unexpected and unpleasant events where the product or one of its components fails to perform as expected. By nature, those rare "failure" events are driven by uncertainties in the design. Here, probabilistic design methods help you to assess how often "failure" events may happen. In turn, you can improve the design for those cases where the "failure" event rate is above your customers' tolerance limit.

3.2. Probabilistic Design Terminology

To understand the terminology involved in probabilistic design, consider the following problem.

Find the scatter of the maximum deflection of a beam under a random load of snow. The snow should have a linear distribution along the length of the beam with a height of H_1 at one end and a height of H_2 at the other. The beam material is described by various parameters including the Young's modulus, for which a mean value and standard deviation have been derived.

Figure 3.1 A Beam Under a Snow Load



PDS Term	Description
random input variables (RVs)	Quantities that influence the result of an analysis. In probabilistic design, RVs are often called "drivers" because they drive the result of an analysis. You must specify the type of statistical distribution the RVs follow and the parameter values of their distribution functions. For the beam example, the heights H_1 and H_2 and the Young's modulus *E* are clearly the random input variables. Naturally, the heights H_1 and H_2 cannot be negative and more often there will be little snow and only a few times there will be a lot of snow. Therefore, it might be appropriate to model the height of the snow as an exponential or a lognormal distribution, both of which have the bulk of the data at lower values.

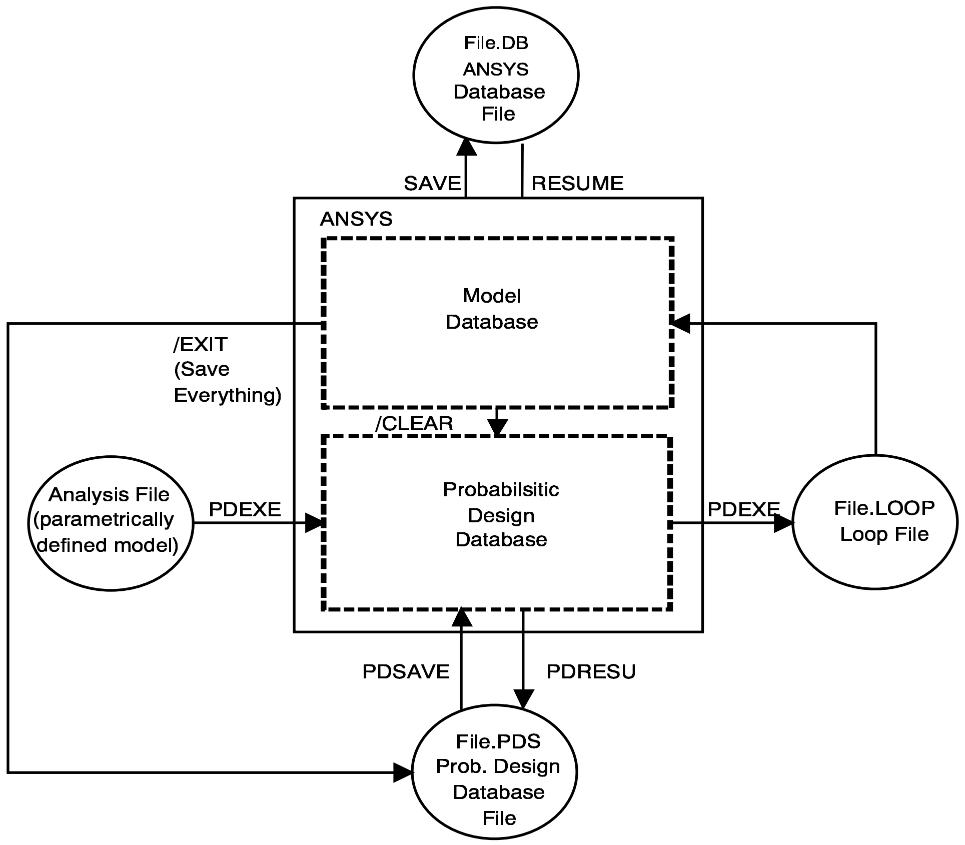
PDS Term	Description
correlation	Two (or more) RVs which are statistically dependent upon each other. In the beam example, it is unlikely that one side of the roof (beam) supports a great deal of snow while almost no snow exists on the other. It is not necessary that H_1 and H_2 are exactly identical, but with a lot of snow then H_1 and H_2 both likely have larger values and with little snowfall then both would be lower. Therefore, H_1 and H_2 are correlated, although the correlation must not be mistaken for a direct mathematical dependency. In the beam example, no numerical dependency exists but rather a certain trend between the two heights; with this particular H_1 and H_2 it is unlikely that their values will be drastically different at any given point in time. *Note* — Mathematical dependencies have some numerical dependence on each other. For example, a true correlation exists if, when one parameter value doubles, another parameter value also doubles.
random output parameters (RPs)	The results of a finite element analysis. The RPs are typically a function of the RVs; that is, changing the values of the random input variables should change the value of the random output parameters. In our beam example, the maximum beam deflection is a random output parameter.
probabilistic design variables	The random input variables (RVs) and random output parameters (RPs) are collectively known as *probabilistic design variables*. In the ANSYS Probabilistic Design System (PDS), you must identify which parameters in the model are RVs and which are RPs. You can define up to a total of 5,000 RVs and RPs.
sample	A unique set of parameter values that represents a particular model configuration. A sample is characterized by random input variable values. If you measure the Young's modulus of a given beam, and the snow height on a given day at each end of that beam, then the three values for E, H_1, and H_2 would constitute one sample. Think of a sample as one virtual prototype. Every component manufactured represents one sample, because you can measure its particular properties (material, geometry, etc.) and obtain specific values for each. In statistics, however, *sample* also has a wider and more general use. For example, any single measured value of any physical property is considered to be one sample. Because a probabilistic analysis is based on a statistical evaluation of the random output parameters (RPs), the values of the RPs are also called samples.
simulation	The collection of all samples that are required or that you request for a given probabilistic analysis. The simulation contains the information used to determine how the component would behave under real-life conditions (with all the existing uncertainties and scatter); therefore, all samples represent the simulation of the behavior.
analysis file	An ANSYS input file containing a complete analysis sequence (preprocessing, solution, and postprocessing). The file must contain a parametrically defined model using parameters to represent all inputs and outputs to be used as RVs and RPs.

PDS Term	Description
loop or *simulation loop*	A single pass through the analysis file. In each loop, the PDS uses the values of the RVs from one sample and executes the user-specified analysis. The PDS collects the values for the RPs following each loop.
loop file	The probabilistic design loop file (**Jobname.LOOP**), created automatically by ANSYS via the analysis file. The PDS uses the loop file to perform analysis loops.
probabilistic model	The combination of definitions and specifications for the deterministic model (in the form of the analysis file). The model has these components: • Random input variables (RVs) • Correlations • Random output parameters (RPs) • The selected settings for probabilistic method and its parameters. If you change any part of the probabilistic model, then you will generate different results for the probabilistic analysis (that is, different results values and/or a different number of results). For example, modifying the analysis file may affect the results file. If you add or take away an RV or change its distribution function, you solve a different probabilistic problem (which again leads to different results). If you add an RP, you will still solve the same probabilistic problem, but more results are generated.
probabilistic design database (PDS data-base)	The database containing the current probabilistic design environment, which includes: • Random input variables (RVs) • Correlations between RVs • Random output parameters (RPs) • Settings for probabilistic methods • Which probabilistic analyses have been performed and in which files the results are stored • Which output parameters of which probabilistic analyses have been used for a response surface fit, the regression model that has been used for the fitting procedure, and the results of that fitting procedure. The database can be saved (to **Jobname.PDS**) or resumed at any time. The results of a probabilistic analysis are not stored in the database but in separate files. The samples generated for a fitted response surface are in neither the database nor in files, because they can be regenerated very quickly. (Files consume disk space, and reading the files requires as much time as regenerating the sample data.)
mean value	A measure of location often used to describe the general location of the bulk of the scattering data of a random output parameter or of a statistical distribution function. Mathematically, the mean value is the arithmetic average of the data. The mean value also represents the center of gravity of the data points. Another name for the mean value is the *expected value*.
median value	The statistical point where 50% of the data is below the median value and the 50% is above. For symmetrical distribution functions (Gaussian, uniform, etc.) the median value and the mean value are identical, while for nonsymmetrical distributions they are different.

PDS Term	Description
standard deviation	A measure of variability (dispersion or spread) about the arithmetic mean value, often used to describe the width of the scatter of a random output parameter or of a statistical distribution function. The larger the standard deviation, the wider the scatter and the more likely it is that there are data values further apart from the mean value.
solution set	The collection of results derived from the simulation loops performed for a given probabilistic analysis. The solution set includes the values of all random input variables and all random output parameters for all simulation loops of a probabilistic analysis. A unique label identifies each solution set.
response surface set	The collection of response surfaces derived from a fitting procedure (regression analysis) *and* the results derived from using the response surfaces for a probabilistic analysis. A response surface set is identified by a unique response surface label.
remote host	A computer in your local area network used to execute a probabilistic analysis in parallel mode. A remote host can have more than one CPU. In parallel processing, you can use multiple CPUs on remote hosts.

The following figure shows the flow of information during a probabilistic design analysis. Note that the analysis file must exist as a separate entity, and that the probabilistic design database is not part of the ANSYS model database.

Figure 3.2 Probabilistic Design Data Flow



3.3. Employing Probabilistic Design

You can approach an ANSYS probabilistic design as a batch run or interactively through the Graphical User Interface (GUI). The approach you take depends upon your experience and preference for interacting with the ANSYS program.

If you are familiar with ANSYS commands, you may want to perform the entire probabilistic design analysis by creating an ANSYS command input file and submitting it as a batch job. The command method may be more efficient for complex analyses (for example, nonlinear) requiring extensive run time.

The interactive features of probabilistic design offer flexibility and immediate feedback for review of loop results. When performing probabilistic design via the GUI, it is important to first establish the analysis file for your model.

The usual process for probabilistic design consists of the following general steps. The steps may vary slightly, depending on whether you are performing probabilistic design interactively (through the GUI) or in batch mode. The items in parentheses indicate which ANSYS processor is necessary to perform the given task.

1. Create an analysis file for use during looping. The file should represent a complete analysis sequence and must do the following:

    - Build the model parametrically (PREP7).
    - Obtain the solution(s) (SOLUTION).
    - Retrieve and assign to parameters the quantities that will be used as random input variables and random output parameters (POST1/POST26).

2. Establish parameters in the ANSYS database which correspond to those used in the analysis file. This step is typical, but not required (Begin or PDS); however, if you skip this step, then the parameter names are not available for selection in interactive mode.
3. Enter PDS and specify the analysis file (PDS).
4. Declare random input variables (PDS).
5. Visualize random input variables (PDS). Optional.
6. Specify any correlations between the RVs (PDS).
7. Specify random output parameters (PDS).
8. Choose the probabilistic design tool or method (PDS).
9. Execute the loops required for the probabilistic design analysis (PDS).
10. Fit the response surfaces (if you did not use a Monte Carlo Simulation method) (PDS).
11. Review the results of the probabilistic analysis (PDS).

Because analyzing complex problems can be time-consuming, ANSYS offers you the option of running a probabilistic analysis on a single processor or distributing the analyses across multiple processors. By using the ANSYS PDS parallel run capabilities, you can run many analysis loops simultaneously and reduce the overall run time for a probabilistic analysis.

3.3.1. Create the Analysis File

The analysis file is crucial to ANSYS probabilistic design. The probabilistic design system (PDS) uses the analysis file to form the loop file, which in turn is used to perform analysis loops. *Any type* of ANSYS analysis (structural, thermal, magnetic, etc.; linear or nonlinear) may be incorporated into the analysis file.

The model must be defined in terms of parameters (both RVs and RPs). Only numerical scalar parameters are used by the PDS. You can use up to a total of 5000 parameters (RVs and RPs together). See Section 5.10.6.2: Use ANSYS Parameters in the *ANSYS Modeling and Meshing Guide* for a discussion of parameters. See the *ANSYS APDL Programmer's Guide* for a discussion of the ANSYS Parametric Design Language (APDL).

It is your responsibility to create and verify the analysis file. It must represent a clean analysis that will run from start to finish. Most nonessential commands (such as those that perform graphic displays, listings, status requests, etc.) should be stripped off or commented out of the file. Maintain only those display commands that you want to see during an interactive session (such as **EPLOT**), or direct desired displays to a graphics file (**/SHOW**). Because the analysis file will be used iteratively during probabilistic design looping, any commands not essential to the analysis will decrease efficiency.

You can create an analysis file by inputting commands line by line via a system editor, or you can create the analysis interactively in the ANSYS program and use the ANSYS command log as the basis for the analysis file.

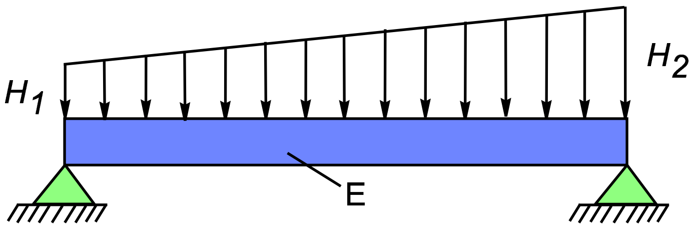
Creating the file with a system editor is the same as creating a batch input file for the analysis. (If you are performing the entire probabilistic design in batch mode, the analysis file is usually the first portion of the complete batch input stream.) This method allows you full control of parametric definitions through exact command inputs. It also eliminates the need to clean out unnecessary commands later. If you are unfamiliar with ANSYS commands, however, this method may be inconvenient.

You prefer to perform the initial analysis interactively, and then use the resulting command log as the basis for the analysis file. In this case, you must edit the log file to make it suitable for probabilistic design looping. For more information about using the log files, see Chapter 8, "Using the ANSYS Command Log" in the *ANSYS Operations Guide*.

3.3.1.1. Sample Problem Description

The simple beam problem introduced earlier illustrates a probabilistic design analysis.

Figure 3.3 A Beam Under a Snow Load



Young's modulus is 20E4.

3.3.1.2. Build the Model Parametrically

PREP7 is used to build the model in terms of the RV parameters. For our beam example, the RV parameters are *H1* (snow height at left end), *H2* (snow height at right end), and the Young's modulus *E*.

```
...
!  Initialize ANSYS parameters:
H1=100                       ! Initialize snow height H1 @ left end (in mm)
H2=100                       ! Initialize snow height H2 @ right end(in mm)
YOUNG=200.0e3                ! Initialize the Young's modulus (in N/mm**2)
ROOFWDT=1000.0               ! Initialize roof width left and right of beam (in mm)
BWDT=10.0                    ! Initialize beam width (in mm)
BHGT=40.0                    ! Initialize beam height (in mm)
BLEN=3000.0                  ! Initialize beam length (in mm)
SNOWDENS = 200e-9            ! Density of snow (200 kg/m**3)
GRAVACC  = 9.81              ! Earth gravity (in N/kg)
LOAD1 = H1*GRAVACC*ROOFWDT*SNOWDENS  ! Pressure load due to snow @ left end
LOAD2 = H2*GRAVACC*ROOFWDT*SNOWDENS  ! Pressure load due to snow @ right end
DELLOAD = LOAD2-LOAD1
!
! Material definitions:
MP,EX,1,YOUNG                ! Young's modulus
MP,PRXY,1,0.3                ! Poisson's ratio
!
! Create the geometry (a line)
K,1,0,0,0                    ! keypoint at left end
K,2,BLEN,0,0                 ! keypoint at right end
L,1,2,100                    ! line between keypoints
!
! Mesh definitions
ET,1,BEAM3                   ! 2-D beam element
AREA=BWDT*BHGT               ! Beam cross-sectional area
IZZ=(BWDT*(BHGT**3))/12      ! Moment of inertia about Z
R,1,AREA,IZZ,BHGT            ! Real constants in terms of RV parameters
LATT,1,1,1
LMESH,1                      ! mesh the line
FINISH                       ! Leave PREP7
...
```

As mentioned earlier, you can vary virtually any aspect of the design: dimensions, shape, material property, support placement, applied loads, etc. The only requirement is that the design be defined in terms of parameters. The RV parameters (*H1*, *H2*, and *E* in this example) may be initialized anywhere, but are typically defined in PREP7.

Caution: If you build your model interactively (through the GUI), you will encounter many situations where data can be input through graphical picking (such as when defining geometric entities). Because some picking operations do not allow parametric input (and PDS requires parametric input), you should avoid picking operations. Instead, use menu options that allow direct input of parameters.

3.3.1.3. Obtain the Solution

The SOLUTION processor is used to define the analysis type and analysis options, apply loads, specify load step options, and initiate the finite element solution. All data required for the analysis should be specified: master degrees of freedom in a reduced analysis, appropriate convergence criteria for nonlinear analyses, frequency range for harmonic response analysis, and so on. Loads and boundary conditions may also be RVs as illustrated for the beam example here.

The SOLUTION input for our beam example could look like this:

```
...
/SOLU
ANTYPE,STATIC                  ! Static analysis (default)
D,1,UX,0,,,,UY                 ! UX=UY=0 at left end of the beam
D,2,UY,0,,,,                   ! UY=0 at right end of the beam
!D,2,UX,0,,,,UY                ! UX=UY=0 at right end of the beam
elem=0
*get,numele,ELEM,,COUNT
*DO,i,1,numele
  elem=elnext(elem)            ! get number of next selected element
  node1=NELEM(elem,1)          ! get the node number at left end
  node2=NELEM(elem,2)          ! get the node number at right end
  x1 = NX(node1)               ! get the x-location of left node
  x2 = NX(node2)               ! get the x-location of rigth node
  ratio1 = x1/BLEN
  ratio2 = x2/BLEN
  p1 = LOAD1 + ratio1*DELLOAD  ! evaluate pressure at left node
  p2 = LOAD1 + ratio2*DELLOAD  ! evaluate pressure at left node
  SFBEAM,elem,1,PRES,p1,p2     ! Transverse pressure varying linearly
                               ! as load per unit length in negative Y
*ENDDO
SOLVE
FINISH                         ! Leave SOLUTION
...
```

This step is not limited to just one analysis. You can, for instance, obtain a thermal solution and then obtain a stress solution (for thermal stress calculations).

If your solution uses the multiframe restart feature, all changes to the parameter set that are made after the first load step will be lost in a multiframe restart. To ensure that the correct parameters are used in a multiframe restart, you must explicitly save (**PARSAV**) and resume (**PARESU**) the parameters for use in the restart. See the *ANSYS Basic Analysis Guide* for more information on multiframe restarts.

3.3.1.4. Retrieve Results and Assign as Output Parameters

This is where you retrieve results data and assign them to random output parameters to be used for the probabilistic portion of the analysis. Use the ***GET** command (**Utility Menu> Parameters> Get Scalar Data**), which assigns ANSYS calculated values to parameters, to retrieve the data. POST1 is typically used for this step, especially if the data are to be stored, summed, or otherwise manipulated.

In our beam example, the maximum deflection and the maximum stress of the beam are random output parameters (RPs). The parameters for these data may be defined as follows:

```
...
/POST1
SET,FIRST
NSORT,U,Y              ! Sorts nodes based on UY deflection
```

```
*GET,DMAX,SORT,,MIN      ! Parameter DMAX = maximum deflection
!
! Derived data for line elements are accessed through ETABLE:
ETABLE,VOLU,VOLU         ! VOLU = volume of each element
ETABLE,SMAX_I,NMISC,1    ! SMAX_I = max. stress at end I of each
                         !  element
ETABLE,SMAX_J,NMISC,3    ! SMAX_J = max. stress at end J of each
                         !  element
!
ESORT,ETAB,SMAX_I,,1     ! Sorts elements based on absolute value
                         !  of SMAX_I
*GET,SMAXI,SORT,,MAX     ! Parameter SMAXI = max. value of SMAX_I
ESORT,ETAB,SMAX_J,,1     ! Sorts elements based on absolute value
                         !  of SMAX_J
*GET,SMAXJ,SORT,,MAX     ! Parameter SMAXJ = max. value of SMAX_J
SMAX=SMAXI>SMAXJ         ! Parameter SMAX = greater of SMAXI and
                         !  SMAXJ, that is, SMAX is the max. stress
FINISH
...
```

See the ***GET** and **ETABLE** commands for more information.

3.3.1.5. Prepare the Analysis File

If you choose to create your model interactively in ANSYS, you must now derive the analysis file from the interactive session. Use the command log or the session log file to do so. For more information on using these log files, see Chapter 8, “Using the ANSYS Command Log” in the *ANSYS Operations Guide*.

Note — Do not use the **/CLEAR** command in your analysis file as this will delete the probabilistic design database during looping. If this happens, the random input variables are no longer recognized during looping and you will get the same (deterministic) results for all simulation loops. However resume the ANSYS database using the **RESUME** command as part of your analysis file. For example, this is helpful if the variations of the random input variables do not require that meshing is done in every loop (because the mesh is not effected). In this case you can mesh your model, save the ANSYS database, and resume the database at the beginning of the analysis file.

3.3.2. Establish Parameters for Probabilistic Design Analysis

After completing the analysis file, you can begin the probabilistic design analysis. (You may need to reenter ANSYS if you edited the analysis file at the system level.)

When performing probabilistic design interactively, it is advantageous (but optional) to first establish the parameters from the analysis file in the ANSYS database. (It is unnecessary to do so in batch mode.)

To establish the parameters in the database:

- Resume the database file (**Jobname.DB**) associated with the analysis file. This establishes your entire model database in ANSYS, including the parameters. To resume the database file:
  Command(s): RESUME
  GUI: Utility Menu> File> Resume Jobname.db
  Utility Menu> File> Resume from
- Read the analysis file into ANSYS to perform the complete analysis. This establishes your entire model database in ANSYS, but might be time-consuming for a large model. To read the analysis file:
  Command(s): /INPUT
  GUI: Utility Menu> File> Read Input from

- Restore only the parameters from a previously saved parameter file; that is, read in a parameter file that you saved using either the **PARSAV** command or the **Utility Menu> Parameters> Save Parameters** menu path. To resume the parameters:
  Command(s): PARRES
  GUI: Utility Menu> Parameters> Restore Parameters
- Recreate the parameter definitions as they exist in the analysis file. Doing this requires that you know which parameters were defined in the analysis file.
  Command(s): *SET
  GUI: Utility Menu> Parameters> Scalar Parameters

You may choose to do none of the above, and instead use the **PDVAR** command to define the parameters that you declare as probabilistic design variables. See Section 3.3.4: Declare Random Input Variables for information on using **PDVAR**.

Note — The ANSYS database does not need to contain model information corresponding to the analysis file to perform probabilistic design. The model input is automatically read from the analysis file during probabilistic design looping.

3.3.3. Enter the PDS and Specify the Analysis File

The remaining steps are performed within the PDS processor.
Command(s): /PDS
GUI: Main Menu> Prob Design

In interactive mode, you must specify the analysis file name. This file is used to derive the probabilistic design loop file **Jobname.LOOP**. The default for the analysis file name is **Jobname.pdan**. You can also specify a name for the analysis file:
Command(s): PDANL
GUI: Main Menu> Prob Design> Analysis File> Assign

For a probabilistic design run in batch mode, the analysis file is usually the first portion of the batch input stream, from the first line down to the first occurrence of **/PDS**. In batch mode, the analysis file name defaults to **Jobname.BAT** (a temporary file containing input copied from the batch input file). Therefore, you normally do not need to specify an analysis filename in batch mode. However, if for some reason you have separated the batch probabilistic design input into two files (one representing the analysis and the other containing all probabilistic design operations), then you will need to specify the analysis file using **PDANL** after entering the PDS (**/PDS**).

Note — In the analysis file, the **/PREP7** and **/PDS** commands must occur as the *first nonblank characters on a line*. (Do not use the $ delimiter on a line containing either of these commands.) This requirement is necessary for proper loop file construction.

You cannot assign a different analysis file using the **PDANL** command after a probabilistic analysis has been performed. This ensures the integrity of the previously generated results with the specified probabilistic model.

Of course, ANSYS cannot restrain you from editing the analysis file or exchanging it with system level commands. If you do so, then it is your responsibility to ensure the integrity of the generated results with the definitions in the analysis file. If you are not sure that this integrity is maintained or if you know that it is not, then we recommend that you save the current PDS database via the **PDSAVE** command and then clear the probabilistic analysis results from the probabilistic design database using the **PDCLR**, POST command. The **PDCLR** command does not delete the result files that have been generated; it erases the link to the result files from the database.

In the example of a beam supporting a roof with a snow load you could store the analysis file in a macro called "beam.mac". Here, the analysis is specified with the commands:

```
...
/PDS
PDANL,beam,mac
...
```

3.3.4. Declare Random Input Variables

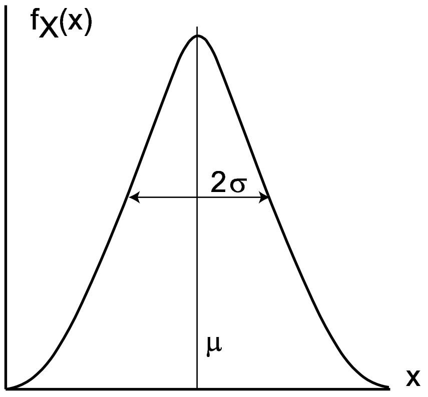
The next step is to declare random input variables, that is, specify which parameters are RVs. As mentioned earlier, the PDS allows for a combined total of 5000 random input variables and random output parameters. To declare random input variables:

Command(s): PDVAR
GUI: Main Menu> Prob Design> Prob Definitns> Random Input

If the parameter name that you specify with the **PDVAR** command is not an existing parameter, the parameter is automatically defined in the ANSYS database with an initial value of zero.

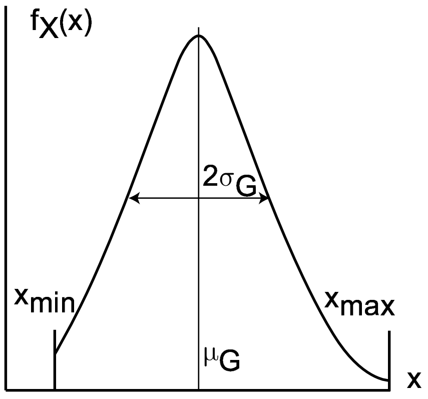
For random input variables you must specify the type of statistical distribution function used to describe its randomness as well as the parameters of the distribution function. For the distribution type, you can select one of the following:

- Gaussian (Normal) (GAUS):



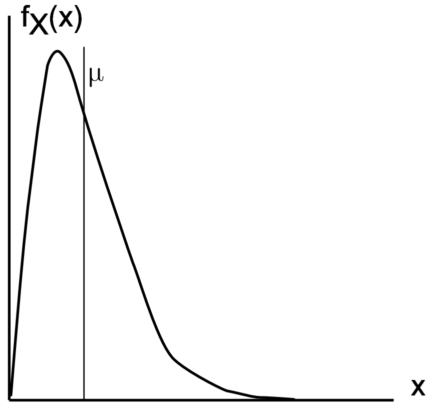
You provide values for the mean value μ and the standard deviation σ of the random variable *x*.

- Truncated Gaussian (TGAU):



You provide the mean value μ and the standard deviation σ of the non-truncated Gaussian distribution and the truncation limits x_{min} and x_{max}.

- Lognormal option 1 (LOG1):

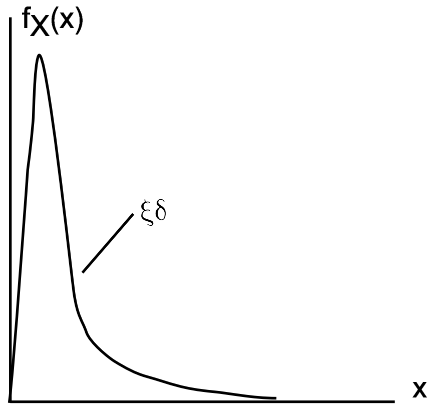


You provide values for the mean value μ and the standard deviation σ of the random variable *x*. The PDS calculates the logarithmic mean ξ and the logarithmic deviation δ:

$$f(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi} \cdot x \cdot \sigma} \cdot \exp\left(-\frac{1}{2}\left(\frac{\ln x - \xi}{\delta}\right)^2\right)$$

$$\delta = \sqrt{\ln\left[\left(\frac{\sigma}{\mu}\right)^2 + 1\right]} \text{ and } \xi = \ln \mu - 0.5 \cdot \delta$$

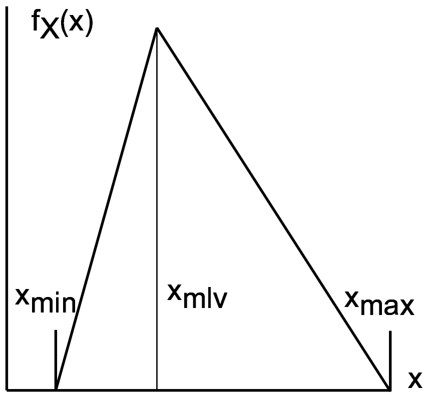
- Lognormal option 2 (LOG2):



You provide values for the logarithmic mean value ξ and the logarithmic deviation δ. The parameters ξ and δ are the mean value and standard deviation of ln(*x*):

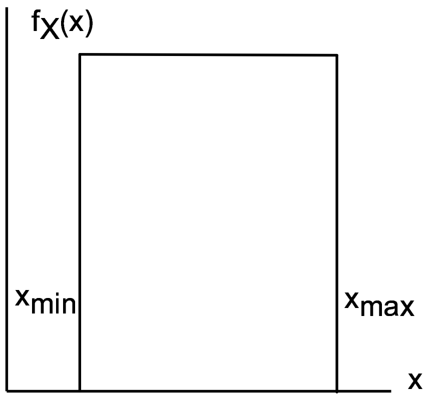
$$f(x,\xi,\delta) = \frac{1}{\sqrt{2\pi} \cdot x \cdot \sigma} \cdot \exp\left(-\frac{1}{2}\left(\frac{\ln x - \xi}{\delta}\right)^2\right)$$

- Triangular (TRIA):



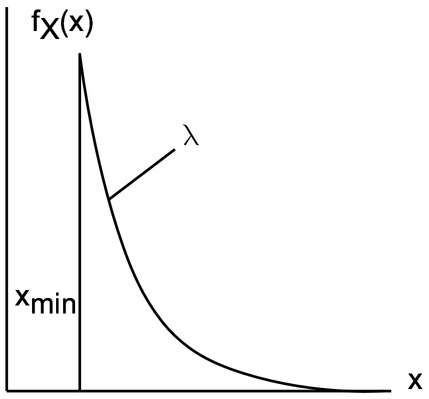
You provide the minimum value x_{min}, the most likely value limit x_{mlv} and the maximum value x_{max}.

- Uniform (UNIF):



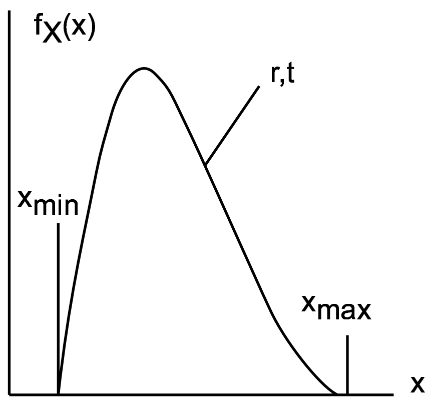
You provide the lower and the upper limit x_{min} and x_{max} of the random variable x.

- Exponential (EXPO):



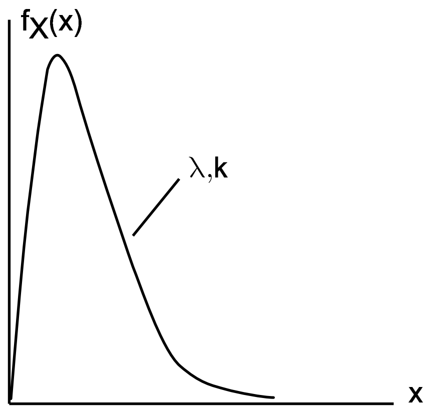
You provide the decay parameter λ and the shift (or lower limit) x_{min} of the random variable x.

- Beta (BETA):



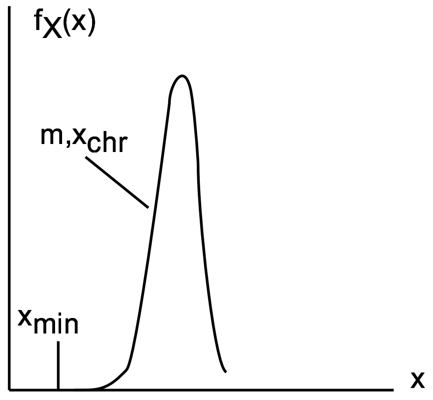
You provide the shape parameters *r* and *t* and the lower and the upper limit x_{min} and x_{max} of the random variable *x*.

- Gamma (GAMM):



You provide the decay parameter λ and the power parameter *k*.

- Weibull (Type III smallest) (WEIB):



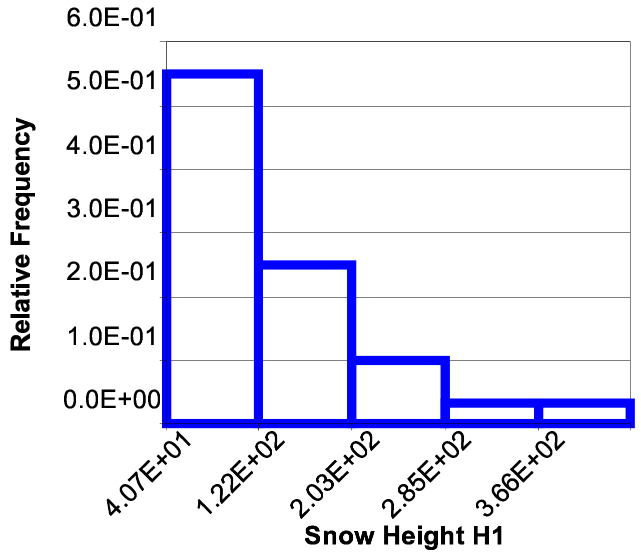
You provide the Weibull characteristic value x_{chr}, the Weibull exponent m and the minimum value x_{min}. Special cases: For $x_{min} = 0$ the distribution coincides with a two-parameter Weibull distribution. The Rayleigh distribution is a special case of the Weibull distribution with $\alpha = x_{chr} - x_{min}$ and $m = 2$.

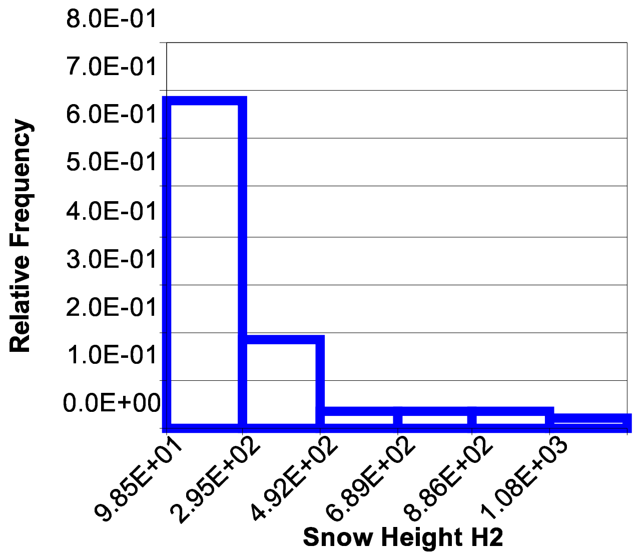
You may change the specification of a previously declared random input variable by redefining it. You may also delete a probabilistic design variable (**PDVAR**,*Name*,DEL). The delete option does not delete the parameter from the database; it simply deactivates the parameter as a probabilistic design variable.

Note — Changing the probabilistic model by changing a random input variable is not allowed after a probabilistic analysis has been performed. This ensures the integrity of the previously generated results with the specified probabilistic model. If you need to change one or more random input variables (for example, because you learned that some specifications were incorrect after running an analysis), then we recommend that you save the current PDS database (using the **PDSAVE** command) and then clear the probabilistic analysis results from the probabilistic design database (using the **PDCLR**,POST command). The **PDCLR** command does not delete the result files that have been generated, it simply removes the link to the results file from the database.

In the example of a beam supporting a roof with a snow load, you could measure the snow height on both ends of the beam 30 different times. Suppose the histograms from these measurements look like the figures given below.

Figure 3.4 Histograms for the Snow Height H1 and H2





From these histograms you can conclude that an exponential distribution is suitable to describe the scatter of the snow height data for H1 and H2. Suppose from the measured data we can evaluate that the average snow height of H1 is 100 mm and the average snow height of H2 is 200 mm. The parameter λ can be directly derived by 1.0 divided by the mean value which leads to λ_1 = 1/100 = 0.01 for H1, and λ_1 = 1/200 = 0.005 for H2. From measurements of the Young's modulus you see that the Young's modulus follows a Gaussian distribution with a standard deviation of 5%. Given a mean value of 200,000 N/mm^2 for the Young's modulus this gives a standard deviation of 10,000 N/mm^2. These definitions can be specified using the following commands:

```
...
PDVAR,H1,EXPO,0.01
PDVAR,H2,EXPO,0.005
PDVAR,YOUNG,GAUS,200000,10000
...
```

3.3.5. Visualize Random Input Variables

After you define your random input variables, you should use ANSYS' visual tools to verify them. You can plot individual RVs, and you can obtain specific information about them through an inquiry.

Command(s): PDPLOT, **PDINQR**
GUI: Main Menu> Prob Design> Prob Definitns> Plot
Main Menu> Prob Design> Prob Definitns> Inquire

The **PDPLOT** command plots the probability density function as well as the cumulative distribution function of a defined random input variable. This allows you to visually check that your definitions are correct.

Use the **PDINQR** command to inquire about specific information for a defined random input variable by retrieving statistical properties or probing the two function curves that are plotted with the **PDPLOT** command. For example you can inquire about the mean value of a random input variable or evaluate at which value the cumulative distribution function reaches a certain probability. The result of this inquiry is stored in a scalar ANSYS parameter.

3.3.6. Specify Correlations Between Random Variables

In a probabilistic design analysis, random input variables can have specific relationships to each other, called correlations. If two (or more) random input variables are statistically dependent on each other, then there is a correlation between those variables.

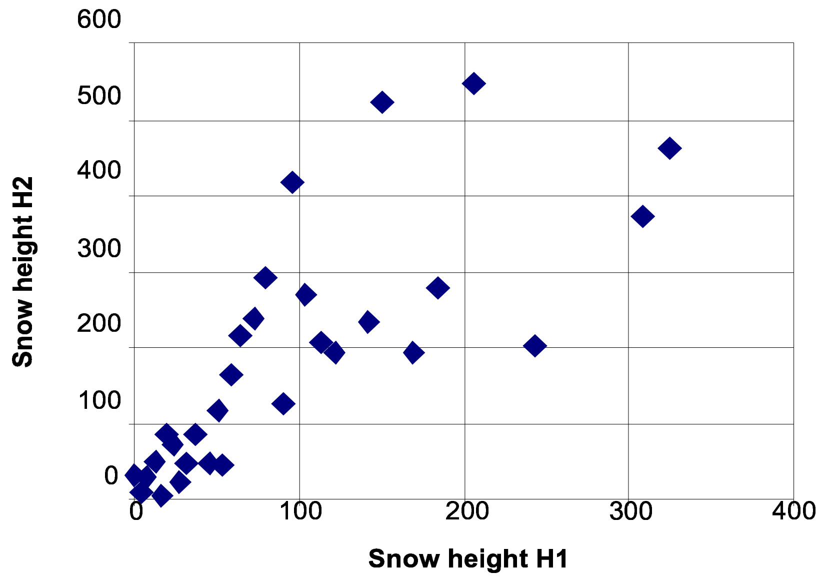
To define these correlations:

Command(s): PDCORR
GUI: Main Menu> Prob Design> Prob Definitns> Correlation

You specify the two random input variables for which you want to specify a correlation, and the correlation coefficient (between -1 and 1). To remove a correlation, enter DEL in the correlation coefficient field (**PDCORR**,*Name1*,*Name2*,DEL)

In the example of a beam supporting a roof with a snow load, the data for the snow height indicates that there is a correlation between the snow height at one end versus the snow height at the other end. This is due to the fact that it is unlikely that one end of the beam has very little snow (or no snow at all) at the same time that the other end carries a huge amount of snow. In the average the two snow heights tend to be similar. This correlation is obvious if you plot the measured data values for H2 versus the data value for H1. This scatter plot looks like this:

Figure 3.5 A Scatter Plot of Snow Height H1 vs. H2



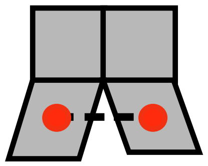
Performing a statistical evaluation of the data, we can conclude that the linear correlation coefficient between the values of H1 and H2 is about 0.8. You can define this correlation using the commands:

```
...
PDVAR,H1,EXPO,0.01
PDVAR,H2,EXPO,0.005
PDCORR,H1,H2,0.8
...
```

You may have a more complex correlation where you have a spatial dependency. If so, you can use the **PDCFLD** command to calculate a correlation field and store it into an ANSYS array.

Random fields are random effects with a spatial distribution; the value of a random field not only varies from simulation to simulation at any given location, but also from location to location. The correlation field describes the correlation coefficient between two different spatial locations. Random fields can be either based on element properties (typically material) or nodal properties (typically surface shape defined by nodal coordinates). Hence, random fields are either associated with the selected nodes or the selected elements. If a random field is associated with elements, then the correlation coefficients of the random field are calculated based on the distance of the element centroids.

Note that for correlation fields, the "domain distance" D($\{x_i\}$, $\{x_j\}$) is not the spatial distance $|\{x_i\} - \{x_j\}|$, but the length of a path between $\{x_i\}$ and $\{x_j\}$ that always remains inside the finite element domain. However, exceptions are possible in extreme meshing cases. For elements that share at least one node, the **PDCFLD** evaluates the distance by directly connecting the element centroids with a straight line. If these neighboring elements form a sharp inward corner then it is possible that the "domain distance" path lies partly outside the finite element domain, as illustrated below.

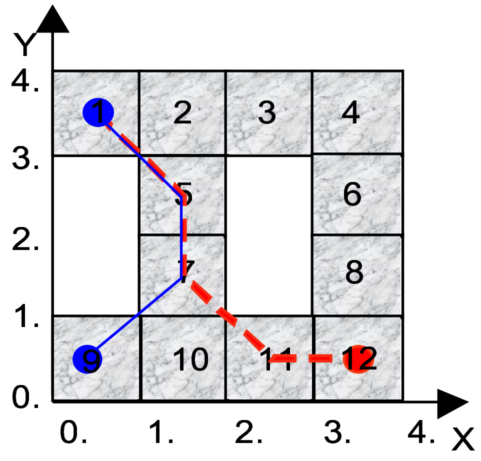


After the correlation coefficients have been calculated and stored in an ANSYS parameter (**PDCFLD**,*ParR*), then you can use the **PDCORR** command to define the correlations between the elements of the random field.

> *Note* — When specifying one variable (A) with correlations to two or more other variables (B, C, etc.), be certain that you consider the relationship implied between the other variables B and C, etc. If you specify high correlations between A and B and A and C, without specifying the relationship between B and C, you might receive an error. Specifying a relatively high correlation between A and B, with only a moderate correlation between A and C might work because the logical correlation between B and C could still be low or nonexistent.

Example

The structure illustrated below is modeled with 12 elements. We will evaluate the domain distances of the element centroids.



First, build the finite element structure:

```
...
/PREP7
et,1,shell63
! create the nodes
N,1,0,4
N,2,1,4
N,3,2,4
N,4,3,4
N,5,4,4
N,6,0,3
N,7,1,3
N,8,2,3
N,9,3,3
N,10,4,3
N,11,1,2
N,12,2,2
N,13,3,2
N,14,4,2
N,15,0,1
N,16,1,1
N,17,2,1
N,18,3,1
N,19,4,1
N,20,0,0
N,21,1,0
N,22,2,0
N,23,3,0
N,24,4,0
! create the elements
E,1,2,7,6
```

```
E,2,3,8,7
E,3,4,9,8
E,4,5,10,9
E,7,8,12,11
E,9,10,14,13
E,11,12,17,16
E,13,14,19,18
E,15,16,21,20
E,16,17,22,21
E,17,18,23,22
E,18,19,24,23
...
```

Next, calculate the domain distances and store the results in the array “elemdist”:

```
...
/PDS
PDCFLD,elemdist,ELEM,DIST
...
```

Finally, get all the element domain distances and print them:

```
...
*GET,numsel,ELEM,0,COUNT    ! Get the number of selected elements
!
! Outer loop through all selected elements from first to last
index=0
elem1=0
! Pipe output to file
/OUT,elements,dat
*DO,i,1,numsel
  elem1=ELNEXT(elem1)       ! get number of next selected element
  *IF,elem1,EQ,0,CYCLE      ! Leave do loop if no more elements
  !
  ! Inner loop through selected elements from "elem1+1" to last
  elem2=elem1
  *DO,j,i+1,numsel
    elem2=ELNEXT(elem2)     ! get number of next selected element
    *IF,elem2,EQ,0,CYCLE    ! Leave do loop if no more elements
    index=index+1
    !
    ! Print out the element distance
    *MSG,INFO,elem1,elem2,elemdist(index)
    Distance between element %i and %i is %g
  *ENDDO                     ! go to next element for inner loop
*ENDDO                       ! go to next element for outer loop
...
```

The print out will show that for the structure illustrated above the "domain distance" between the element centroids of elements 1 and 9 is 3.8284 and between the element centroids of elements 1 and 12 it is 4.8284. The paths related to these distances are sketched in the illustration with a solid line and a dashed line respectively. In this example there are 12 elements, thus the array "elemdist" has a length of 12*(12-1) = 66.

3.3.7. Specify Random Output Parameters

After declaring your input variables and correlations among them you must define the random output parameters. The random output parameters (RPs) are results parameters that you are interested in. To define random output parameters:

Command(s): PDVAR,*Name*,RESP
GUI: Main Menu> Prob Design> Prob Definitns> Random Output

3.3.8. Choose a Probabilistic Design Method

In the Probabilistic Design System, several probabilistic design methods are available.

You can select one of two *primary* methods, the Monte Carlo Simulation (default) or the Response Surface Method.

Options under the Monte Carlo Simulation method include the Latin Hypercube Sampling method (default) and the Direct Monte Carlo Sampling method.

Options under the Response Surface Method include the Central Composite Design and the Box-Behnken Matrix Design method.

Both the Monte Carlo Simulation and the Response Surface Methods allow a user-defined option. See the **PDDMCS**, **PDLHS**, and the **PDDOEL**, commands or Section 3.5: Probabilistic Design Techniques for further details about these methods.

To specify a method to be used for probabilistic design looping:

Command(s): PDMETH
GUI: Main Menu> Prob Design> Prob Method> Monte Carlo Sims
Main Menu> Prob Design> Prob Method> Response Surface

Note — To use Response Surface Methods, the random output parameters must be smooth and continuous functions of the involved random input variables. Do not use Response Surface Methods if this condition is not satisfied.

3.3.8.1. Probabilistic Method Determination Wizard

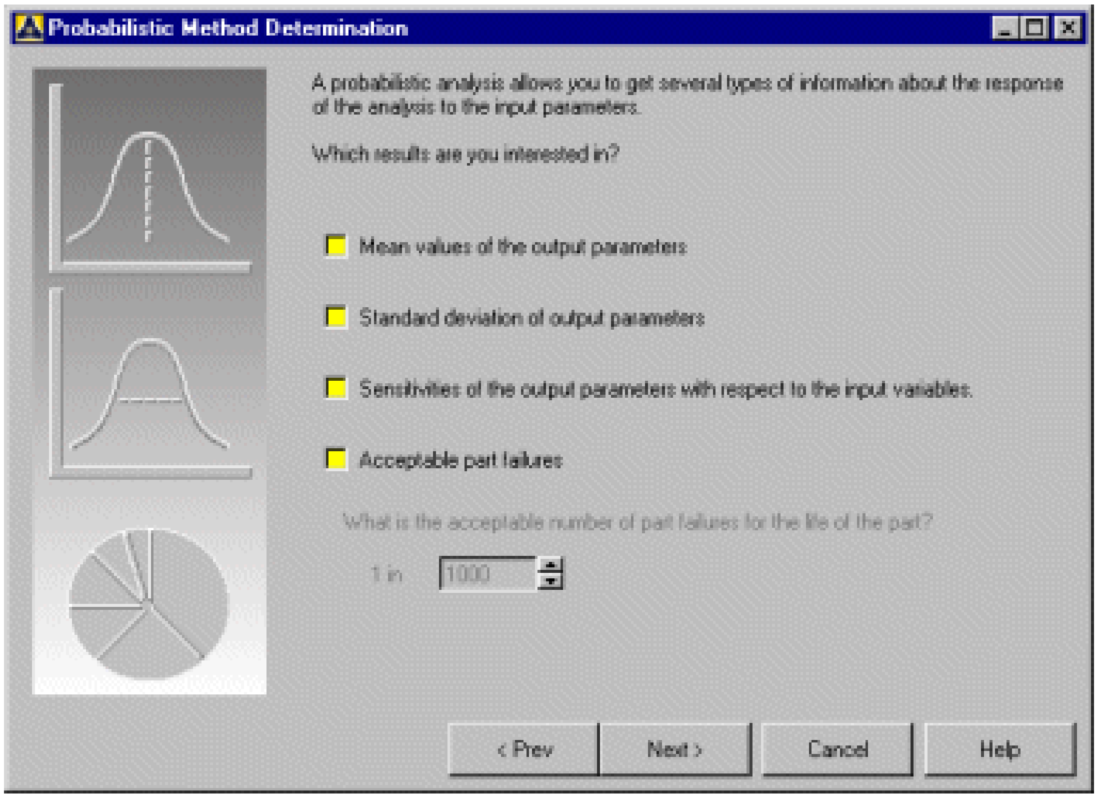
You can use the Probabilistic Method Determination Wizard to find the fastest method for solving your probabilistic analysis. You should have completed one analysis prior to starting the wizard.

Use **Main Menu> Prob Design> Prob Method> Methods Wizard** to access the wizard. Answer the questions on the screens as they are presented. Before you start the wizard, you should know:

- How long did it take to run the analysis file (hours, minutes, seconds)? Or, you should give an estimation of the time it will take if you haven't run the analysis yet.
- How many CPUs are available to run the probabilistic analysis (if running parallel processing)?
- Is your output parameter a smooth/continuous function of the input parameters?
- What results are you interested in evaluating? Options are mean values, standard deviation, sensitivities, or acceptable part failures.

Below is one of the wizard screens, as an example.

Figure 3.6 The PDS Method Determination Wizard



A probabilistic analysis allows you to get several types of information about the response of the analysis to the input parameters.

Which results are you interested in?

- Mean values of the output parameters
- Standard deviation of output parameters
- Sensitivities of the output parameters with respect to the input variables.
- Acceptable part failures

What is the acceptable number of part failures for the life of the part?

1 in 1000

Based on the information you provide, the wizard will tell you the fastest method for solving your probabilistic design problem. The wizard will issue the **PDMETH** command and either the **PDLHS** or the **PDDOEL** command. You will still need to run the analysis, then fit the results to a response surface, etc. to evaluate your results.

3.3.9. Execute Probabilistic Analysis Simulation Loops

You can execute your analysis on your computer alone (serial execution), or using other computers in your network to save running time and speed processing (parallel execution).

If you want to use serial processing only, select the serial option from the Run menu.

If you want to run parallel analyses on multiple CPUs, you must first set the parallel options before performing the analysis. (See Section 3.3.9.3: PDS Parallel Analysis Runs for more details).

Command(s): PDEXE
GUI: Main Menu> Prob Design> Run> Exec Serial> Run Serial
Main Menu> Prob Design> Run> Exec Parallel> Run Parallel

Caution: For security reasons ANSYS strongly recommends that you use parallel processing only within the firewall of your local area network.

If you choose serial processing to perform a probabilistic analysis then you will utilize only the CPU of the computer you are working on. If you have access to only one license of ANSYS or if you have access to only one computer, then this is the only way in which you can run a probabilistic analysis. While the simulation loops are running in serial mode, your ANSYS session is locked (you cannot perform other tasks in the same ANSYS session). If you are running your ANSYS session in interactive mode then the simulation loops are also performed in interactive mode. If you are running your ANSYS session in batch mode then the simulation loops are performed in batch mode.

If you choose the PDS parallel-processing option, you can use other CPUs that you have access to for running the probabilistic analysis. PDS parallel processing can distribute the necessary jobs in a local area network. With this option, the simulation loops are sent to CPUs that you can specify, where they are executed in "server mode." This looks the same as a batch run (in other words, there is no interactive visualization during the execution of a simulation loop). While the simulation loops are running in parallel mode, your ANSYS session is locked; however, you *can* instruct the ANSYS session to start postprocessing the probabilistic results as they are calculated so you can review and visualize the results before all simulation loops are finished.

In parallel processing, you can monitor the running jobs and the completed results.

3.3.9.1. Probabilistic Design Looping

Regardless of whether you opt for serial or parallel processing, ANSYS controls the analysis looping; you cannot modify this process. It does the following:

- Always reads the analysis file from the beginning.
- Always ignores the RV settings and replaces their value with the derived value that the probabilistic method has assigned to an RV for a particular loop.
- Always reads the PDS database after each loop.

For the execution of the simulation loops you must specify a solution label (*Slab* on the **PDEXE** command). The solution label is used for several purposes:

- The results are stored in an ASCII readable file under the name "jobname_*Slab*.pdrs". Here, *Slab* is the user-specified solution label.
- If you want to fit response surfaces to the results of the random output parameters then you need to specify the solution label to indicate which set of results you want to use for the fitting procedure.
- If you want to postprocess the results generated with the **PDEXE** command then you must specify the solution label to indicate which results you want to postprocess.

When you execute the probabilistic analysis (**PDEXE**), ANSYS creates a probabilistic design loop file (**Jobname.LOOP**) from the analysis file. This loop file is used to perform analysis loops. Looping continues until all parameters have been evaluated.

If a loop is interrupted due to an error in the execution run (for example, a meshing failure, a non-converged nonlinear solution, etc.), ANSYS PDS aborts that loop. Further processing depends if you are in serial or parallel processing mode. If you are using:

- Serial interactive processing: you can choose to terminate when you receive the error, or continue processing.
- Serial batch processing: processing terminates at the first error.
- Parallel processing: processing terminates if the allowed number of failed loops is exceeded (set in **PDEXE**), otherwise it continues.

Note that for all failed loops (loops with errors), the results for that loop are discarded, no data from that loop is used for post processing.

After the **PDEXE** command is issued, the PDS generates a file containing the input sample values. The file is called **jobname.samp**. An example of the content of the file is given below:

```
TEST1
ITER CYCL     LOOP                X1                X2                X3
   1    1        1  1.619379209e+000  2.364528435e-001  1.470789050e+000
```

```
      1    1        2  2.237676559e-001  5.788049712e-001  1.821263115e+000
      1    1        3  7.931615474e+000  8.278689033e-001  2.170793522e+000
     ..   ..       ..               ...               ...               ...
     ..   ..       ..               ...               ...               ...
```

The first line contains the solution label (the parameter `Slab` is set via the **PDEXE** command); the second line contains the headers of the data columns - the iteration number, cycle number, loop number, and the random variable names. The iteration number and cycle number tell the PDS to which group (with specific PDS method and settings) the loops belong. Subsequent lines provide specific iteration, cycle, loop, and input sample values for the defined input variables.

The PDS also creates a file where the results are stored. The name of the results file is **jobname_*Slab*.pdrs**. Before the job is executed, the file looks like this:

```
TEST1
  ITER CYCL     LOOP ERR           X1           X2           X3       RESULT
```

In the first line, the PDS enters the solution label. In the second line are the headers for the data columns: the first four columns are the iteration, cycle number, loop number, and an error flag. The fifth and subsequent columns are for the random input variable and random output parameter values. If you run a subsequent analysis (same type of analysis with the same solution label), the cycle is incremented, the loop count is reset to 1, and the result file is appended.

For example, the content of the result file could look like this:

```
TEST1
ITER CYCL     LOOP ERR                X1                X2                X3            RESULT
   1    1        1   0  1.619379209e+000  2.364528435e-001  1.470789050e+000  4.162928057e+000
   1    1        2   0  2.237676559e-001  5.788049712e-001  1.821263115e+000  4.744249212e+000
   1    1        3   0  7.931615474e+000  8.278689033e-001  2.170793522e+000  1.149997825e+001
  ..   ..       ..  ..               ...               ...               ...               ...
  ..   ..       ..  ..               ...               ...               ...               ...
```

Note — Loops ending with an ANSYS error are deemed “not trustworthy”, i.e. if the loop lead to an error, then the calculated results are probably wrong. Those loops will have the error flag in the fourth column set to “1” instead of “0”. Those loops will be excluded from the probabilistic post-processing altogether, i.e. the loops will not be used for the response surface fitting and also the statistical analysis in connection with a Monte Carlo simulation will skip those loops.

3.3.9.2. Serial Analysis Runs

Serial execution of the probabilistic analysis is useful for smaller models, or if you have access to only one machine.

Command(s): PDEXE
GUI: Main Menu> Prob Design> Run> Exec Serial> Run Serial

3.3.9.3. PDS Parallel Analysis Runs

To save time when analyzing large models with PDS, use the parallel processing option available under the **PDEXE** command, if you have multiple CPUs available.

To successfully execute a probabilistic analysis in parallel mode you need to follow three steps:

1. Configure the remote machines that you want to use for parallel processing. See Section 3.3.9.3.1: Machine Configurations for details. You must have account privileges on all remote hosts.
2. Configure the local machine that you want to use for managing the parallel processing. See Section 3.3.9.3.1.4: Configuring the Master Machine for details.

3. Start the parallel execution.

Each of these steps is explained in detail in the following sections.

An understanding of the following terms will be useful in the discussion of the parallel analysis.

- **Simulation:** A simulation is a set of input variables used with the analysis file to produce the output variables
- **Parent Process:** The parent process is the ANSYS executable which manages the creation of input parameters, communication with child processes, and postprocessing of output parameters (i.e. this is where the ANSYS session runs, which you started to perform a probabilistic design analysis). There is only one parent process for a parallel analysis.
- **Child Process:** A child process is an ANSYS server running on a particular slave machine which processes a simulation. There can be any number of children for a given parent process.
- **MasterMachine:** This is the machine on which you are running the ANSYS parent process. If this machine has more than one CPU you may want to also use it as a slave machine.
- **SlaveMachine:** This is the name of the machine on which you are running the ANSYS child process. If the slave machine has more than one CPUyou may want to run more than one child processl.
- **ANSYS Nanny:** This is a program in the parent process that starts a child process on a slave machine; manages the transfer of files and data between the parent process and child processes and eventually terminates the child processes on the various slave machines *and removes temporary files*.
- **ANSYS Thin Server:** This is an application, which must be running on the slave machine, which is responsible for tranferring files between the master machine and slave machine as well as starting an ANSYS session for each child process. This application may be started automatically (remote shell option) or you can start it manually (connection port option).
- **ANSYS Server:** This is the ANSYS application using TCP/IP which recieves commands and return information to the parent process (client-server).
- **Connection Port:** This is the port on which the ANSYS Thin Server accepts connections from a Master Machine.
- **Communication Port:** This is a port on which the ANSYS Thin Server will allow communications from a Master Machine.

Caution: For security reasons, ANSYS recommends that you use parallel processing within your local network protected by a firewall. There is minimal security when using the ANSYS Thin Server. The client that connects to the server has all of the permissions allowed by the person or account starting the ANSYS Thin Server.

3.3.9.3.1. Machine Configurations

The slave machines host one or more child processes. To be able to host a child process the slave machine must have the ANSYS Thin Server running. The ANSYS Thin Server may be started automatically using the remote shell option or manually using the connection port option.

3.3.9.3.1.1. Choosing Slave Machines

You can designate any machine that has a licensed copy of ANSYS installed as a remote host (slave machine) but it is a good practice to follow these guidelines when selecting remote hosts:

- Use idling computer power. During regular work hours, computers usually have designated "owners" using the machines for interactive work. Using these computers for parallel processing can slow down the machine performance and possibly disturb your colleagues in their work. However, computers are usually idle at night and can then be used without disturbing others.

- Use machines that are dedicated as computer servers. There might be some computers in your work environment that are not used for interactive work at all. It is fairly safe to use these computers as remote hosts for parallel-processing at all times.
- Check with your System Administrator for additional guidelines.

3.3.9.3.1.2. Using the Remote Shell Option

With this option the ANSYS Thin Server is started and configured to communicate with the Master machine automatically. To use the remote shell option requires the following on each slave machine:

- A remote shell daemon must be running
- An account that the master machines remote shell command can communicate with
- The master machine and user name must be granted access for the account
- The installation path for the ANSYS executable must be in the lookup path
- A valid ANSYS license available for the child process

Remote Shell Daemon

Your system administrator should install and configure your machine to assure that this daemon or service is running. The PC does not ship with a remote shell daemon facility so you will need to acquire one (you may use the connection port ooption instead).

Account

Your system administrator should create an account specifically for you or an account utilized only for PDS parallel.

Account Access

Depending on how and which remote shell daemon is installed it will need to know which master machines and users can access this account. This is typically done using the .rhosts file located in the account's HOME directory, refer to the documentation for your particular remote shell daemon. So for example:

On all slave machines edit/create the ".rhosts"-file in your home directory to include:

```
MasterMachine1 UserId
MasterMachine2 UserId
MasterMachine3 UserId
MasterMachine4 UserId
…
```

each on a separate line. This will allow the user with the account user identification UserId to access the slave machine from the master machines listed in the file using the remote shell service.

ANSYS in PATH

To be able to run the parallel process the ANSYS executable and ANSYS Thin Server script must be in your path. The most effective way to determine this is to do the following.

Determining if the ANSYS executable is in your path:

On the master machine use the command "rsh SlaveMachine which ansys81" (replace rsh by remsh if master machine is an HP machine). This should return the string

```
…/ansys81/bin
```

Here "…" is for the directory where ANSYS is installed. If you do not get this then you need to modify your PATH variable to contain the installation path. The way this is done depends on the operating system being used, refer to the documentation for your particular O/S environment. Here are some examples for typical uses:

UNIX

To change the PATH requires you to modify a certain system file depending which shell you are running under. To find out which shell you are using, issue the UNIX command

```
echo $shell or echo $SHELL
```

If the prompt is "/bin/csh" or similar, then you are running under c-shell. If the prompt is "/bin/ksh" or similar, then you are running under k-shell. If the prompt is "/bin/tcsh" or similar, then you are running under tc-shell, which can be treated the same as c-shell.

If you are running under c-shell (or tc-shell) you need include the following line at the end of your ".cshrc"-file in your home directory:

```
set path=( .../ansys81/bin $path)
```

If you are running under k-shell you need include the following line at the end of your ".kshrc"-file in your home directory:

```
export PATH={ .../ansys81/bin $PATH }
```

If you don't have a ".cshrc"-file or ".kshrc"-file in your home directory, then you need to create one and include the respective commands mentioned above.

PC

To change the PATH variable go to Control Panel and choose System on the Advanced tab choose Environment Variables and add the ANSYS executable to the PATH under System variables.

Determining if the ANSYS Thin Server script is in your path:

On the master machine issue the command "rsh SlaveMachine which ansysts81". This should return the string

```
…/ansys81/bin
```

Here "…" is for the directory where ANSYS is installed. If you do not get this then you need to modify your PATH variable as outlined above.

Licensing

If the licensing is not managed from a license server, but installed locally then the access to the license must be made available. Include the following line at the end of your ".cshrc"-file or ".kshrc-file in your home directory depending which shell you are running under (see above):

```
setenv ANSYSLMD_LICENSE_FILE. ../shared_files/licensing/FileName.lic
```

Here "…" is for the directory where ANSYS is installed. The name "FileName.lic" is for the file containing the license key.

3.3.9.3.1.3. Using the Connection Port Option

With this option the communication between the ANSYS Nanny and the ANSYS Thin Server is manually configured and started. The ANSYS Nanny running on a master machine can only make a connection to an ANSYS Thin Server using a specific connection port. The ANSYS Nanny reads the connection port from the "hosts81.ans"-file. When the ANSYS Thin Server is started, with the same specific connection port, it reads the file "AnsysClients" from the same directory it is started within to determine the communication ports and master machines with which it will communicate.

The purpose of these two different port numbers is a two-level authentication mechanism. You can only use an ANSYS Thin Server on a certain slave machine, if you know which connection port it is using and those communication ports from which it will accept communications. If the connection port is not correct, then you will not be able to even make a connection to the ANSYS Thin Server. If the communication port number is incorrect, then the ANSYS Thin Server will refuse the connection.

To use the connection port option requires the following on each slave machine:

- An account that you have access to
- The installation path for the ANSYS executable must be in the lookup path
- Configuring the AnsysClients file
- A valid ANSYS license available for the child process
- Start the ANSYS Thin Server
- Stop the ANSYS Thin Server

Account

Your system administrator should create an account specifically for you.

ANSYS in PATH

See Section 3.3.9.3.1.2: Using the Remote Shell Option for information on how to place the ANSYS executable in your path.

AnsysClients

The AnsysClients file is read by the ANSYS Thin Server on startup and must be in the directory in which you are going to run the ANSYS Thin Server. This file must contain a list of each master machine IP address and the communication port that the ANSYS Thin Server will accept communications from. The master machine's IP address must be used, but may be specified with a wildcard to allow connection from a network domain. The communication port is to be a value between 49512 and 65535. The communication port can also be specified as a range, which allows for that many connections. It is best to always specify a range for the communication port numbers. For example:

```
10.3.* 59100-59130
10.3.20.1 59200-59230
10.2.5.55 59300-59330
10.1.1.104 59400-59430
192.1.10.34 59500-59530
…
```

…

Make sure that the port number ranges are not overlapping and are unique for each master machine or network domain. You should make the range at least as wide as the number of possible users connecting from a master machine.

Start the ANSYS Thin Server

Starting on a PC

On a PC open a command prompt window and go to the directory containing the AnsysClients file mentioned above.

Issue the command: ansysts81 connection port

This will start the ANSYS Thin Server using the specific connection port. The value of connection port is the port on which the master machines will connect to the ANSYS Thin Server. The value of "connection_port" should be between 49512 and 65535. For example:

ansysts81 62000

The ANSYS Thin Server will start without any message and will continue running until the command prompt window is closed or using Ctrl-C to stop the process. The command prompt window may be minimized.

Starting on a UNIX Machine

On a UNIX machine go to the directory containing the AnsysClients file mentioned above.

Issue the command: ansysts81 connection_port &

This will start the ANSYS Thin Server using the specific connection port. The value of connection port is the port on which the master machines will connect to the ANSYS Thin Server. The value of "connection_port" should be between 49512 and 65535. For example:

ansysts81 62000&

The ANSYS Thin Server will start without any message and will continue running in the background until the machine is restarted or you kill the process. You may close or minimize the window in which the ANSYS Thin Server was started.

Stop the ANSYS Thin Server

The ANSYS Thin Server should be stopped once the parallel process is complete.

Stopping on a PC

You may stop the process by closing the command prompt window or using Ctrl-C.

Stopping on a UNIX Machine

On a UNIX machine use a "ps -u UserId | grep tclsh" in the command line to find out under which process-id the ANSYS Thin Server is running. Here, UserId is your user account name on the machine. With this "ps" command should get two processes running. Typically, there will be two processes listed, one process for "anstclsh" and another for "tclsh". Use "kill -9 process-id" to kill both processes. Under certain circumstances killing the "anstclsh"-process will also take away the "tclsh"-process (just issue the "ps" command again to verify). If this does not happen, then just kill the "tclsh"-process separately.

3.3.9.3.1.4. Configuring the Master Machine

After you have configured the slave machines you need to configure the master machine for the type of ANSYS Thin Server startup you chose to use on the slave machines. The communication from the master machine to the slave machines is done by the ANSYS Nanny which is a program running in the parent process. It takes care of running the necessary simulations on different slave machines. It will establish the connection to the slave machines, copy the necessary files over to the slave machines, start and monitor the running simulations and clean up the working directories after the entire sequence of simulations is finished. The ANSYS Nanny will be started automatically as you start executing PDS simulations in parallel, distributed mode.

For parallel processing you need to specify the remote hosts you want to use. This information is placed in a file called hosts 81.ans. You can create this file using a text editor or you can use the ANS_ADMIN utility (see the online help available with the ANS_ADMIN utility for more information). This file contains host information for all remote machines on which you may want to run. This file is global and does not contain job-specific information. Typically, your IT administrator would provide this file, including all the information about the slave machines a typical user can use in your network. If you have multiple users running parallel or distributed jobs at your site, you should have one hosts 81.ans file for all users. But you can copy this file to your local working directory and make adjustments. ANSYS searches for this file first in the local directory, followed by the home directory, and finally the apdl directory.

You have two options when setting up the Master Machine to use the ANSYS Thin Server on the slave machines.

Configuration when the Thin Server uses the Remote Shell Option

Let's assume that the slave machine called "MySlaveMachine" has been prepared to work under the remote shell option as outlined above. The slave machine "MySlaveMachine" is an SGI UNIX machine and it has 4 CPUs. On the slave machine we also want to use the directory "/tmp/sdr/pds_runs" as the local directory for the remote simulations to run in. In this case your hosts 81.ans file must include the line:

#						
# HOST	OS	PORT	CPU	TIME	LocPORT	I/O Directory
MySlaveMachine	SGI64	0	4	15	0	/tmp/sdr/pds_runs

Configuration when the Thin Server uses the Port Option

Let's assume you now want to use the same slave machine "MySlaveMachine" using the ANSYS Thin Server. The ANSYS Thin Server has been started on a slave machine called "MySlaveMachine" using the command "ansysts81 62000as illustrated above. Let us also assume that the file "AnsysClients" looks exactly like shown in the section above, i.e. your master machine "MyMasterMachine" can communicate to the ANSYS Thin Server using the communication port numbers 59400-59430. In this case your hosts 81.ans file must include the line:

#						
# HOST	OS	PORT	CPU	TIME	LocPORT	I/O Directory
MySlaveMachine	SGI64	62000	4	15	59400-59430	/tmp/sdr/pds_runs

This will make sure that the ANSYS Thin Server on the slave machine will be contacted using the same connection port it has been started with, i.e. 62000 in this case. Also the communication will use the same communication port numbers the ANSYS Thin Server accepts from the machine "MyMasterMachine" where you try to connect from.

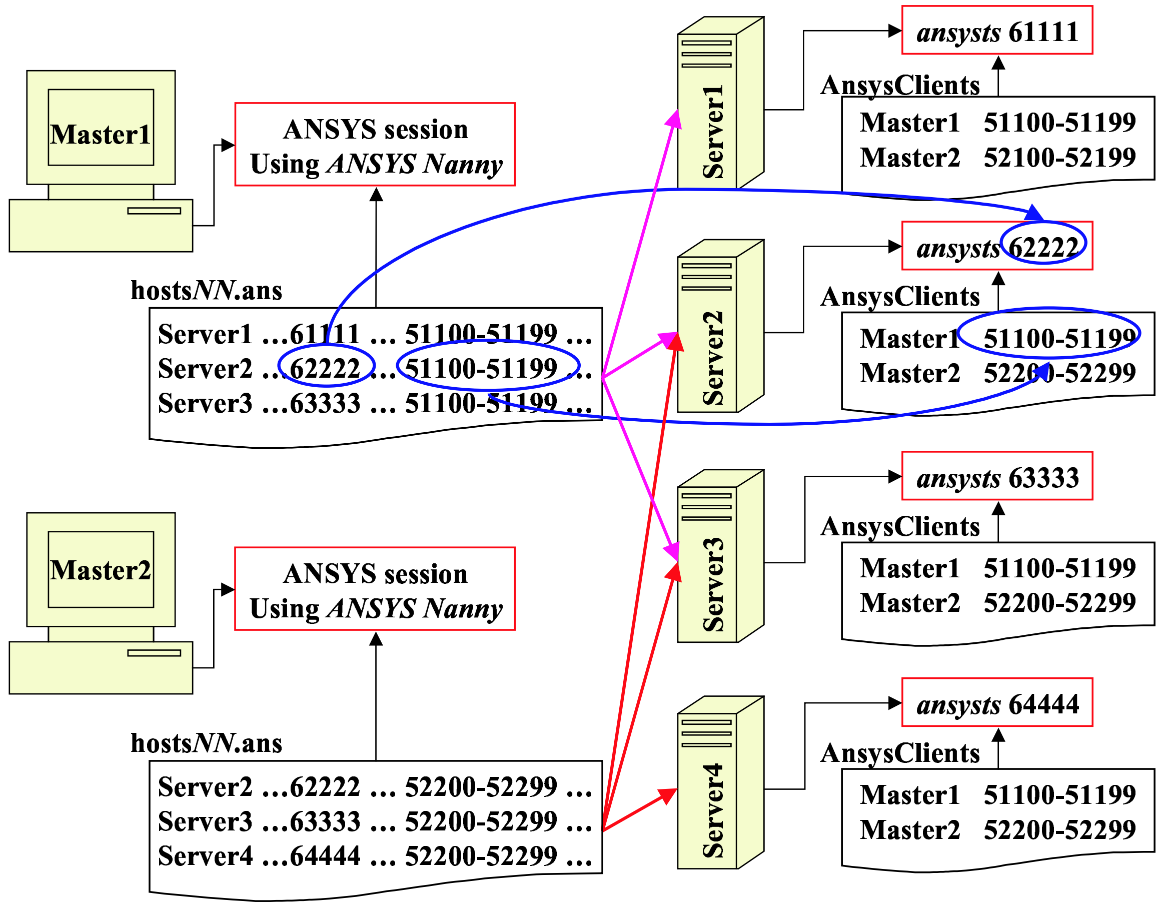
A sample **hosts81.ans** file looks like this:

```
# This file is used to specify those hosts that the ANSYS Nanny may
# run children on.
```

```
#
# Each host entry is to be on its own line.  The host entry consists of
# several fields which are space delimited.
#
#  Field 1 - host IP address or name
#  Field 2 - host machine type
#  Field 3 - execution key (used for Probabilistic Design only):
#        0-Use a remote shell to start the child process;
#          this requires a remote shell server to be
#          running on the host machine.
#    >1024-Use a running ANSYS thin server on the host
#          which is listening on this port number.
#  Field 4 - The default maximum number of jobs to run on this host
#  Field 5 - The time in minutes to check again if the host is available.
#            If this is zero then the host will not be checked again.
#  Field 6 - The local port number to start the communication with the
#            ANSYS Thin Server on. This is tied to authentication on the
#            ANSYS Thin Server.
#  Field 7 - The directory to create the children subdirectories in
#  Field 8 - The cluster type.  Only valid entry is MPI.
#  Field 9 - The speed factor (relative speed to other machines listed).
             Only valid entry is 1.
#  Field 10 - Number of OpenMP threads.  Only valid entry is 1.
# Example:
#
#  UNIX box that has five processors
#  zeus  sgi64    0     5  30    2000     /scratch/wjc
#  Microsoft box using the ANSYS Thin Server
#  wjcpc XP       2010  1  0     2000     C:\TEMP
alpha1   alpha    0      1     15    2000     /scratch/epc   MPI 1 1
athena   sgi64    0      1     15    2000     /scratch/epc   MPI 1 1
rs43p    rs6000   0      1     15    2000     /home/pdstest  MPI 1 1
rs260    rs64     0      1     15    2000     /home/pdstest  MPI 1 1
snoopy   hppa8000 0      1     15    2000     /home/pdstest  MPI 1 1
alpha24  alpha    0      1     15    2000     /home/pdstest  MPI 1 1
hp770    hppa8000 0      1     15    2000     /home/pdstest  MPI 1 1
us60     usparc   0      1     15    2000     /home/pdstest  MPI 1 1
ss60     sun64    0      1     15    2000     /home/pdstest  MPI 1 1
```

3.3.9.3.1.5. Illustration of the host set-up using port option

The picture below illustrates an example of the set-up of a network with 2 master machines and 4 slave machines using the connection port option. Here, the first master machines uses slave machines 1, 2 and 3, while the second master machine is using only slave machines 2, 3 and 4. In this illustration "NN" represents the revision number of ANSYS.



3.3.9.3.1.6. Host and Product selection for a particular analysis

After the **host81.ans** file is specified you need to provide specific information that may change from analysis to analysis.

If you are working in interactive mode then select:

: Main Menu> Prob Design> Run> Exec Parallel> Host Select

In this menu you can:

- Select the slave machines you want to use for a particular analysis. This is necessary, for example, if you know that some machines in your network are in use by another user at the time you want to start the analysis.
- Choose the licenses you want to use for the particular analysis. If there are multiple levels of licenses of ANSYS available in your network, then you should first select the lower license that will be able to run your analysis.

Based on the information provided in this menu a **Jobname.hosts** file is created or updated as you press OK in the menu.

If you are working in batch mode, you must create this file using a text editor. However, for sake of simplicity it is recommended to let ANSYS create this file using interactive mode and then to proceed with batch mode operation. This file must reside in the directory where you are running ANSYS PDS. This file must include the following information:

- Remote hosts to be used for this particular parallel run.

- Number of processes that can run in parallel on each host. If a remote host has more than one CPU, you can use all of the CPUs on the remote host. For performance reasons, we recommend leaving one CPU for system tasks and using only *N*-1 CPUs for parallel processing (if *N* is the number of CPUs on the remote host).
- Directories in which you want the child processes to be executed. It is recommend that you use temporary directories like "/scratch" or "/tmp" on UNIX or "C:\TEMP" on PCs. These remote directories are cleaned up automatically after the parallel processes finish running. Make sure that there is enough disk space in the directory for the files created by the analysis.

A sample **jobname.hosts** file looks like this:

```
# This file is used to specify those hosts that the ANSYS Parent may
# run children processes on.
#
# Each host entry is to be on its own line. The host entry consists of
# several fields which are space delimited.
#
# Field 1 - host IP address or name
# Field 2 - username to use for a remote shell to the host
# Field 3 - execution key:
# 0 - Use a remote shell to start the child process
# >1024 - Use a running ANSYS thin server on the host
# which is communicating on this port number
# Field 4 - the product code sequence to attempt to start ANSYS jobs
# on the host with. This is a colon delimited list.
# Field 5 - the number of jobs to run on this host
# Field 6 - The time in minutes to check again if the host is available.
# If this is zero then the host will not be checked again.
# Field 7 - If field 3 is nonzero then this is the local port number
# or range to start the communication with the ANSYS Thin
# Server on. This is tied to the "authentication" on the
# ANSYS Thin Server.
# If field 3 is zero then this should be set to zero.
# Field 8 - directory to create the children subdirectories in
#
# Example:
#
# UNIX box that has five processors and will first attempt to
# run with ANSYS Mechanical Batch Child and then ANSYS Mechanical
# zeus wjc 0 MEBACH:ANSYS 5 30 2000 /scratch/wjc
# XP box running the thin server on port 2010
# wjcpc wjc 2010 ANSYS 1 0 2000 C:\TEMP
alpha1 epc 0 MEBA:MEBACH 1 15 2000 /scratch/epc
athena epc 0 MEBA:MEBACH 1 15 2000 /scratch/epc
rs43p pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
rs260 pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
snoopy pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
hp160 pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
alpha24 pdstest 0 MEBA:MEBACH 2 15 2000 /home/pdstest
hp770 pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
us60 pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
ss60 pdstest 0 MEBA:MEBACH 1 15 2000 /home/pdstest
```

3.3.9.3.2. Files Needed for Parallel Run

You also need to specify all other files that will be used in the distributed processing run. These files are listed in a separate file called **Jobname.ifiles**. At a minimum, this file must include the analysis file. Additionally, you may need other files, such as data files or ANSYS macros. You *must* include all files that the analysis files calls in order for the PDS run to execute successfully. These files, which are typically found only on your local machine, will then be copied to all remote hosts during the PDS run. The **Jobname.ifiles** must reside in your working directory. If you are working in batch mode, you need to create this file manually. The first file listed must be the analysis file. Other files listed must include the full path.

Command(s): PDANL

GUI: Main Menu> Prob Design> Analysis File> Assign

A sample **jobname.ifiles** file looks like this:

```
# Files to copy to the ANSYS server for job Jobname
# created on Fri Oct 13 10:33:44 EDT 2000.
pdsrun1.inp
/home/staff/epc/ddts/pds/tags
```

3.3.9.3.3. Controlling Server Processes

If you are working interactively, you need to complete one additional step. When you select Run Parallel, you see a dialog box listing all remote hosts specified in the **Jobname.hosts** file, as well as their current status. You can use this dialog box to specify a solution label, set the number of allowed failures (loops leading to an ANSYS error during execution) as well as monitor the processes and remote machines, and stop or start processes on those machines as necessary.

The number of allowed failures is used to automatically stop the parallel execution process if the number of loops leading to an error exceeds the allowed maximum number. In a typical case you don't expect your analysis file to cause an ANSYS error. If in this case there are several loops leading to an error, then this is most likely due to a bug in the APDL programming in the analysis file. Here, the ANSYS PDS automatically terminates the parallel execution if the number of allowed failures (loops with errors) exceeds the allowed maximum.

In certain cases however, you will know that executing the analysis file always leads to an error that can be ignored. An example is a non-linear burst analysis, which terminates with an error as the maximum displacement grows out of proportion. I.e. ultimately all loops will terminate with an error and the result of a burst analysis is the value of the load factor at which this instability happens. In this case you can force the ANSYS PDS to ignore the error and set the error flag in the result file to “0”. To do this set the number of allowed failures in the exactly equal to the total number of loops. Since you are bypassing the error checking mechanism it is strongly recommended to thoroughly review of the probabilistic results in the file **jobname_*Slab*.pdrs**. It is not recommended to do this as a standard procedure, but only in cases where you are sure that the error generated in the analysis loop can be safely ignored.

The status codes you could see include:

- DOWN - The remote host is not available (reason unknown).
- DN-TS - The Thin Server is down.
- DN-DRF - ANSYS was unable to create the necessary directories on that host.
- DN-ANS - ANSYS would not start on that host.
- DN-ATH - Authentication failure (the batch child product was unable to authenticate with the parent product).
- UP - The Thin Server is up.
- RUN - The ANSYS simulation is running on that host.

If you want to check diagnostics for a status code in parallel processing mode, choose the **Diagnostics** button in the **Server Process Control** dialog. The diagnostics system will show you the status of the slave machine(s), and details to help you address any issues that arise. Suggested actions are presented.

Note — In batch mode, ANSYS will always attempt to start remote processing on all machines listed in the **Jobname.hosts** file.

3.3.9.3.4. Initiate Parallel Run

You can now activate the parallel execution:

Command(s): PDEXE
GUI: Main Menu> Prob Design> Run> Exec Parallel> Run Parallel

Several things happen when you initiate a parallel processing run.

1. The networked machines are initialized (ANSYS checks for machine permissions, licenses, and directory permissions), and any necessary directories are created.
2. The relevant files are copied to the networked machines.
3. ANSYS is launched in server mode on each networked machine.

Simulations are sent to each machine as that machine finishes a previous simulation; faster machines will naturally process more simulations. If a slave machine (for whatever reason) does not complete a simulation, that simulation is automatically sent to another machine to be processed.

When the analyses are finished (either completed or stopped manually), then the PDS removes any files or directories it created and stops any processes it started.

3.3.10. Fit and Use Response Surfaces

After you have executed a probabilistic analysis, you can use the results stored in the result files to fit response surfaces.

If the probabilistic analysis was based on the Response Surface Method, this step is mandatory. The random output parameter values generated using the Response Surface Method are meant to be fitted with a response surface; therefore, the Response Surface Method determines the values of the random input variables such that fitting of a response surface is most efficient (that is, so that it uses the fewest sampling points).

If the probabilistic analysis was based on the Monte Carlo Simulation method, this step is optional and you can go directly to the results postprocessing. If you use Monte Carlo Simulation results to derive response surfaces, then the sampling points are not as efficiently located for the fitting process, so you should accommodate by using more sample points.

- You should use at least 20% more Monte Carlo Simulation points than what would be required for a Response Surface Method for the same problem. For a list of the number of sampling points required for a Response Surface Method please see Section 3.5: Probabilistic Design Techniques.
- If you cannot determine how many sampling points a Response Surface Method needs (for example, because there are too many random input parameters), then you should have at least two times more sampling points than the number of coefficients of the response surfaces.

3.3.10.1. About Response Surface Sets

The results generated by the response surface fitting procedure as well as the results generated by applying the approximation equation (Monte Carlo Simulation) are combined in a response surface set. Each response surface set includes the following information:

- A unique name that you provide. This name is used to identify the response surface set during probabilistic postprocessing.
- The name of the solution set you used for the fitting procedure (containing the data points).

- The number and the names of the random output parameters for which a response surface has been evaluated. If you have many random output parameters you might not be interested in fitting a response surface for every one, but only for those that are most important.
- For each random output parameter that was fitted with a response surface, the response surface set includes information about the regression model that was used to fit the response surface (linear, quadratic, or quadratic with cross-terms), as well as the terms and coefficients that were derived as result of the regression analysis.
- The Monte Carlo Simulation samples created using the response surface equations.

There is a one-to-one relationship between solution sets and response surface sets. For each solution set containing sample points you can have only one response surface set containing the response surfaces fitting these sample points. The reverse is also true, that each response surface set can only contain the response surfaces that are based on the sample points of one solution set.

3.3.10.2. Fitting a Response Surface

To fit a response surface you must specify a name for the response surface set where you want the results to be stored, the solution set you want to use, one random output parameter, the regression model, an output transformation technique (if any), and whether to filter terms.

The regression model describes which regression terms are used for the approximation function of the response surface. In the ANSYS PDS, the following regression models are implemented:

- Linear approximation function
- Quadratic approximation function without cross-terms
- Quadratic approximation function including cross-terms

While you can use all terms included in the regression model, the ANSYS PDS also offers an option that automatically filters out insignificant terms. This technique is called the forward-stepwise regression analysis. For example, where the Young's modulus E and the thermal expansion coefficient are random input variables, a full quadratic regression model reads:

$$\sigma_{therm} = c_0 + c_1 \cdot E + c_2 \cdot \alpha + c_3 \cdot E \cdot \alpha$$

A full regression model uses the available sampling points to determine values for all regression coefficients c_0 to c_3. Of course the values for c_0 to c_2 will be zero or very close to zero; taking more coefficients into account than really necessary reduces the degrees of freedom of the algebraic equation to be solved to evaluate the coefficients. This in turn reduces the accuracy of the coefficients that are important for the regression fit. The forward-stepwise regression analysis takes this into account and automatically eliminates terms that are not needed.

The ANSYS PDS offers a variety of transformation functions that can be used to make the random response parameter to be more appropriately described by a quadratic function after the transformation has been applied. These transformation functions can be found in Transformation of Random Output Parameter Values for Regression Fitting in the *ANSYS, Inc. Theory Reference*.

Here, y_i is the value of a random output parameter obtained in the *i*-th sampling loop and y_i^* is the corresponding transformed value. The physical nature of the problem should indicate which transformation to use; for example, lifetime parameters (such as the number of cycles until low cycle fatigue occurs) are usually transformed with a logarithmic transformation. If you do not have this kind of information, then you should start with the Box-Cox transformation. The PDS automatically searches for an optimum value for the Box-Cox parameter λ within the interval (-2,2). As guidelines:

- If λ is close to -1.0 then the data is best transformed by a reciprocal transformation, which is a power transformation with an exponent of -1.0.
- If λ is close to zero then the data is best transformed by a logarithmic transformation.
- If λ is close to 0.5 then use the square root transformation.
- If λ is close to 1.0, then no transformation should be used.
- If λ is *not* close to any of these specific values then the Box-Cox transformation is appropriate.

To fit a response surface:

Command(s): RSFIT
GUI: Main Menu> Prob Design> Response Surf> Fit Resp Surf

3.3.10.3. Plotting a Response Surface

Whether a response surface is a good representation of the sampling point that it is supposed to fit can be best illustrated by plotting the response surface. The ANSYS PDS plots the sampling points as symbols and the response surface as a contour plot so you can visually compare them. However, you can only plot one random output parameter as a function of two random input variables at a time.

To plot a response surface:

Command(s): RSPLOT
GUI: Main Menu> Prob Design> Response Surf> Plt Resp Surf

3.3.10.4. Printing a Response Surface

After using a regression analysis to fit a response surface, ANSYS automatically prints all necessary results in the output window:

- The transformation function that has been used or has been determined automatically (in case of Box-Cox transformation)
- Regression terms
- Regression coefficients
- Goodness-of-fit measures

The goodness-of-fit measures provide a means to verify the quality of the response surface and whether it is a good representation of the underlying data (in other words, the sample points).

You can request a print out of this data at any time.

Command(s): RSPRNT
GUI: Main Menu> Prob Design> Response Surf> Prn Resp Surf

3.3.10.5. Generating Monte Carlo Simulation Samples on the Response Surfaces

After you have generated a response surface set that includes one or more response surfaces for one or more random output parameters then you also need to perform Monte Carlo Simulations using these response surfaces to generate probabilistic results. This is where the PDS generates sampling values for the random input variables in the same way it did for the simulation looping performed using your analysis file. But instead of using the random input variable values in the analysis file and running through the analysis file, it uses the approximation function derived for the response surfaces to calculate approximated response values. The process of calculating an explicitly known equation is much faster than running through the analysis file and performing a finite element analysis, so you can run a large number of simulation loops in a relatively short time. Usually, several thousand simulation loops are performed if you utilize the response surfaces.

After you have generated the Monte Carlo Simulation loops on the response surfaces, you can begin probabilistic postprocessing and review the probabilistic results the same way as you would for Monte Carlo Simulations. However, there is one difference for postprocessing between Monte Carlo results and Monte Carlo results derived from response surface approximations. For Monte Carlo simulation results, the accuracy of the results is determined by the number of simulation loops that are performed. The PDS can visualize the accuracy of Monte Carlo results by means of confidence limits or confidence bounds. For Monte Carlo results derived from response surface approximations, the confidence bounds are suppressed. This is necessary because the accuracy is not determined by the number of simulation loops (as mentioned above, you typically perform a large number of these) but by the goodness-of-fit or the response surface model itself. With increasing numbers of simulation loops the confidence bounds tend to merge with the result curve you are plotting (the width of the confidence band shrinks to zero). This could lead you to conclude that the results are very, very accurate. However, the underlying response surface approximation could have been completely inadequate (for example, using a linear approximation function for a highly nonlinear problem).

Command(s): RSSIMS
GUI: Main Menu> Prob Design> Response Surf> RS Simulation

3.3.11. Review Results Data

After probabilistic design looping is complete, you can review the results sets in a variety of ways using the commands described in this section. These commands can be applied to the results from any probabilistic design method or tool.

- Statistics
  - Sample History
  - Histogram
  - Cumulative Distribution Function
  - Probabilities
  - Inverse Probabilities
- Trends
  - Scatter Plot
  - Sensitivities
  - Correlation Matrix
- Report
  - Print HTML Report

3.3.11.1. Viewing Statistics

To postprocess one particular design variable, choose this option. The statistics options are described below.

Plot Sample History

Use the **PDSHIS** command to request a sample history plot.

Command(s): PDSHIS
GUI: Main Menu> Prob Design> Prob Results> Statistics> Sampl History

You must choose the results set you want to use, the design variable you want to review, the plot type to use, and the confidence level.

Plot Histogram

Use the **PDHIST** command to request a histogram plot of a design variable.

Command(s): PDHIST
GUI: Main Menu> Prob Design> Prob Results> Statistics> Histogram

You must choose the results set you want to use, the design variable you want to review, the number of classes/points to use, and the type of histogram.

CumulativeDF

Use the **PDCDF** command to request a histogram plot of a design variable.

Command(s): PDCDF
GUI: Main Menu> Prob Design> Prob Results> Statistics> CumulativeDF

You must choose the results set you want to use, the design variable you want to review, and the confidence level.

The confidence level is a probability expressing the confidence that the values for the cumulative distribution function are in fact between the confidence bounds. The larger the confidence level, the wider the confidence bounds. Plotting of the confidence bounds only makes sense for the postprocessing of Monte Carlo simulation results. Here, the confidence bounds represent the accuracy of the results and with increasing sample size the width of the confidence bounds gets smaller for the same confidence level. For response surface methods the number of simulations done on the response surface is usually very large. Therefore, the accuracy of the results is determined by the goodness of the response surface fit and not by the confidence level.

Probabilities

Use the **PDPROB** command to request the value of a design variable at an specific point on the cumulative distribution curve.

Command(s): PDPROB
GUI: Main Menu> Prob Design> Prob Results> Statistics> Probabilities

You must choose the results set you want to use, the design variable you want to review, the relation (greater than, less than), the limit value, and the confidence level.

Inverse Probabilities

Use the **PDPINV** command to request the value of a design variable at a specific point on the cumulative distribution curve.

Command(s): PDPINV
GUI: Main Menu> Prob Design> Prob Results> Statistics> Inverse Prob

You must choose the results set you want to use, the design variable you want to review, the relation (greater than, less than), the limit value, and the confidence level.

3.3.11.2. Viewing Trends

To postprocess one particular design variable as it relates to another, choose this option. The trend options are described below.

Scatter Plot

Use the **PDSCAT** command to request a scatter plot showing the correlation between two design variables.

> **Command(s): PDSCAT**
> **GUI: Main Menu> Prob Design> Prob Results> Trends> Scatter Plot**

You must select the results set that you want to use, the design variables that you want to review, the type of trendline curve to use (and if plotted, the polynomial order), and the maximum number of point to include in the scatter plot.

Sensitivities

Use the **PDSENS** command to request the sensitivities of an output parameter to the input variables.

> **Command(s): PDSENS**
> **GUI: Main Menu> Prob Design> Prob Results> Trends> Sensitivities**

You must choose the results set and output parameter you want to use, the type of chart to plot, the type of correlation coefficient, and the sensitivity level.

Correlation Matrix

Use the **PDCMAT** command to calculate the correlation coefficient matrix.

> **Command(s): PDCMAT**
> **GUI: Main Menu> Prob Design> Prob Results> Statistics> Probabilities**

You must choose the results set you want to use, which type of design variables you are looking at, the specific design variable names, the type of correlation, the significance level, and whether you want to see the probabilities with the correlation coefficients.

3.3.11.3. Creating Reports

To visualize and summarize all results of a probabilistic analysis, choose this option. Details on specifying a report are described below.

Report Options

Use the **PDROPT** command to request an HTML report.

> **Command(s): PDROPT**
> **GUI: Main Menu> Prob Design> Prob Results> Report> Report Options**

You must choose which statistics and trends to show in the report and in what form you want to see them. See the **PDROPT** command for details, and see the other probabilistic results options for further details.

Generate Report

Use the **PDWRITE** command to request the sensitivities of an output parameter to the input variables.

> **Command(s): PDWRITE**
> **GUI: Main Menu> Prob Design> Prob Results> Report> Generate Report**

You must enter a name for the report file, your first and last name, and whether links should be generated between your report and the analysis file, each analysis loop, and the response surface output parameter details (if the response surface method was used).

3.4. Guidelines for Selecting Probabilistic Design Variables

This section presents useful guidelines for defining your probabilistic design variables.

3.4.1. Choosing and Defining Random Input Variables

Here are a few tips you can use to determine which random input variable in your finite element model follows which distribution function and what the parameters of the distribution function are.

First, you should know to

- Specify a reasonable range of values for each random input variable.
- Set reasonable limits on the variability for each RV.

Note — The values and hints given below are simply guidelines; none are absolute. Always verify this information with an expert in your organization and adapt it as needed to suit your analysis.

3.4.1.1. Random Input Variables for Monte Carlo Simulations

The number of simulation loops that are required for a Monte Carlo Simulation does not depend on the number of random input variables. The required number of simulation loops only depends on the amount of the scatter of the output parameters and the type of results you expect from the analysis. In a Monte Carlo Simulation, it is a good practice to include all of the random input variables you can think of even if you are not sure about their influence on the random output parameters. Exclude only those random input variables where you are very certain that they have no influence. The probabilistic design system will then automatically tell you which random input variables have turned out to be significant and which one are not. The number of simulations that are necessary in a Monte Carlo analysis to provide that kind of information is usually about 50 to 200. However, the more simulation loops you perform, the more accurate the results will be.

3.4.1.2. Random Input Variables for Response Surface Analyses

The number of simulation loops that are required for a Response Surface analysis depends on the number of random input variables. Therefore, you want to select the most important input variable(s), the ones you know have a significant impact on the random output parameters. If you are unsure which random input variables are important, it is usually a good idea to include all of the random variables you can think of and then perform a Monte Carlo Simulation. After you learn which random input variables are important and therefore should be included in your Response Surface Analysis, you can eliminate those that are unnecessary.

3.4.1.3. Choosing a Distribution for a Random Variable

The type and source of the data you have determines which distribution functions can be used or are best suited to your needs.

3.4.1.3.1. Measured Data

If you have measured data then you first have to know how reliable that data is. Data scatter is not just an inherent physical effect, but also includes inaccuracy in the measurement itself. You must consider that the person taking the measurement might have applied a "tuning" to the data. For example, if the data measured represents a load, the person measuring the load may have rounded the measurement values; this means that the data you receive are not truly the measured values. Depending on the amount of this "tuning," this could provide a deterministic bias in the data that you need to address separately. If possible, you should discuss any bias that might have been built into the data with the person who provided that data to you.

If you are confident about the quality of the data, then how to proceed depends on how much data you have. In a single production field, the amount of data is typically sparse. If you have only few data then it is reasonable to use it only to evaluate a rough figure for the mean value and the standard deviation. In these cases, you could model the random input variable as a Gaussian distribution if the physical effect you model has no lower and upper limit, or use the data and estimate the minimum and maximum limit for a uniform distribution. In a mass production field, you probably have a lot of data, in which case you could use a commercial statistical package that will allow you to actually fit a statistical distribution function that best describes the scatter of the data.

3.4.1.3.2. Mean Values, Standard Deviation, Exceedence Values

The mean value and the standard deviation are most commonly used to describe the scatter of data. Frequently, information about a physical quantity is given in the form that its value is; for example, "100±5.5". Often, but not always, this form means that the value "100" is the mean value and "5.5" is the standard deviation. To specify data in this form implies a Gaussian distribution, but you must verify this (a mean value and standard deviation can be provided for any collection of data regardless of the true distribution type). If you have more information (for example, you know that the data must be lognormal distributed), then the PDS allows you to use the mean value and standard deviation for a definition of a lognormal distribution.

Sometimes the scatter of data is also specified by a mean value and an exceedence confidence limit. The yield strength of a material is sometimes given in this way; for example, a 99% exceedence limit based on a 95% confidence level is provided. This means that derived from the measured data we can be sure by 95% that in 99% of all cases the property values will exceed the specified limit and only in 1% of all cases they will drop below the specified limit. The supplier of this information is using mean value, the standard deviation, and the number of samples of the measured data to derive this kind of information. If the scatter of the data is provided in this way, the best way to pursue this further is to ask for more details from the data supplier. Because the given exceedence limit is based on the measured data and its statistical assessment, the supplier might be able to provide you with the details that were used.

If the data supplier does not give you any further information, then you could consider assuming that the number of measured samples was large. If the given exceedence limit is denoted with $x_{1-\alpha/2}$ and the given mean value is denoted with x_μ then the standard deviation can be derived from the equation:

$$\sigma = \frac{\left| x_{1-\alpha/2} - x_\mu \right|}{C}$$

where the values for the coefficient *C* are:

Exceedence Probability	**C**
99.5%	2.5758
99.0%	2.3263
97.5%	1.9600

Exceedence Probability	C
95.0%	1.6449
90.0%	1.2816

3.4.1.3.3. No Data

In situations where no information is available, there is never just one right answer. Below are hints about which physical quantities are usually described in terms of which distribution functions. This might help you with the particular physical quantity you have in mind. Also below is a list of which distribution functions are usually used for which kind of phenomena. Keep in mind that you might need to choose from multiple options.

Geometric Tolerances

- If you are designing a prototype, you could assume that the actual dimensions of the manufactured parts would be somewhere within the manufacturing tolerances. In this case it is reasonable to use a uniform distribution, where the tolerance bounds provide the lower and upper limits of the distribution function.
- Sometimes the manufacturing process generates a skewed distribution; for example, one half of the tolerance band is more likely to be hit than the other half. This is often the case if missing half of the tolerance band means that rework is necessary, while falling outside the tolerance band on the other side would lead to the part being scrapped. In this case a Beta distribution is more appropriate.
- Often a Gaussian distribution is used. The fact that the normal distribution has no bounds (it spans minus infinity to infinity), is theoretically a severe violation of the fact that geometrical extensions are described by finite positive numbers only. However, in practice this is irrelevant if the standard deviation is very small compared to the value of the geometric extension, as is typically true for geometric tolerances.

Material Data

- Very often the scatter of material data is described by a Gaussian distribution.
- In some cases the material strength of a part is governed by the "weakest-link-theory". The "weakest-link-theory" assumes that the entire part would fail whenever its weakest spot would fail. for material properties where the "weakest-link" assumptions are valid, then the Weibull distribution might be applicable.
- For some cases, it is acceptable to use the scatter information from a similar material type. Let's assume that you know that a material type very similar to the one you are using has a certain material property with a Gaussian distribution and a standard deviation of ±5% around the measured mean value; then let's assume that for the material type you are using, you only know its mean value. In this case, you could consider using a Gaussian distribution with a standard deviation of ±5% around the given mean value.
- For temperature-dependent materials it is prudent to describe the randomness by separating the temperature dependency from the scatter effect. In this case you need the mean values of your material property as a function of temperature in the same way that you need this information to perform a deterministic analysis. If *M(T)* denotes an arbitrary temperature dependent material property then the following approaches are commonly used:

  – Multiplication equation:

$$M(T)_{rand} = C_{rand} \overline{M}(T)$$

  – Additive equation:

$$M(T)_{rand} = \overline{M}(T) + \Delta M_{rand}$$

– Linear equation:

$$M(T)_{rand} = C_{rand}\overline{M}(T) + \Delta M_{rand}$$

Here, $\overline{M}(T)$ denotes the mean value of the material property as a function of temperature. In the "multiplication equation" the mean value function is scaled with a coefficient C_{rand} and this coefficient is a random variable describing the scatter of the material property. In the "additive equation" a random variable ΔM_{rand} is added on top of the mean value function $\overline{M}(T)$. The "linear equation" combines both approaches and here both C_{rand} and ΔM_{rand} are random variables. However, you should take into account that in general for the "linear equation" approach C_{rand} and ΔM_{rand} are, correlated.

Deciding which of these approaches is most suitable to describing the scatter of the temperature dependent material property requires that you have some raw data about this material property. Only by reviewing the raw data and plotting it versus temperature you can tell which approach is the better one.

Load Data

- For loads, you usually only have a nominal or average value. You could ask the person who provided the nominal value the following questions: If we have 1000 components that are operated under real life conditions, what would the lowest load value be that only one of these 1000 components is subjected to and all others have a higher load? What would the most likely load value be, i.e. the value that most of these 1000 components have (or are very close to)? What would the highest load value be that only one of the 1000 components is subjected to and all others have a lower load? To be safe you should ask these questions not only of the person who provided the nominal value, but also to one or more experts who are familiar with how your products are operated under real-life conditions. From all the answers you get, you can then consolidate what the minimum, the most likely, and the maximum value probably is. As verification you can compare this picture with the nominal value that you would use for a deterministic analysis. If the nominal value does not have a conservative bias to it then it should be close to the most likely value. If the nominal value includes a conservative assumption (is biased), then its value is probably close to the maximum value. Finally, you can use a triangular distribution using the minimum, most likely, and maximum values obtained.
- If the load parameter is generated by a computer program then the more accurate procedure is to consider a probabilistic analysis using this computer program as the solver mechanism. Use a probabilistic design technique on that computer program to assess what the scatter of the output parameters are, and apply that data as input to a subsequent analysis. In other words, first run a probabilistic analysis to generate an output range, and then use that output range as input for a subsequent probabilistic analysis.

  Here, you have to distinguish if the program that generates the loads is ANSYS itself or your own in-house program. If you have used ANSYS to generate the loads (for example, FLOTRAN analysis calculating fluid loads on a structure or a thermal analysis calculating the thermal loads of a structure) then we highly recommend that you include these load calculation steps in the analysis file (and therefore in the probabilistic analysis). In this case you also need to model the input parameters of these load calculation steps as random input variables. If you have used your own in-house program to generate the loads, you can still integrate the load calculation program in the analysis file (see the **/SYS** command for details), but you must have an interface between that program and ANSYS that allows the programs to communicate with each other and thus automatically transfer data.

You also have to distinguish if the load values are random fields or single random variables. If the load is different from node to node (element to element) then it is most appropriate to include the program calculating the load in the analysis file. If the load is described by one or very few constant values then you can also consider performing a probabilistic analysis with the program calculating these load values. Again you need to provide an interface to transfer input data to this program and get output data (the loads) back to ANSYS. If there is more than just one single load value generated by the program then you should also check for potential correlations.

3.4.1.4. Distribution Functions

Beta Distribution

The Beta distribution is very useful for random variables that are bounded at both sides. If linear operations are performed on random variables that are all subjected to a uniform distribution then the results can usually be described by a Beta distribution. An example is if you are dealing with tolerances and assemblies, where the components are assembled and the individual tolerances of the components follow a uniform distribution. In this case the overall tolerances of the assembly are a function of adding or subtracting the geometrical extension of the individual components (a linear operation). Hence, the overall tolerances of the assembly can be described by a Beta distribution. Also, as previously mentioned, the Beta distribution can be useful for describing the scatter of individual geometrical extensions of components as well. The uniform distribution is a special case of the Beta distribution.

Exponential Distribution

The exponential distribution is useful in cases where there is a physical reason that the probability density function is strictly decreasing as the random input variable value increases. The distribution is mostly used to describe time-related effects; for example, it describes the time between independent events occurring at a constant rate. It is therefore very popular in the area of systems reliability and lifetime-related systems reliability, and it can be used for the life distribution of non-redundant systems. Typically, it is used if the lifetime is not subjected to wear-out and the failure rate is constant with time. Wear-out is usually a dominant life-limiting factor for mechanical components, which would preclude the use of the exponential distribution for mechanical parts. However in cases where preventive maintenance exchanges parts before wear-out can occur, then the exponential distribution is still useful to describe the distribution of the time until exchanging the part is necessary.

Gamma Distribution

The Gamma distribution is again a more time-related distribution function. For example it describes the distribution of the time required for exactly *k* events to occur under the assumption that the events take place at a constant rate. It is also used to describe the time to failure for a system with standby components.

Gaussian (Normal) Distribution

The Gaussian or normal distribution is a very fundamental and commonly used distribution for statistical matters. It is typically used to describe the scatter of the measurement data of many physical phenomena. Strictly speaking, every random variable follows a normal distribution if it is generated by a linear combination of a very large number of other random effects, regardless which distribution these random effects originally follow. The Gaussian distribution is also valid if the random variable is a linear combination of two or more other effects if those effects also follow a Gaussian distribution.

Lognormal Distribution

The lognormal distribution is a basic and commonly used distribution. It is typically used to describe the scatter of the measurement data of physical phenomena, where the logarithm of the data would follow a normal distribution. The lognormal distribution is very suitable for phenomena that arise from the multiplication of a large number of error effects. It is also correct to use the lognormal distribution for a random variable that is the result of multiplying two or more random effects (if the effects that get multiplied are also lognormally distributed). If is often used for lifetime distributions; for example, the scatter of the strain amplitude of a cyclic loading that a material can endure until low-cycle-fatigue occurs is very often described by a lognormal distribution.

Uniform Distribution

The uniform distribution is a very fundamental distribution for cases where no other information apart from a lower and an upper limit exists. If is very useful to describe geometric tolerances. It can also be used in cases where there is no evidence that any value of the random variable is more likely than any other within a certain interval. In this sense it can be used for cases where "lack of engineering knowledge" plays a role.

Triangular Distribution

The triangular distribution is most helpful to model a random variable when actual data is not available. It is very often used to cast the results of expert-opinion into a mathematical form, and is often used to describe the scatter of load parameters. However, regardless of the physical nature of the random variable you want to model, you can always ask some experts questions like "What is the one-in-a-thousand minimum and maximum case for this random variable? and other similar questions. You should also include an estimate for the random variable value derived from a computer program, as described earlier. This is also described in more detail above for load parameters in Section 3.4.1.3: Choosing a Distribution for a Random Variable.

Truncated Gaussian Distribution

The truncated Gaussian distribution typically appears where the physical phenomenon follows a Gaussian distribution, but the extreme ends are cut off or are eliminated from the sample population by quality control measures. As such, it is useful to describe the material properties or geometric tolerances.

Weibull Distribution

In engineering, the Weibull distribution is most often used for strength or strength-related lifetime parameters, and it is the standard distribution for material strength and lifetime parameters for very brittle materials (for these very brittle material the "weakest-link-theory" is applicable). For more details see Section 3.4.1.3: Choosing a Distribution for a Random Variable.

3.4.2. Choosing Random Output Parameters

Output parameters are usually parameters such as length, thickness, diameter, or model coordinates.

The ANSYS PDS does not restrict you with regard the number of random output parameters, provided that the total number of probabilistic design variables (that is random input variables and random output parameters together) does not exceed 5000.

ANSYS recommends that you include all output parameters that you can think of and that might be useful to you. The additional computing time required to handle more random output parameters is marginal when compared to the time required to solve the problem. It is better to define random output parameters that you might not consider important before you start the analysis. If you forgot to include a random output parameter that later turns out to be important, you must redo the entire analysis.

3.5. Probabilistic Design Techniques

Understanding the algorithm used by a computer program is always helpful; this is particularly true in the case of probabilistic design. This section presents details on the method types and the sampling options associated with each. See the *ANSYS, Inc. Theory Reference* for more information.

3.5.1. Monte Carlo Simulations

The Monte Carlo Simulation method is the most common and traditional method for a probabilistic analysis. This method lets you simulate how virtual components behave the way they are built. One simulation loop represents one manufactured component that is subjected to a particular set of loads and boundary conditions.

For Monte Carlo simulations, you can employ either the Direct Sampling method or the Latin Hypercube Sampling method.

When you manufacture a component, you can measure its geometry and all of its material properties (although typically, the latter is not done because this can destroy the component). In the same sense, if you started operating the component then you could measure the loads it is subjected to. Again, to actually measure the loads is very often impractical. But the bottom line is that once you have a component in your hand and start using it then all the input parameters have very specific values that you could actually measure. With the next component you manufacture you can do the same; if you compared the parameters of that part with the previous part, you would find that they vary slightly. This comparison of one component to the next illustrates the scatter of the input parameters. The Monte Carlo Simulation techniques mimic this process. With this method you “virtually” manufacture and operate components or parts one after the other.

The advantages of the Monte Carlo Simulation method are:

- The method is always applicable regardless of the physical effect modeled in a finite element analysis. It not based on assumptions related to the random output parameters that if satisfied would speed things up and if violated would invalidate the results of the probabilistic analysis. Assuming the deterministic model is correct and a very large number of simulation loops are performed, then Monte Carlo techniques always provide correct probabilistic results. Of course, it is not feasible to run an infinite number of simulation loops; therefore, the only assumption here is that the limited number of simulation loops is statistically representative and sufficient for the probabilistic results that are evaluated. This assumption can be verified using the confidence limits, which the PDS also provides.
- Because of the reason mentioned above, Monte Carlo Simulations are the only probabilistic methods suitable for benchmarking and validation purposes.
- The individual simulation loops are inherently independent; the individual simulation loops do not depend on the results of any other simulation loops. This makes Monte Carlo Simulation techniques ideal candidates for parallel processing.

The Direct Sampling Monte Carlo technique has one drawback: it is not very efficient in terms of required number of simulation loops.

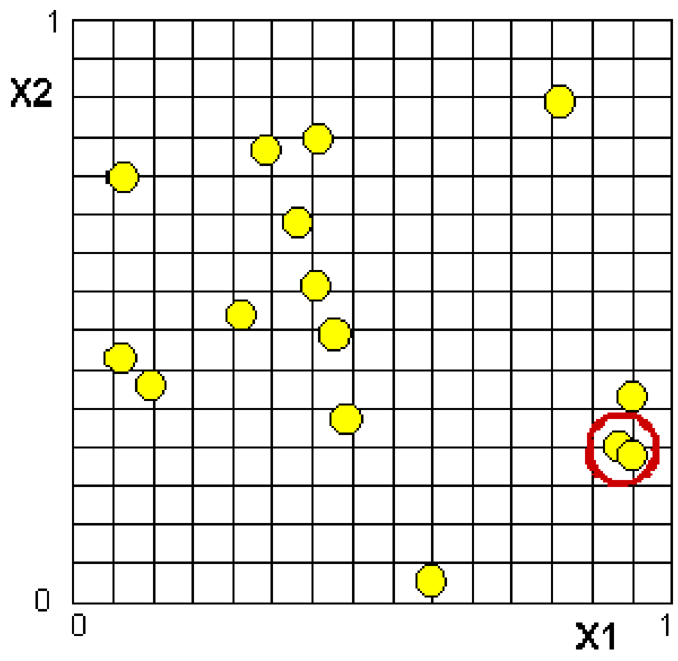
3.5.1.1. Direct Sampling

Direct Monte Carlo Sampling is the most common and traditional form of a Monte Carlo analysis. It is popular because it mimics natural processes that everybody can observe or imagine and is therefore easy to understand. For this method, you simulate how your components behave based on the way they are built. One simulation loop represents one component that is subjected to a particular set of loads and boundary conditions.

The Direct Monte Carlo Sampling technique is not the most efficient technique, but it is still widely used and accepted, especially for benchmarking and validating probabilistic results. However, benchmarking and validating requires many simulation loops, which is not always feasible. This sampling method is also inefficient due to the fact that the sampling process has no "memory."

For example, if we have two random input variables X1 and X2 both having a uniform distribution ranging from 0.0 to 1.0, and we generate 15 samples, we *could* get a cluster of two (or even more) sampling points that occur close to each other if we graphed the two variables (see figure below). While in the space of all random input variables, it can happen that one sample has input values close to another sample, this does not provide new information and insight into the behavior of a component in a computer simulation if the same (or almost the same) samples are repeated.

Figure 3.7 Graph of X1 and X2 Showing Two Samples with Close Values



To use Direct Monte Carlo Sampling, do the following

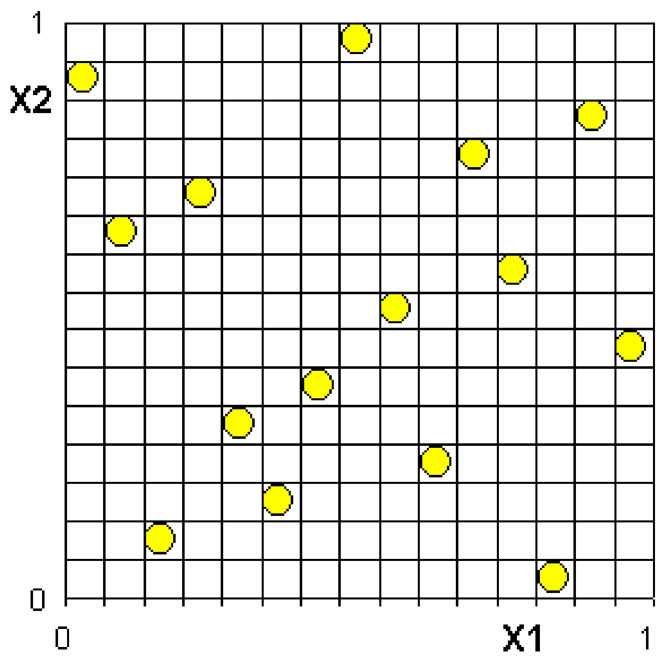
> **Command(s): PDMETH**,MCS,DIR **PDDMCS**
> **GUI: Main Menu> Prob Design> Prob Method> Monte Carlo Sims**

In this sampling method, you set the number of simulations, whether to stop simulations loops when certain criteria are met (accuracy for mean values and standard deviations), and the seed value for randomizing input variable sample data.

3.5.1.2. Latin Hypercube Sampling

The Latin Hypercube Sampling (LHS) technique is a more advanced and efficient form for Monte Carlo Simulation methods. The only difference between LHS and the Direct Monte Carlo Sampling technique is that LHS has a sample "memory," meaning it avoids repeating samples that have been evaluated before (it avoids clustering samples). It also forces the tails of a distribution to participate in the sampling process. Generally, the Latin Hypercube Sampling technique requires 20% to 40% fewer simulations loops than the Direct Monte Carlo Simulation technique to deliver the same results with the same accuracy. However, that number is largely problem dependent.

Figure 3.8 Graph of X1 and X2 Showing Good Sample Distribution



To use the Latin Hypercube Sampling technique:

Command(s): PDMETH,MCS,LHS **PDLHS**
GUI: Main Menu> Prob Design> Prob Method> Monte Carlo Sims

In this sampling method, you set the number of simulations and repetitions, the location in the interval for the sample, whether the simulations stop when certain criteria are met (accuracy of mean values and standard deviations), and random number seed for variability in the sample input variable data.

3.5.1.3. User-Defined Sampling

For this method, you provide the file containing the samples.

Command(s): PDMETH,MCS,USER **PDUSER**
GUI: Main Menu> Prob Design> Prob Method> Monte Carlo Sims

By using this option you have complete control over the sampling data. You are required to give the file name and path.

If user-specified sampling methods are requested with the **PDMETH**,MCS,USER command or the **PDMETH**,RSM,USER command, then you need to specify which file contains the sample data. The sample data is a matrix, where the number of columns is equal to the number of defined random variables and the number of lines is equal to the number of simulation loops requested. This data must be contained in an ASCII file and the content must obey the following notations and format requirements:

- Column separators allowed: blank spaces, commas, semicolons, and tabs.
- Multiple blank spaces and multiple tabs placed directly one after the other are allowed and are considered as one single column separator.

- Multiple commas or semicolons placed directly one after the other are not allowed; for example, two commas with no data between them (just blanks) are read as an empty column, which leads to an error message.
- The first line of the file must contain a solution label. No additional data is allowed on the first line, and if found, will lead to an error message. An error message is also issued if the solution label is missing.
- The solution label is just a placeholder. For consistency, you should use the same solution label you specify in the **PDEXE** command, but if they are different, you will always use the solution label specified in the **PDEXE** command for postprocessing. The PDS system does not check if the solution label in the user-specified file and the one given in the **PDEXE** command match.
- The second line of the file must contain the headers of the data columns. The first three column headers must be “ITER”, “CYCL”, and “LOOP”, respectively; then subsequent columns should contain the names of the random variables. You must use one of the allowed separators as described above between the column headers. No additional data is allowed on this line, and if found, will prompt an error message. An error message is also issued if any of the required column headers are missing.
- The random variable names in the file must match the names of the defined random variables.
- Columns four to *n* can be in arbitrary order. The ANSYS PDS tool determines the order for the random variable data based on the order of the random variable names in the second line.
- The third and subsequent lines must contain the order number for the iteration, cycle, and simulation loop, then the random variable values for that loop. The iteration, cycle, and simulation loop numbers must be in the first, second, and third columns respectively, followed by the random variable values. The iteration and cycle numbers are used by the ANSYS PDS (internally) and for a user-defined sampling method you will typically use a value of "1" for all simulation loops. The loop number is an ascending number from 1 to the total number of loops requested. Additional data is not allowed, and if found, will lead to an error message. An error message is also issued if any of the data columns are missing.
- You must be sure that the order of the random variable values in each line is identical to the order of the random variable names in the second line.
- The user-specified sampling file must contain a minimum of one data line for the random variable values.

When the **PDUSER** command is issued, the PDS checks that the specified file exists, then verifies it for completeness and consistency. Consistency is checked according to the possible minimum and maximum boundaries of the distribution type of the individual random variables. An error message is issued if a random variable value is found in the file that is below the minimum boundary or above the maximum boundary of the distribution of that random variable. This also means that any value will be accepted for a random variable if its distribution has no minimum or maximum boundary; for example, this is the case for the Gaussian (normal) distribution. Apart from this check, it is your responsibility to provide values for the random variables that are consistent with their random distribution.

Note — It is your responsibility to ensure that parameters defined as random input variables are actually input parameters for the analysis defined with the **PDANL** command. Likewise, you must ensure that parameters defined as random output parameter are in fact results generated in the analysis file.

Example

An excerpt of the content of a user-specified sampling file is given below. This example is based on three random variables named X1, X2, and X3. A total of 100 simulation loops are requested.

```
USERSAMP
ITER CYCL     LOOP                X1                 X2                 X3
   1    1        1  1.619379209e+000  2.364528435e-001  1.470789050e+000
   1    1        2  2.237676559e-001  5.788049712e-001  1.821263115e+000
   1    1        3  7.931615474e+000  8.278689033e-001  2.170793522e+000
```

```
  ..   ..      ..              ...               ...               ...
  ..   ..      ..              ...               ...               ...
   1    1      98  1.797221666e+000  3.029471373e-001  1.877701081e+000
   1    1      99  1.290815540e+001  9.271606216e-001  2.091047328e+000
   1    1     100  4.699281922e+000  6.526505821e-001  1.901013985e+000
```

3.5.2. Response Surface Analysis Methods

For Response Surface Analysis, you can choose from three sampling methods: Central Composite Design, Box-Behnken Matrix, and user-defined.

Response Surface Methods are based on the fundamental assumption that the influence of the random input variables on the random output parameters can be approximated by mathematical function. Hence, Response Surface Methods locate the sample points in the space of random input variables such that an appropriate approximation function can be found most efficiently; typically, this is a quadratic polynomial. In this case the approximation function $\hat{Y}$ is described by

$$\hat{Y} = c_0 + \sum_{i=1}^{NRV} c_i X_i + \sum_{i=1}^{NRV} \sum_{j=i}^{NRV} c_{ij} X_i \cdot X_j$$

where c_0 is the coefficient of the constant term, c_i, $i = 1,...NRV$ are the coefficients of the linear terms and c_{ij}, $i = 1,...NRV$ and $j = i, ...,NRV$ are the coefficients of the quadratic terms. To evaluate these coefficients a regression analysis is used and the coefficients are usually evaluated such that the sum of squared differences between the true simulation results and the values of the approximation function is minimized.

Hence, a response surface analysis consists of two steps:

1. Performing the simulation loops to calculate the values of the random output parameters that correspond to the sample points in the space of random input variables.
2. Performing a regression analysis to derive the terms and the coefficients of the approximation function.

The fundamental idea of Response Surface Methods is that once the coefficients of a suitable approximation function are found, then we can directly use the approximation function instead of looping through the finite element model. To perform a finite element analysis might require minutes to hours of computation time; in contrast, evaluating a quadratic function requires only a fraction of a second. Hence, if using the approximation function, we can afford to evaluate the approximated response parameter thousands of times.

A quadratic polynomial is sufficient in many cases of engineering analysis (for example, the evaluation of the thermal stress mentioned above). For that evaluation, the Young's modulus and the thermal expansion coefficient both have a linear effect on the thermal stresses, which is taken into account in a quadratic approximation by the mixed quadratic terms. However, there are cases where a quadratic approximation is not sufficient; for example, if the finite element results are used to calculate the lifetime of a component. For this evaluation, the lifetime typically shows an exponential behavior with respect to the input parameters; thus the lifetime results cannot be directly or sufficiently described by a quadratic polynomial. But often, if you apply a logarithmic transformation to the lifetime results, then these transformed values can be approximated by a quadratic polynomial. The ANSYS PDS offers a variety of transformation functions that you can apply to the response parameters, and the logarithmic transformation function is one of them.

Assuming the approximation function is suitable for your problem, the advantages of the Response Surface Method are:

- It often requires fewer simulation loops than the Monte Carlo Simulation method.

- It can evaluate very low probability levels. This is something the Monte Carlo Simulation method cannot do unless you perform a great number of simulation loops.
- The goodness-of-fit parameters tell you how good the approximation function is (in other words, how accurate the approximation function is that describes your "true" response parameter values). The goodness-of-fit parameters can warn you if the approximation function is insufficient.
- The individual simulation loops are inherently independent (the individual simulation loops do not depend on the results of any other simulation loops). This makes Response Surface Method an ideal candidate for parallel processing.

The disadvantages of the Response Surface Method are:

- The number of required simulation loops depends on the number of random input variables. If you have a very large number of random input variables (hundreds or even thousands), then a probabilistic analysis using Response Surface Methods would be impractical.
- This method is not usually suitable for cases where a random output parameter is a non-smooth function of the random input variables. For example, a non-smooth behavior is given if you observe a sudden jump of the output parameter value even if the values for the random input variables vary only slightly. This typically occurs if you have instability in your model (such as bulking). The same might happen if the model includes a sharp nonlinearity such as a linear-elastic-ideal-plastic material behavior. Or, if you are analyzing a contact problem, where only a slight variation in your random input variables can change the contact situation from contact to non-contact or vice versa, then you also might have problems using the Response Surface Method.

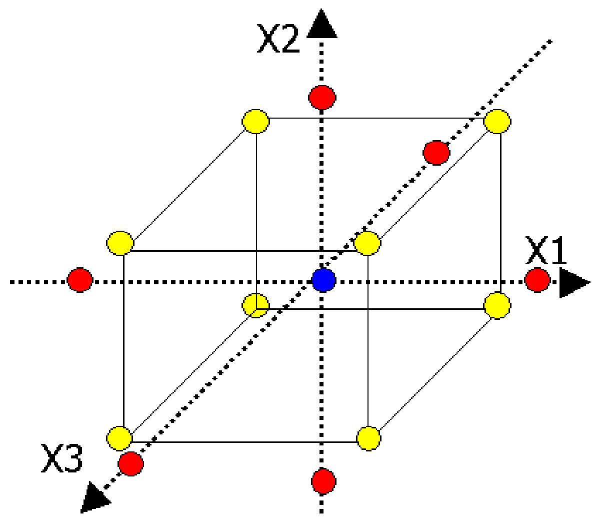
Note — To use Response Surface Methods, the random output parameters must be smooth and continuous functions of the involved random input variables. Do not use Response Surface Methods if this condition is not satisfied.

3.5.2.1. Central Composite Design Sampling

A central composite design consists of a central point, the *N* axis point plus 2^{N-f} factorial points located at the corners of an *N*-dimensional hypercube. Here, *N* is the number of random input variables and *f* is the fraction of the factorial part of the central composite design. A fraction $f = 0$ is a called a full factorial design, $f = 1$ gives a half-factorial design, and so on. The PDS gradually increases the fraction *f* as you increase the number of random input variables. This keeps the number of simulation loops reasonable. The fraction *f* is automatically evaluated such that a resolution V design is always maintained. A resolution V design is a design where none of the second order terms of the approximation function are confined with each other. This ensures a reasonable accuracy for the evaluation of the coefficients of the second order terms.

The locations of the sampling points for a problem with three random input variables is illustrated below.

Figure 3.9 Locations of Sampling Points for Problem with Three Input Variables for CCD



The number of sample points (simulation loops) required for a central composite design as a function of the number of random input variables is given in the table below:

Number of random input variables	Number of coefficients in a quadratic function (with cross-terms)	Factorial number *f*	Number sample points (simulation loops)
1	3	N/A	N/A
2	6	0	9
3	10	0	15
4	15	0	25
5	21	1	27
6	28	1	45
7	36	1	79
8	45	2	81
9	55	2	147
10	66	3	149
11	78	4	151
12	91	4	281
13	105	5	283
14	120	6	285
15	136	7	287
16	153	8	289
17	171	9	291
18	190	9	549
19	210	10	551
20	231	11	553

To use the Response Surface Method with a Central Composite Design, do the following:

Command(s): PDMETH,RSM,CCD **PDDOEL**,*Name*,CCD,...

GUI: Main Menu> Prob Design> Prob Method> Response Surface

PDDOEL allows you to specify design of experiment options.

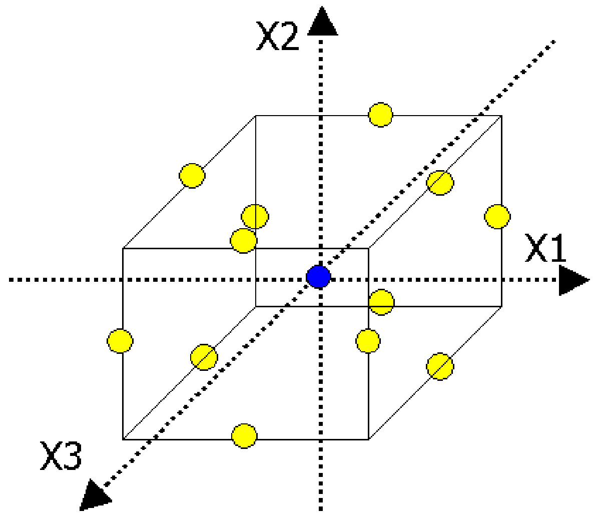
See the *ANSYS, Inc. Theory Reference* for more details.

3.5.2.2. Box-Behnken Matrix Sampling

A Box-Behnken Design consists of a central point plus the midpoints of each edge of an *N*-dimensional hypercube.

The location of the sampling points for a problem with three random input variables is illustrated below.

Figure 3.10 Location of Sampling Points for Problem with Three Input Variables for BBM



The number of sample points (simulation loops) required for a Box-Behnken design as a function of the number of random input variables is given in the table below:

Number of random input variables	Number of coefficients in a quadratic function (with cross-terms)	Number sample points (simulation loops)
1		N/A
2	6	N/A
3	10	12
4	15	25
5	21	41
6	28	49
7	36	57
8	45	65
9	55	121
10	66	161
11	78	177
12	91	193

To use Response Surface Analysis with the Box-Behnken design option, do the following:

Command(s): PDMETH,RSM,BBM **PDDOEL**,*Name*,BBM,...

GUI: Main Menu> Prob Design> Prob Method> Response Surface

PDDOEL allows you to specify design of experiment options.

See the *ANSYS, Inc. Theory Reference* for more details.

3.5.2.3. User-Defined Sampling

For this method, you provide the file containing the samples.

Command(s): PDMETH,RSM,USER **PDUSER**
GUI: Main Menu> Prob Design> Prob Method> Response Surface

By using this option, you have complete control over the sampling data. You are required to give the file name and path.

3.6. Postprocessing Probabilistic Analysis Results

There are two groups of postprocessing functions in the probabilistic design system: statistical and trend.

A statistical analysis is an evaluation function performed on a single probabilistic design variable; for example, a histogram plot of a random output parameter. The ANSYS PDS allows statistical evaluation of either random output parameters or random input variables.

A trend analysis typically involves two or more probabilistic design variables; for example, a scatter plot of one probabilistic design variable versus another.

Probabilistic postprocessing is generally based on result sets. A result set is either a solution set generated by a probabilistic analysis run (**PDEXE**) or a response surface set (**RSFIT**). With a few exceptions, you can perform the same type of probabilistic postprocessing operations regardless of whether you postprocess the results of a solution set or the results of a response surface set.

3.6.1. Statistical Post-Processing

Statistical postprocessing allows you several options for reviewing your results.

3.6.1.1. Sample History

The most fundamental form of postprocessing is directly reviewing the simulation loop results as a function for the number of simulation loops. Here, you can review the simulation values (for either response surface or Monte Carlo methods), or for Monte Carlo Simulations only, the mean, minimum, or maximum values, or the standard deviations.

It is most helpful to review the mean values and standard deviation history for Monte Carlo Simulation results if you want to decide if the number of simulation loops was sufficient. If the number of simulation loops was sufficient, the mean values and standard deviations for all random output parameters should have converged. Convergence is achieved if the curve shown in the respective plots approaches a plateau. If the curve shown in the diagram still has a significant and visible trend with increasing number of simulation loops then you should perform more simulation loops. In addition, the confidence bounds plotted for the requested history curves can be interpreted as the accuracy of the requested curve. With more simulation loops, the width of the confidence bounds is reduced.

Typically, postprocessing Monte Carlo results based on response surfaces is based on a very large number of simulation loops using the response surface equations. Therefore, the accuracy of the results is not a function of the number of simulation loops, but rather the goodness-of-fit measures of the individual response surfaces. As one example, suppose the goodness-of-fit measures indicate a very poor quality of a response surface fit, but the mean value and standard deviation history indicate that the results have converged (because of the large number of loops) and the accuracy is excellent (again because confidence bounds shrink with increasing number of loops). This could lead you to an incorrect conclusion. This is why the PDS only allows you to visualize the sample value history directly, but not the mean value history and so on.

To review the simulation loop results:

Command(s): PDSHIS
GUI: Main Menu> Prob Design> Prob Results> Statistics> Sampl History

You need to select the results set label and the design variable, choose a plot type, and set the confidence level.

3.6.1.2. Histogram

A histogram plot is most commonly used to visualize the scatter of a probabilistic design variable. A histogram is derived by dividing the range between the minimum value and the maximum value into intervals of equal size. Then the PDS determines how many samples fall within each interval, that is, how many "hits" landed in the intervals.

Most likely, you will use histograms to visualize the scatter of your random output parameters. The ANSYS PDS also allows you to plot histograms of your random input variables so you can double check that the sampling process generated the samples according to the distribution function you specified. For random input variables, the PDS not only plots the histogram bars, but also a curve for values derived from the distribution function you specified. Visualizing histograms of the random input variables is another way to make sure that enough simulation loops have been performed. If the number of simulation loops is sufficient, the histogram bars will:

- Be close to the curve that is derived from the distribution function
- Be "smooth" (without large “steps”)
- Not have major gaps

A major gap is given if you have no hits in an interval where neighboring intervals have many hits. However, if the probability density function is flattening out at the far ends of a distribution (for example, the exponential distribution flattens out for large values of the random input variable) then there might logically be gaps. Hits are counted only as positive integer numbers and as these numbers gradually get smaller, a zero hit can happen in an interval.

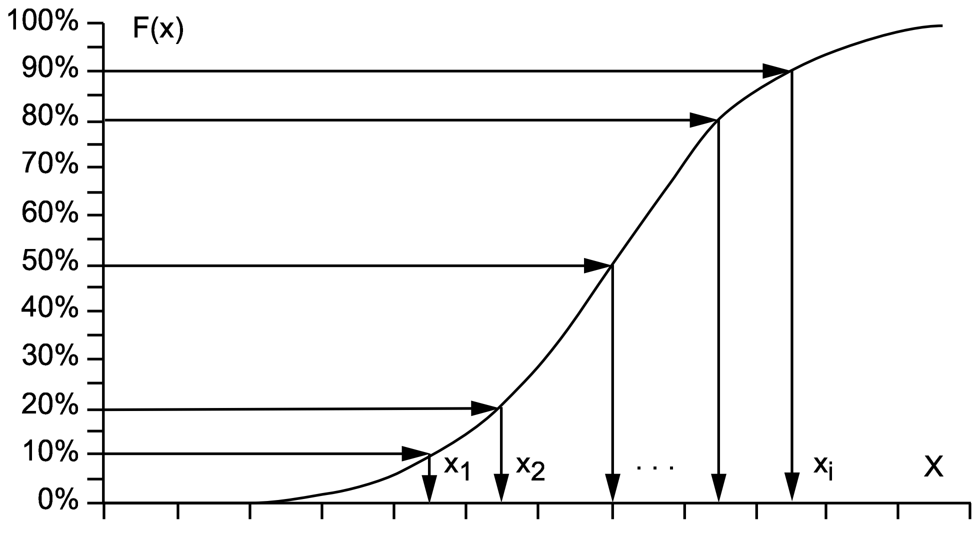
To plot histograms:

Command(s): PDHIST
GUI: Main Menu> Prob Design> Prob Results> Statistics> Histogram

3.6.1.3. Cumulative Distribution Function

The cumulative distribution function is a primary review tool if you want to assess the reliability or the failure probability of your component or product. Reliability is defined as the probability that no failure occurs. Hence, in a mathematical sense reliability and failure probability are two sides of the same coin and numerically they complement each other (are additive to 1.0). The cumulative distribution function value at any given point expresses the probability that the respective parameter value will remain below that point. The figure below shows the cumulative distribution function of the random property *X*:

Figure 3.11 Cumulative Distribution Function of *X*



The value of the cumulative distribution function at the location x_0 is the probability that the values of *X* stay below x_0. Whether this probability represents the failure probability or the reliability of your component depends on how you define failure; for example, if you design a component such that a certain deflection should not exceed a certain admissible limit then a failure event occurs if the critical deflection exceeds this limit. Thus for this example, the cumulative distribution function is interpreted as the reliability curve of the component. On the other hand, if you design a component such that the eigenfrequencies are beyond a certain admissible limit then a failure event occurs if an eigenfrequency drops below this limit. Thus for this example, the cumulative distribution function is interpreted as the failure probability curve of the component.

The cumulative distribution function also lets you visualize what the reliability or failure probability would be if you chose to change the admissible limits of your design.

Often you are interested in visualizing low probabilities and you want to assess the more extreme ends of the distribution curve. In this case plotting the cumulative distribution function in one of the following ways is more appropriate:

- As a Gauss plot (also called a "normal plot"). If the probabilistic design variable follows a Gaussian distribution then the cumulative distribution function is displayed as a straight line in this type of plot.
- As a lognormal plot. If the probabilistic design variable follows a lognormal distribution then the cumulative distribution function is displayed as a straight line in this type of plot
- As a Weibull plot. If the probabilistic design variable follows a Weibull distribution then the cumulative distribution function is displayed as a straight line in this type of plot.

The advantage of these plots is that the probability axis is scaled in a nonlinear fashion such that the extreme ends of the distribution function are emphasized and more visible.

To plot the cumulative distribution function:

Command(s): PDCDF
GUI: Main Menu> Prob Design> Statistics> Prob Results> CumulativeDF

3.6.1.4. Print Probabilities

The PDS offers a function where you can determine the cumulative distribution function at any point along the axis of the probabilistic design variable, including an interpolation function so you can evaluate the probabilities between sampling points. This feature is most helpful if you want to evaluate the failure probability or reliability of your component for a very specific and given limit value.

To print probabilities:

> **Command(s): PDPROB**
> **GUI: Main Menu> Prob Design> Prob Results> Statistics> Probabilities**

3.6.1.5. Print Inverse Probabilities

The PDS offers a function where you can probe the cumulative distribution function by specifying a certain probability level; the PDS tells you at which value of the probabilistic design variable this probability will occur. This is helpful if you want to evaluate what limit you should specify to *not* exceed a certain failure probability, or to specifically achieve a certain reliability for your component.

To print inverse probabilities:

> **Command(s): PDPINV**
> **GUI: Main Menu> Prob Design> Prob Results> Statistics> Inverse Prob**

3.6.2. Trend Postprocessing

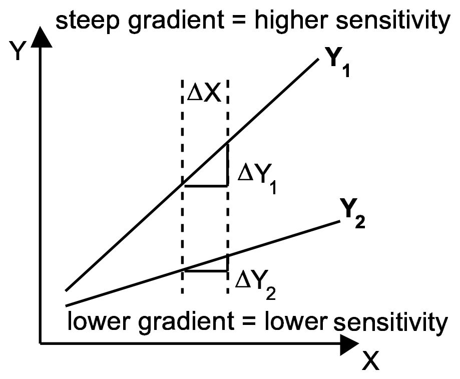
Trend postprocessing allows you several options for reviewing your results.

3.6.2.1. Sensitivities

Probabilistic sensitivities are important in allowing you to improve your design toward a more reliable and better quality product, or to save money while maintaining the reliability or quality of your product. You can request a sensitivity plot for any random output parameter in your model.

There is a difference between probabilistic sensitivities and deterministic sensitivities. Deterministic sensitivities are mostly only local gradient information. For example, to evaluate deterministic sensitivities you can vary each input parameters by ±10% (one at a time) while keeping all other input parameters constant, then see how the output parameters react to these variations. As illustrated in the figure below, an output parameter would be considered very sensitive with respect to a certain input parameter if you observe a large change of the output parameter value.

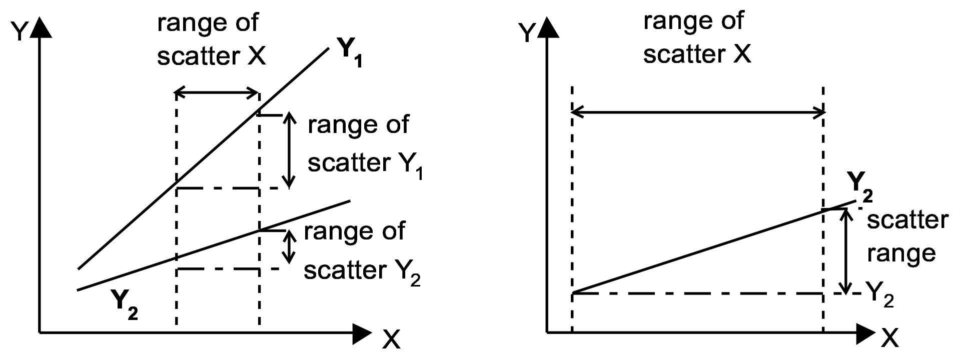
Figure 3.12 Sensitivities


steep gradient = higher sensitivity
Y
Y_1
ΔX
ΔY_1
Y_2
ΔY_2
lower gradient = lower sensitivity
X

These purely deterministic considerations have various disadvantages that are taken into consideration for probabilistic sensitivities, namely:

- A deterministic variation of an input parameter that is used to determine the gradient usually does not take the physical range of variability into account. An input parameter varied by ±10% is not meaningful for the analysis if ±10% is too much or too little compared with the *actual* range of physical variability and randomness. In a probabilistic approach the physical range of variability is inherently considered because of the distribution functions for input parameters. Probabilistic sensitivities measure how much the range of scatter of an output parameter is influenced by the scatter of the random input variables. Hence, two effects have an influence on probabilistic sensitivities: the slope of the gradient, plus the width of the scatter range of the random input variables. This is illustrated in the figures below. If a random input variable has a certain given range of scatter, then the scatter of the corresponding random output parameter is larger, and the larger the slope of the output parameter curve is (first illustration). But remember that an output parameter with a moderate slope can have a significant scatter if the random input variables have a wider range of scatter (second illustration).

Figure 3.13 Range of Scatter



- Gradient information is local information only. It does not take into account that the output parameter may react more or less with respect to variation of input parameters at other locations in the input parameter space. However, the probabilistic approach not only takes the slope at a particular location into account, but also all the values the random output parameter can have within the space of the random input variables.
- Deterministic sensitivities are typically evaluated using a finite-differencing scheme (varying one parameter at a time while keeping all others fixed). This neglects the effect of interactions between input parameters. An interaction between input parameters exists if the variation of a certain parameter has a greater or lesser effect if at the same time one or more other input parameters change their values as well. In some cases interactions play an important or even dominant role. This is the case if an input parameter is not significant on its own but only in connection with at least one other input parameter. Generally, interactions play an important role in 10% to 15% of typical engineering analysis cases (this figure is problem dependent). If interactions are important, then a deterministic sensitivity analysis can give you completely incorrect results. However, in a probabilistic approach, the results are always based on Monte Carlo simulations, either directly performed using you analysis model or using response surface equations. Inherently, Monte Carlo simulations always vary all random input variables at the same time; thus if interactions exist then they will always be correctly reflected in the probabilistic sensitivities.

To display sensitivities, the PDS first groups the random input variables into two groups: those having a significant influence on a particular random output parameter and those that are rather insignificant, based on a statistical significance test. This tests the hypothesis that the sensitivity of a particular random input variable is identical to zero and then calculates the probability that this hypothesis is true. If the probability exceeds a certain significance level (determining that the hypothesis is likely to be true), then the sensitivity of that random input variable is negligible. The PDS will plot only the sensitivities of the random input variables that are found to be significant. However, insignificant sensitivities *are* printed in the output window. You can also review the significance probabilities used by the hypothesis test to decide which group a particular random input variable belonged to.

The PDS allows you to visualize sensitivities either as a bar chart, a pie chart, or both. Sensitivities are ranked so the random input variable having the highest sensitivity appears first.

In a bar chart the most important random input variable (with the highest sensitivity) appears in the leftmost position and the others follow to the right in the order of their importance. A bar chart describes the sensitivities in an absolute fashion (taking the signs into account); a positive sensitivity indicates that increasing the value of the random input variable increases the value of the random output parameter for which the sensitivities are plotted. Likewise, a negative sensitivity indicates that increasing the random input variable value reduces the random output parameter value. In a pie chart, sensitivities are relative to each other.

In a pie chart the most important random input variable (with the highest sensitivity) will appear first after the 12 o'clock position, and the others follow in clockwise direction in the order of their importance.

Using a sensitivity plot, you can answer the following important questions.

How can I make the component more reliable or improve its quality?

If the results for the reliability or failure probability of the component do not reach the expected levels, or if the scatter of an output parameter is too wide and therefore not robust enough for a quality product, then you should make changes to the important input variables first. Modifying an input variable that is insignificant would be waste of time.

Of course you are not in control of all random input parameters. A typical example where you have very limited means of control are material properties. For example, if it turns out that the environmental temperature (outdoor) is the most important input parameter then there is probably nothing you can do. However, even if you find out that the reliability or quality of your product is driven by parameters that you cannot control, this has importance — it is likely that you have a fundamental flaw in your product design! You should watch for influential parameters like these.

If the input variable you want to tackle is a geometry-related parameter or a geometric tolerance, then improving the reliability and quality of your product means that it might be necessary to change to a more accurate manufacturing process or use a more accurate manufacturing machine. If it is a material property, then there is might be nothing you can do about it. However, if you only had a few measurements for a material property and consequently used only a rough guess about its scatter and the material property turns out to be an important driver of product reliability and quality, then it makes sense to collect more raw data. In this way the results of a probabilistic analysis can help you spend your money where it makes the most sense — in areas that affect the reliability and quality of your products the most.

How can I save money without sacrificing the reliability or the quality of the product?

If the results for the reliability or failure probability of the component are acceptable or if the scatter of an output parameter is small and therefore robust enough for a quality product then there is usually the question of how to save money without reducing the reliability or quality. In this case, you should first make changes to the input variables that turned out to be *insignificant*, because they do not effect the reliability or quality of your product. If it is the geometrical properties or tolerances that are insignificant, you can consider applying a less expensive manufacturing process. If a material property turns out to be insignificant, then this is not typically a good way to save money, because you are usually not in control of individual material properties. However, the loads or boundary conditions can be a potential for saving money, but in which sense this can be exploited is highly problem dependent.

Command(s): PDSENS
GUI: Main Menu> Prob Design> Prob Results> Trends> Sensitivities

3.6.2.2. Scatter Plots

While the sensitivities point indicate which probabilistic design parameters you need to modify to have an impact on the reliability or failure probability, scatter plots give you a better understanding of *how* and *how far* you should modify the input variables. Improving the reliability and quality of a product typically means that the scatter of the relevant random output parameters must be reduced.

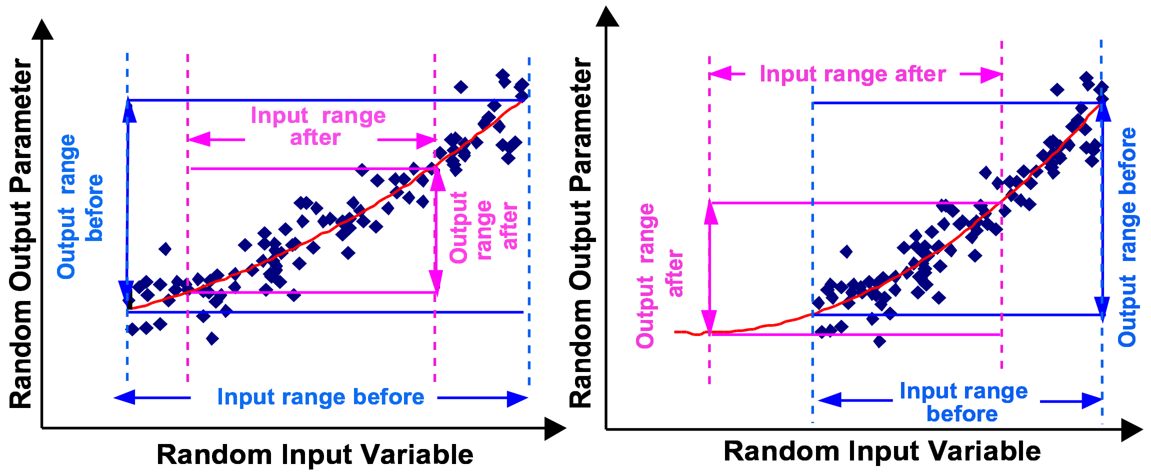
The PDS allows you to request a scatter plot of any probabilistic design variable versus any other one, so you can visualize the relationship between two design variables (input variables or output parameters). This allows you to verify that the sample points really show the pattern of correlation that you specified (if you did so). Typically, random output parameters are correlated as because they are generated by the same set of random input variables. To support the process of improving the reliability or quality of your product, a scatter plot showing a random output parameter as a function of the most important random input variable can be very helpful.

When you display a scatter plot, the PDS plots the sampling points and a trendline. For this trendline, the PDS uses a polynomial function and lets you chose the order of the polynomial function. If you plot a random output parameter as a function of a random input variable, then this trendline expresses how much of the scatter on the random output parameter (Y-axis) is controlled by the random input variable (X-axis). The deviations of the sample points from the trendline are caused and controlled by all the other random input variables. If you want to reduce the scatter of the random output parameter to improve reliability and quality, you have two options:

- Reduce the width of the scatter of the most important random input variable(s) (that you have control over).
- Shift the range of the scatter of the most important random input variable(s) (that you have control over).

The effect of reducing and shifting the scatter of a random input variable is illustrated in the figures below. "Input range before" denotes the scatter range of the random input variable before the reduction or shifting, and "input range after" illustrates how the scatter range of the random input variable has been modified. In both cases, the trendline tells how much the scatter of the output parameter is affected and in which way if the range of scatter of the random input variable is modified.

Figure 3.14 Effects of Reducing and Shifting Range of Scatter



It depends on your particular problem if either reducing or shifting the range of scatter of a random input variable is preferable. In general, reducing the range of scatter of a random input variable leads to higher costs. A reduction of the scatter range requires a more accurate process in manufacturing or operating the product — the more accurate, the more expensive it is. This might lead you to conclude that shifting the scatter range is a better idea, because it preserves the width of the scatter (which means you can still use the manufacturing or operation process that you have). Below are some considerations if you want to do that:

- Shifting the scatter range of a random input variable can only lead to a reduction of the scatter of a random output parameter if the trendline shows a clear nonlinearity. If the trendline indicates a linear trend (if it is a straight line), then shifting the range of the input variables anywhere along this straight line doesn't make any difference. For this, reducing the scatter range of the random input variable remains your only option.
- It is obvious from the second illustration above that shifting the range of scatter of the random input variable involves an extrapolation beyond the range where you have data. Extrapolation is always difficult and even dangerous if done without care. The more sampling points the trendline is based on the better you can extrapolate. Generally, you should not go more than 30-40% outside of the range of your data. But the advantage of focusing on the important random input variables is that a slight and careful modification can make a difference.

Note — ANSYS strongly recommends that you redo the entire probabilistic analysis using the new and modified random input variables if you have reduced of shifted the scatter range of any input variables using the procedure and recommendations above. To redo the probabilistic analysis, save the probabilistic model using the **PDSAVE** command and clear the current probabilistic analysis results using the **PDCLR**,POST command. Then you can modify the random input variable definitions and redo the probabilistic analysis.

Command(s): PDSCAT
GUI: Main Menu> Prob Design> Prob Results> Trends> Scatter

3.6.2.3. Correlation Matrix

Probabilistic sensitivities are based on a statistical correlation analysis between the individual probabilistic design variables. The PDS lets you review the correlation data that has been used to derive sensitivities and decide if individual sensitivity values are significant or not. This information is collected in the correlation matrix of the random output parameters versus the random input variables. The PDS also lets you review the correlations that have been sampled between random input variables, which is stored in the random input variables correlation matrix. The correlations between random output parameters are important if you want to use the probabilistic results of your probabilistic analysis as input for another probabilistic analysis.

To print out a correlation matrix:

> **Command(s): PDCMAT**
> **GUI: Main Menu> Prob Design> Prob Results> Trends> Correl Matrix**

3.6.3. Generating an HTML Report

The ANSYS probabilistic design system automatically generates an HTML report for all probabilistic analyses that you performed with your probabilistic model. The report explains the problem you analyzed, which probabilistic methods were used, and the results. The report uses pictures as well as verbal description and explanations to document all this. The report is a predefined HTML document that you can modify and add to using an HTML editor. However, you can also influence the contents of the report to some extent.

To generate an HTML report for probabilistic design:

> **Command(s): PDROPT, PDWRITE**
> **GUI: Main Menu> Prob Design> Prob Results> Report> Report Options**
> **Main Menu> Prob Design> Prob Results> Report> Generate Report**

Use the **PDROPT** command to choose the items you want to include in your report, then use the **PDWRITE** command to generate it.

3.7. Multiple Probabilistic Design Executions

There are various reasons why you may wish to perform more than one probabilistic design execution. For example, your initial probabilistic design run may not contain enough samples or loops to find the desired probability limits. Or, you may start by using one probabilistic design analysis method, then try another method to compare or validate results. The knowledge you gain from the first few loops may prompt you to change the method you use or other analysis settings.

If you run a probabilistic analysis using a new probabilistic methods make sure that you also provide a new solution label in the **PDEXE** command. If you want to add data to an existing solution set then make sure that you use the same probabilistic method as the one that has been used in the existing solution set. In this case please also note, that if the existing solution set has already a response surface set associated to it, then all response surface sets will be deleted and need to redifined using the **RSFIT** command.

If you perform all executions within the same ANSYS session (or within the same batch input stream), the procedure is very straightforward. After an execution, redefine the probabilistic design input as desired and initiate the next execution. To initiate the execution:

> **Command(s): PDEXE**
> **GUI: Main Menu> Prob Design> Run> Exec Serial> Run Serial**
> **Main Menu> Prob Design> Run> Exec Parallel> Run Parallel**

For more information on these execution options, see Section 3.3.9: Execute Probabilistic Analysis Simulation Loops. If you left the ANSYS program after performing an analysis and would like to continue your probabilistic design analysis at some later time, you can do a save and restart as described next.

3.7.1. Saving the Probabilistic Design Database

Because the probabilistic database is completely separate from the ANSYS database, if you save the ANSYS database using the command **SAVE** it does not save the probabilistic design database. An exception is when you leave the ANSYS session and choose the "Save Everything" button in the "EXIT from ANSYS" menu, in which case the PDS database is also saved. If you already saved the probabilistic database to a specific file, then the "Save Everything" command will cause the current PDS database to overwrite the data in that file; otherwise, the data is written to **jobname.pds** in your current working directory.

The probabilistic design data is automatically saved at the end of each probabilistic design analysis loop. You can save the probabilistic design data at any time by using the **PDSAVE** command.

To save a probabilistic design analysis:
Command(s): PDSAVE
GUI: Main Menu> Prob Design> Prob Database> Save

Caution: Currently, you *must* use the Save and Resume commands on the Prob Design menu to save your work if interrupted or if you simply want to resume the project later.

3.7.2. Restarting a Probabilistic Design Analysis

The probabilistic database is maintained completely separate from the ANSYS database. This means that if you resume the ANSYS database then the probabilistic design database is *not* also resumed.

To restart a probabilistic design analysis, resume the probabilistic design database file (**Jobname.PDS**):
Command(s): PDRESU
GUI: Main Menu> Prob Design> Prob Database> Resume

Once the data is read in, you can respecify probabilistic design type, controls, etc., and initiate looping. (The analysis file corresponding to the resumed database must be available in order to perform probabilistic design.) Note that the previous results are not loaded into the database until a probabilistic postprocessing command is issued.

See the **/PDS**, **PDRESU**, **PDSAVE**, and **PDEXE** command descriptions for more details.

Note — In addition to probabilistic design data, the ANSYS jobname is saved to the probabilistic design database file (**Jobname.PDS**). Therefore, when a probabilistic design data file is resumed (**PDRESU**), the jobname saved in that file will overwrite the current jobname (**/FILNAME**).

You can use the **PDRESU** command (**Main Menu> Prob Design> Prob Database> Resume**) in an interactive session to resume probabilistic design data that was created through a batch run, thus allowing convenient interactive viewing of batch probabilistic design results.

3.7.3. Clearing the Probabilistic Design Database

The probabilistic database is completely separate from the ANSYS database. This usually means that saving and resuming the ANSYS database has no effect on saving and resuming the PDS database. However, if you issue the **/CLEAR** command to clear the ANSYS database, then the PDS database is also cleared. For this reason you should never use the **/CLEAR** command as part of your PDS analysis file. Clearing the PDS database has no effect on the ANSYS database.

For clearing the probabilistic database you have two options. You can clear the entire PDS database or just the results and solution portion. The latter maintains the probabilistic model definition (the random input variable definitions, correlations, and random output parameter definitions). Clearing only the solution and results part of the PDS database is helpful if you want to modify the probabilistic model (change RVs, set correlations between RVs or RPs, etc.). After you have performed a probabilistic analysis, the PDS will not allow you to change any of the probabilistic model details. This ensures consistency of the probabilistic model with the results.

To clear the probabilistic design database:

> **Command(s): PDCLR**
> **GUI: Main Menu> Prob Design> Prob Database> Clear & Reset**

Because the ANSYS database is not affected by the **PDCLR** command, it may also be necessary to clear the ANSYS database if the resumed probabilistic design problem is totally independent of the previous one. See the **/CLEAR** command for details.

3.8. Sample Probabilistic Design Analysis

In the following example, you will perform a probabilistic design analysis of a simple indeterminate three-bar truss. This analysis uses the Direct Monte Carlo method.

3.8.1. Problem Description

You will analyze the effect of variation in the 3-bar truss cross-section that is subjected to a force with both vertical and horizontal components.

3.8.2. Problem Specifications

The loading for this example is a force of 20000 lbs. located at the vertex of the three truss members. The force acts at a 45° angle from the vertical direction.

The following material property is used for this analysis:

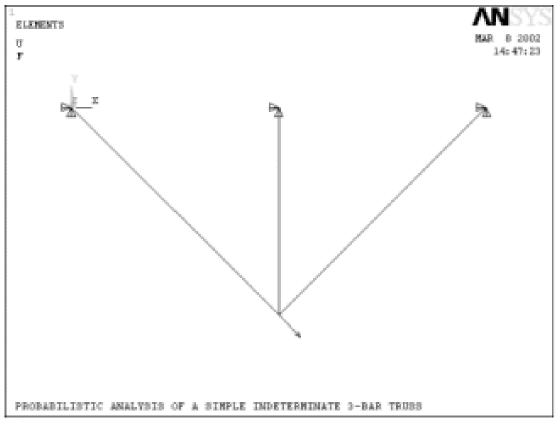
> Young's modulus (E) = 30E6 psi

The following geometric properties are used for this analysis. These properties are the initial cross-sectional areas for each truss member:

ARE1 = 5.0
ARE2 = 5.0
ARE3 = 5.0

3.8.2.1. Problem Sketch

Figure 3.15 The Simple Indeterminate 3-Bar Truss for the Sample Problem


ELEMENTS
U
F
MAR 8 2002
14:47:23
Y
X
PROBABILISTIC ANALYSIS OF A SIMPLE INDETERMINATE 3-BAR TRUSS

3.8.3. Using a Batch File for the Analysis

You can perform the example probabilistic design analysis of this 3-bar truss using the ANSYS commands shown below.

The analysis file is created for use during the probabilistic analysis. It is a parametric model of the problem geometry, materials, and loads. Within the analysis file, input variables are initialized and output variables are retrieved. If you prefer, you can perform the second pass of the example analysis using the GUI method rather than ANSYS commands. See Section 3.8.4: Using the GUI for the PDS Analysis for details.

The following input listing sets up the analysis file for the 3-bar truss.

```
/com
/com, Create an analysis file to be used during looping
/com
*create,pds3bar,pdan
*SET,ARE1,5.00             !INITIALIZE CROSS SECTIONAL AREAS
*SET,ARE2,5.00
*SET,ARE3,5.00
/PREP7
/title, PROBABILISTIC ANALYSIS OF A SIMPLE INDETERMINATE 3-BAR TRUSS
ET,1,1
EX,1,30E6
R,1,ARE1
R,2,ARE2
R,3,ARE3
N,1
N,2,10
N,3,20
N,4,10,-10,,-45            ! ROTATE TIPNODE 45°
REAL,1
E,1,4
REAL,2
E,2,4
REAL,3
E,3,4
D,1,ALL,,,3
F,4,FX,20000
FINISH
/SOLU
SOLVE
FINISH
/POST1
SET,1
```

```
ETABLE,VOLU,VOLU             ! STORE MEMBER VOLUMES
ETABLE,AXST,LS,1             ! STORE MEMBER AXIAL STRESSES
*GET,SIG1,ELEM,1,ETAB,AXST  ! SIG1 IS DEFINED TO BE AXIAL STRESS IN ELEMENT 1
*GET,SIG2,ELEM,2,ETAB,AXST  ! SIG2 IS DEFINED TO BE AXIAL STRESS IN ELEMENT 2
*GET,SIG3,ELEM,3,ETAB,AXST  ! SIG3 IS DEFINED TO BE AXIAL STRESS IN ELEMENT 3
SSUM
*GET,TVOL,SSUM,,ITEM,VOLU
FINI
*end
```

After the analysis file has been created, you can proceed with the probabilistic design analysis. You can do this though ANSYS commands or though the GUI. If you prefer using commands, the following input sets up the probabilistic analysis for the 3-bar truss example.

```
/inp,pds3bar,pdan
/com
/com, Enter PDS and specify the analysis file
/com
/PDS                         ! enter probabilistic design system
pdanl,pds3bar,pdan
/com
/com, Declare random input variables
/com
PDVAR,ARE1,GAUS,5,0.5        ! define area1 with Gaussian distribution
                             ! having mean of 5 and std. dev of 0.5
PDVAR,ARE2,tria,10,11,12     ! define area2 with triangular distribution
                             ! having low bound of 10, most likely point of 11
                             ! and high bound of 12
PDVAR,ARE3,unif,5,6          ! define area3 with uniform distribution
                             ! with low bound of 5 and high bound of 6
/com
/com, Specify any correlations between the random variables
/com
PDCOR,ARE1,ARE3,0.25         ! define a correlation coef of 0.25 between ARE1 and ARE3
/com
/com, Declare random output variables
/com
PDVAR,SIG1,resp              ! define SIG1 a response parameter
PDVAR,SIG2,resp              ! define SIG2 a response parameter
PDVAR,SIG3,resp              ! define SIG3 a response parameter
PDVAR,TVOL,resp              ! define TVOL a response parameter
/com
/com, Choose the probabilistic design tool or method
/com
PDMETH,MCS,DIR               ! specify direct Monte Carlo simulation
PDDMCS,100,NONE,ALL,,,,123457 ! use all 100 samples, initial seed of 123457
/com
/com, Execute the loops required for the probabilistic design analysis
/com
PDEXE,mcs3bar               ! run analysis and define solution label 3bar_mcs
/com
/com, Review the results of the probabilistic analysis
/com
PDSENS,MCS3BAR,TVOL,BOTH,RANK,0.025 !Create Sensitivity plot
fini
/exit
```

3.8.4. Using the GUI for the PDS Analysis

Because of the parametric definition of some variables in the analysis file, the GUI method is not recommended for analysis file creation and is not presented here. It is acceptable, however, to perform the probabilistic design analysis of the 3-bar truss example via the GUI method. The GUI procedure for performing the probabilistic design analysis follows. Section 3.8.3: Using a Batch File for the Analysis describes how to create an analysis file. The GUI procedure for performing the probabilistic design analysis pass follows.

Step 1: Test Analysis File

To test the analysis file, you clear the database and then read input from the pdbeam.lgw file.

1. Choose menu path **Utility Menu> File> Clear & Start New**. Click on OK.
2. In the Verify dialog box, click Yes.
3. Change the jobname. Choose menu path **Utility Menu> File> Change Jobname**. The Change Jobname dialog box appears.
4. Change the jobname to **pds3bar.pdan** and click on OK.
5. Choose menu path **Utility Menu> File> Read Input from**. In the Files list, click on **pds3bar.pdan**. Then click on OK. You see a replay of the entire analysis. Click on Close when the "Solution is done!" message appears.

In the next several steps of this problem, you explore the effects of variation in the parameters.

Step 2: Enter the Probabilistic Design Module and Identify Analysis File

First, enter the optimizer and identify the analysis file.

1. Choose menu path **Main Menu> Prob Design> Analysis File> Assign**. The Assign Analysis File dialog box appears.
2. In the Files list, click once on **pds3bar.pdan** and then click on OK.

Step 3: Identify the Probabilistic Design Variables

1. Choose menu path **Main Menu> Design Opt> Design Variables**. The Design Variables dialog box appears.
2. Click on Add. The Define a Random Variable dialog box appears.
3. In the list of parameter names, click on ARE1. Select GAUSS in the distribution type list. Click on OK.
4. Type 5 in the MEAN VALUE field and 0.5 in the STANDARD DEVIATION field. Click on OK.
5. Click on Add. The Define a Random Variable dialog box appears.
6. In the list of parameter names, click on ARE2. Select TRIANGULAR in the distribution type list. Click on OK.
7. Type 10 in the LOWER BOUNDARY field, 11 in the MOST LIKELY VALUE field, and 12 in the UPPER BOUNDARY field. Click on OK.
8. Click on Add. The Define a Random Variable dialog box appears.
9. In the list of parameter names, click on ARE3. Select UNIFORM in the distribution type list. Click on OK.
10. Type 5 in the LOW BOUND field, and 6 in the HIGH BOUND field. Click on OK.
11. Click on Close to close the Define a Random Variable dialog box.
12. Choose menu path **Main Menu> Prob Design> Prob Definitions> Correlation**. The Add/Edit or Delete Correlation dialog box appears.
13. Select ARE1 and ARE2 from the list of variable names. Click on OK. Type 0.25 in the Correlation Coeff field.
14. Choose menu path **Main Menu> Prob Design> Prob Definitions> Random Output**. The Random Output Parameters dialog box appears.

15. Click on Add. The Define a Random Output Parameter dialog box appears. In the list of parameter names, click on SIG1. Click on OK.
16. Click on Add. The Define a Random Output Parameter dialog box appears. In the list of parameter names, click on SIG2. Click on OK.
17. Click on Add. The Define a Random Output Parameter dialog box appears. In the list of parameter names, click on SIG3. Click on OK.
18. Click on Close to close the Random Output Parameters dialog box.

Step 4: Run the Probabilistic Design Analysis

This step involves specifying the probabilistic design method, executing the run, and saving the data.

1. Choose menu path **Main Menu> Prob Design> Prob Method> Monte Carlo Sims**. The Monte Carlo Simulation dialog box appears.
2. Select Direct Sampling from the Sampling Methods.
3. Type 100 in the Number of Simulations field. Choose Use 123457 from the Random Seed Option. Click on OK.
4. Choose menu path **Main Menu> Prob Design> Run> Exec Serial> Run Serial**. Type `mcs3bar` in the Solution Set Label field. Click on OK.
5. The Run Monte Carlo Simulations dialog box appears. Click on OK.
6. Choose menu path **Main Menu> Prob Design> Prob Database> Save**. Type `mcs3bar` in the selection field. Click on OK.

When the data is saved, you are ready to review the results.

Step 5: Review the Probabilistic Results

In this step, you visualize the probabilistic results.

1. Choose menu path **Main Menu> Prob Design> Prob Results> Trends> Sensitivities**. The Sensitivity of a Response Parameter dialog box appears.
2. Select **MCS3BAR** in the Select Results Set field. Select TVOL in the Select Response Parameter field. Click on OK.
3. A Rank-Order Correlation Sensitivity bar and pie chart is shown in the ANSYS Graphics window.

Step 6: Exit ANSYS

1. Click on Quit in the ANSYS Toolbar.
2. Select an option to save, then click on OK.

Chapter 4: Variational Technology

The results of a finite-element analysis depend on several input variables, such as:

- Material properties
- Physical parameters (for example, the thickness of a plate)
- Boundary conditions

In a traditional finite-element analysis, each change of the value of any input variable requires a new finite-element analysis. To perform a "what-if-study" where several input variables are varied in a certain range, a considerable number of finite-element analyses may be required to satisfactorily evaluate the finite-element results over the space of the input variables.

Variational Technology provides a much more efficient approach by providing a "response surface." A response surface is an explicit approximation function of the finite-element results expressed as a function of all selected input variables. Among other approaches known in literature these approximation functions can be derived from Taylor series expansion or the Padé approximation. The following items characterize each approximation method:

- **Padé approximation** (default) is based on a series of rational functions. Hence, it is capable of better dealing with highly nonlinear cases such as response quantities that have singularities. A recommended upper limit is 100 input variables (problem dependent). It is the only method that works with discrete input variables and is the faster and cheaper method. However, it does not allow for any single finite element in the model to be affected by more than one input variable.
- **Taylor series approximation** is based on a series of polynomial functions. A recommended upper limit is 10 input variables (problem dependent). It is the only method that allows for the finite elements in the model to be affected by more than one input variable or by shape parameters.

Both the Taylor series expansion technique as well as the Padé approximation depends on the order of the approximation function. Naturally, the higher the order of the approximation the more accurate the approximation will be. Variational Technology as implemented in the ANSYS environment automatically determines the necessary order of the approximation based on the requested accuracy of the expected results.

To determine the response surfaces it is necessary to evaluate higher order derivatives of the finite-element results with respect to the selected input variables, where the order of the derivatives corresponds to the order of the approximation function. It is a unique key feature of Variational Technology implemented in the ANSYS environment that all necessary derivatives of any order are calculated automatically within one single finite-element analysis. Because the derivatives are also calculated, this "extended" finite-element analysis may take longer than a regular solve. However, this one "extended" finite-element analysis will take a considerably shorter time if compared to the many solution runs that are required for the "what-if-study" mentioned above. Depending on the analysis problem, typical speed-up factors may be in the order of ten or even up to several thousands.

ANSYS provides two separately licensed Variational Technology modules for the ANSYS environment:

- ANSYS DesignXplorer VT for structural applications: Providing a response surface of finite-element results for linear, elastic, and static analysis types and eigenfrequency analyses.
- ANSYS Frequency Sweep VT: Provides response curves of electromagnetic applications as a function of the excitation frequency.

4.1. ANSYS DesignXplorer VT

4.1.1. What is ANSYS DesignXplorer VT

ANSYS DesignXplorer VT is an add-on module for the ANSYS environment that provides a wide range of accurate, rapid, derived results for structural static analysis with linear elastic materials. ANSYS DesignXplorer VT supports varying the following design parameters:

- Discrete elements and element components. Typical examples of such features would be stiffeners or holes. You can then see the effects of removing these various components from the model. See Section 4.1.3: Element Support for a complete list of supported elements for this application.

  In a discrete element analysis, you must first create element components of the features that you wish to consider as discrete variables. For holes, you must create the elements that define the hole; that is, the geometry that represents the hole must be filled with elements which will be suppressed. For information concerning setting up element components, see Selecting and Components in the *ANSYS Basic Analysis Guide*.
- Material properties, including elastic modulus, material density, and minor Poisson's ratio. See Element Support for a list of supported element types. Note that for orthotropic materials, variations of Young's modulus (E_x, E_y, E_z), shear modulus (G_{xy}, G_{yz}, G_{zx}), and Poisson's ratio (ν_{xy}, ν_{yz}, ν_{zx}) are supported.
- The following real constants or section definition parameters:
  - Shell thickness for SHELL181
  - Mass for MASS21
  - Stiffness for COMBIN14
- Temperature change for thermal stress analysis.

Results are viewed in a separate postprocessor application, called the SolutionViewer, which can be launched from within the ANSYS environment. You can use the SolutionViewer to interactively evaluate and optimize product behavior. This can all be done without the time required for additional finite-element solutions.

The SolutionViewer provides the following tools for design evaluation and optimization:

- Design Curves
- Design Handbooks, consisting of families of design curves.
- Histograms
- Response Surfaces
- Design Sweeps
- Tolerance Analyses
- Contour Plots

4.1.2. Basic Operation

For discrete analyses, you must first create the element components for the discrete variables. For information concerning setting up element components, see Selecting and Components in the *ANSYS Basic Analysis Guide*.

4.1.2.1. Good Practices

Run a Traditional ANSYS Solution Before a ANSYS DesignXplorer VT analysis: Running a single, traditional ANSYS analysis first helps “weed out” any problems that might cause the ANSYS DesignXplorer VT analysis to fail. For example, this will help locate such problems as poorly applied loads, distorted elements, or coincident nodes not merged. You should eliminate any such problems before performing an ANSYS DesignXplorer VT analysis.

Meshing: Having a good mesh is just as important for the ANSYS DesignXplorer VT analyses as it is for traditional ANSYS analyses.

Model Size: While there is no theoretical size limitation on the model used in an ANSYS DesignXplorer VT analysis, there is the practical consideration that very large models may also consume very large amounts of memory. Experience has shown that based on various sizes of a three thickness shell model, the memory required is on an order of magnitude larger than for a single sparse solution.

Number of Input Variables: ANSYS DesignXplorer VT can easily handle a combination of 50 to 100 input variables using the Padé approximation, and 5 to 10 input variables using the Taylor series approximation. While adding significant numbers of input variables will require more memory usage, and more time to arrive at a solution, it will still be many, many times faster than a traditional analysis.

Solving: To perform a static analysis in the ANSYS environment after a solve with Variational Technology, you must use the **STAOPT**,*DEFA* command. After a solution based on ANSYS DesignXplorer VT with the **SOLVE** command, your model is frozen insomuch you cannot add or delete input variables.

4.1.2.2. General Procedure for Using ANSYS DesignXplorer VT

Use the following as a general guideline.

1. Issue the VT processor command (**/SX**) .
2. If desired, clear the previous results from memory with the **SXCLR** command. Setup from previous solutions isn't cleared from memory unless you issue this command. It is a good practice to clear memory before running a new solution that includes different input or results variables.
3. If desired, overwrite the previous results file or specify a new results file using the **SXRFIL** command. This can be quite important as Variational Technology results files can be very large and are not overwritten (merely appended).

   The results file name that you specify with this command is the only results files that can loaded into the SolutionViewerr (through the **SXPOST** command). To load a pre-existing results file into the Solution-Viewer, you must first specify it with this command before issuing **SXPOST**.
4. Choose the Padé or Taylor series approximation function, and the types of derivatives to be evaluated, using the **SXMETH**.
5. Define the design variables for the analysis using the following commands:

   SXDISC
   Defines a discrete (element or element components) variable as an input variable.

   SXMP
   Defines a material property as an input variable.

   SXREAL
   Defines real constant as an input variable.

SXSEC
: Defines a section property as an input variable for the FS Module or the DesignXplorer VT.

SXTEMP
: Defines the temperature as input variable for the DesignXplorer VT.

6. Define the result quantities with one or more **SXRSLT** commands. The variables that you define depend on the type of analysis.
7. Set the Variational Technology solution method depending on the analysis type. For a static analysis (**ANTYPE**,STATIC), set the solution method using the **STAOPT** command. For a modal analysis (**ANTYPE**,MODAL), set the solution method using the **MODOPT** command.
8. Solve.
9. Review results with the **SXPOST** command. This command launches the SolutionViewer.
10. Export the results from the SolutionViewer to either a JPEG file or to a text file for use in other applications, such as Microsoft Excel.
11. An alternative to using **SXPOST** is to use the POST1 processor with **SXVMOD** and **SXEVAL** commands. **SXEVAL** will evaluate the results for parametric values specified for **SXVMOD** and load them into memory. Displacements, reaction forces and stresses can be printed or plotted.
12. Do not issue a set command when using **SXEVAL** as it will use results from the RST file and overwrite them in memory.

4.1.2.3. Additional VT Commands

In addition to the commands used directly in setting up an analysis, there are additional commands for the following tasks:

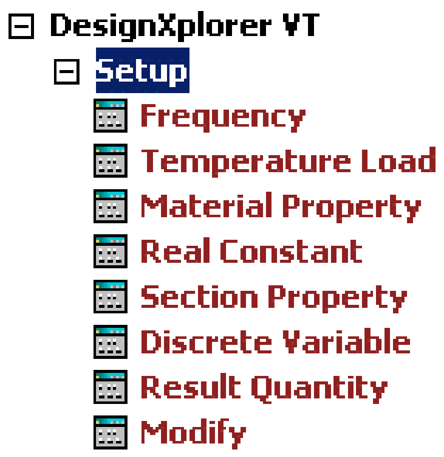
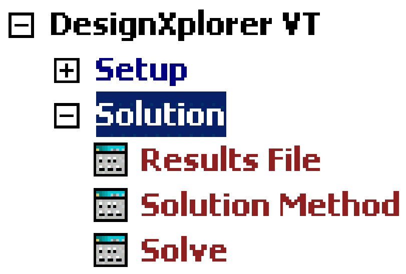
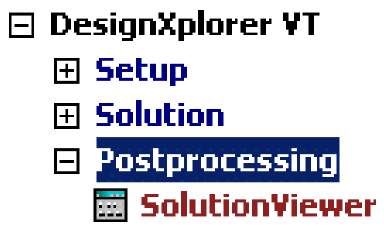
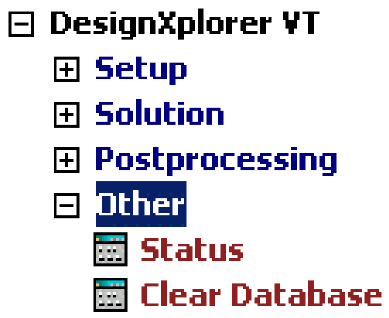
Checking the Status of ANSYS DesignXplorer VT Settings
: You can use the **SXSTAT** command to print the status of all ANSYS DesignXplorer VT command settings in a separate, scrollable window.

Selecting a Subset of Elements Undergoing Variation
: The **SXSL** command allows you to select a subset of elements based on an SX variable name.

Modifying ANSYS DesignXplorer VT Input Variables
: To modify an ANSYS DesignXplorer VT input variable created through the **SXMP**, **SXDISC**, **SXREAL**, or **SXRSLT** commands, you can simply reissue the command with the same input variable name. However, to deactivate, activate, or delete variables, you must use the **SXVMOD** command. Use the **SXSTAT** command to list the names of the input variables.

4.1.2.4. Using ANSYS DesignXplorer VT Interactively

You can perform ANSYS DesignXplorer VT analyses interactively as well. Each of the ANSYS DesignXplorer VT commands has a corresponding dialog box. These are available under **DesignXplorer VT** in the **Main Menu**. The following illustrations map each of the menu items to the appropriate ANSYS DesignXplorer VT command.

 DesignXplorer VT Setup Frequency Temperature Load Material Property Real Constant Section Property Discrete Variable Result Quantity Modify	**Frequency**	**SXFREQ**
	Temperature Load	**SXTEMP**
	Material Property	**SXMP**
	Real Constant	**SXREAL**
	Section Property	**SXSEC**
	Discrete Variable	**SXDISC**
	Result Quantity	**SXRSLT**
	Modify	**SXVMOD**
 DesignXplorer VT Setup Solution Results File Solution Method Solve	**Results File**	**SXRFIL**
	Solution Method	**SXMETH**
	Solve	**SOLVE**
 DesignXplorer VT Setup Solution Postprocessing SolutionViewer	**SolutionViewer**	**SXPOST**
 DesignXplorer VT Setup Solution Postprocessing Other Status Clear Database	**Status**	**SXSTAT**
	Clear Database	**SXCLR**

Setting up ANSYS DesignXplorer VT analyses interactively is considerably easier, as the dialog boxes (at least to some extent) guide you in choosing the proper command settings. For example, the **Define Material Properties** dialog box displays the proper material property labels based on the main material selection. Also, you don't need to specify Variational Technology with the **STAOPT** command, this is done automatically when you select **Solve** under **DesignXplorer VT> Solution** in the **Main Menu**.

4.1.3. Element Support

The following table lists the elements currently supported by ANSYS DesignXplorer VT.

Table 4.1 Elements For Use With ANSYS DesignXplorer VT

Element	Material Properties (defined using SXMP)	Real Constants	Discrete (defined using SXDISC)	Results[a] (as stated in SXRSLT)
LINK180	EX, NUXY, DENS		X	MASS, RF, U
SHELL181	EX, NUXY, DENS	TK (defined using **SXREAL** or **SXSEC**)	X	MASS, S, RF, U
PLANE182	EX, NUXY, DENS		X	MASS, S, RF, U
PLANE183	EX, NUXY, DENS		X	MASS, S, RF, U
SOLID185	EX, NUXY, DENS		X	MASS, S, RF, U
SOLID186	EX, NUXY, DENS		X	MASS, S, RF, U
SOLID187	EX, NUXY, DENS		X	MASS, S, RF, U
BEAM188	EX, NUXY, DENS		X	MASS, RF, U
BEAM189	EX, NUXY, DENS		X	MASS, RF, U
COMBIN14	EX, NUXY, DENS	STIFF (defined using **SXREAL**)	X	RF, U
MASS21	EX, NUXY, DENS	MASS (defined using **SXREAL**)	X	MASS, S, RF, U

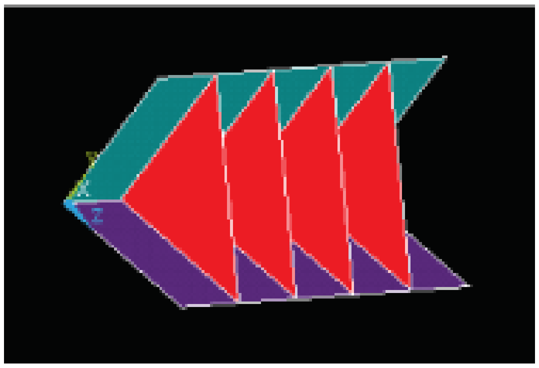
[a]Beam elements have no stress results.

4.1.4. Limitations

- An element cannot belong to more than one variable specification; that is, an element cannot be part of both a thickness and material property variable definition using **SXMETH,,PADE**.
- For Normal Modes analysis:
  - Non-zero prescribed displacements are not supported.
  - The result type MASS cannot be selected.
- **SXTEMP** cannot be used with the following elements:
  - LINK180
  - BEAM188
  - BEAM189
  - COMBIN14
  - MASS21
- On an element which has both supressed DOFs and prescribed DOFs, the calculation of reaction forces is incorrect.

4.1.5. Complete Discrete Analysis Example

The following example builds a cantilever model with a pressure load. The example uses both continuous and discrete variables. Continuous variables are characterized by a non-interrupted range of values, such as thickness, force, or temperature. Discrete variables are valid only at particular values, such as the number of holes or number of weld points. In this example, the flange and web thicknesses are continuous input variables. The four stiffening ribs are set as discrete variables, allowing you to experiment with the effects caused by removing individual ribs. Note that this requires creating an element component for each of the ribs.



A standard analysis is then performed in the ANSYS environment. Then, a DesignXplorer VT analysis is performed following the steps outlined in the table below that shows both the GUI paths and the equivalent ANSYS commands.

Task		GUI Input	Command Input
1.	Enter the ANSYS DesignXplorer VT Preprocessor.	No action required. Occurs automatically at Step 2 or 3.	**/SX**
2.	Define the results file where the DesignXplorer VT results are to be stored.	• **Main Menu> DesignXplorer VT> Solution> Results File** • [Browse ...] and pick, or type **file.rsx** (include path) • [OK]	**SXRFIL,file,rsx**
3.	Identify thickness variables by real constant numbers.	• **Main Menu> DesignXplorer VT> Setup> Real Constant** • “Name for the variable” = Thick1 • “Minimum value of variable” = 0.1 • “Maximum value of variable” = 0.3 • [OK] • “Sel Real Constant Id” = 1 • [Apply]	**SXREAL,Thick1,0.10,0.30,CONT, TK, 1,,VAL**
		• “Name for the variable” = Thick2 • “Minimum value of variable” = 0.05 • “Maximum value of variable” = 0.35 • [OK] • “Sel Real Constant Id” = 2 • [OK]	**SXREAL,Thick2,0.05,0.35,CONT, TK, 2,,VAL**

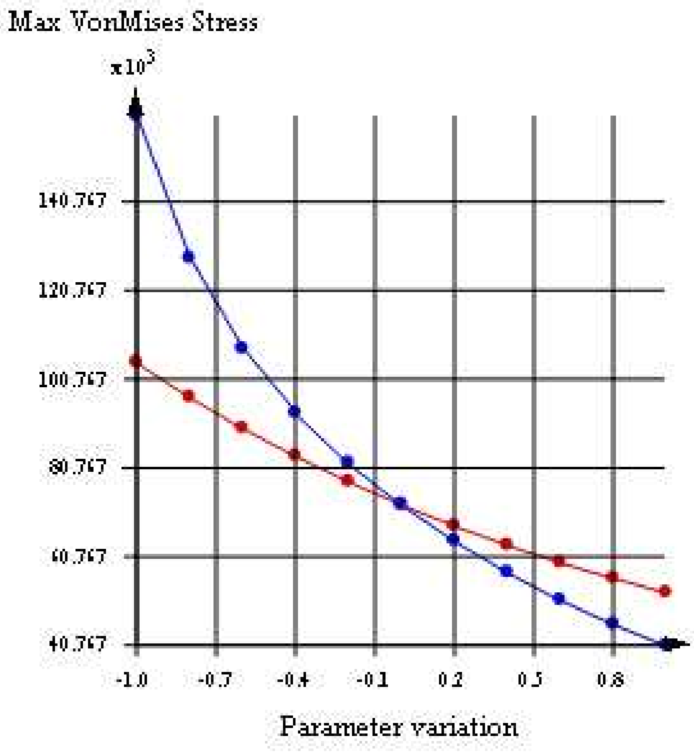
Task		GUI Input	Command Input
4.	Identify discrete variables.	• **Main Menu> DesignXplorer VT> Setup> Discrete Variable** • "Name for the variable" = rib1 • [OK] • "Sel Elem Comp Name" = RIB1 • [Apply]	**SXDISC,RIB1,rib1**
		• "Name for the variable" = rib2 • [OK] • "Sel Elem Comp Name" = RIB2 • [Apply]	**SXDISC,RIB2,rib2**
		• "Name for the variable" = rib3 • [OK] • "Sel Elem Comp Name" = RIB3 • [Apply]	**SXDISC,RIB3,rib3**
		• "Name for the variable" = rib4 • [OK] • "Sel Elem Comp Name" = RIB4 • [OK]	**SXDISC,RIB4,rib4**
5.	Identify desired element and nodal result items.	• **Main Menu> DesignXplorer VT> Setup> Result Quantity** • "Name of result quantity" = defl • "Type Comp" = Nodal Results (left), Displacements U (right) • [OK] • [Apply]	**SXRSLT,defl,NODE,U,ALL,,**
		• "Name of result quantity" = stress • "Type Comp" = Element Results (left), Stresses S (right) • [OK] • [Apply]	**SXRSLT,stress,ELEM,S,ALL,,**
		• "Name of result quantity" = mass • "Type Comp" = Element Results (left), Mass (right) • [OK] • [OK]	**SXRSLT,mass,ELEM,MASS,ALL,,**

Task		GUI Input	Command Input
6.	Review input status listing and obtain a ANSYS DesignXplorer VT solution.	• **Main Menu> DesignXplorer VT> Solution> Solve** • **File> Close** after reviewing • [OK] • Review solution progress in the Output Window.	**SXSTAT** **FINISH** **/SOLU** **STAOPT,SX** **SOLVE** **FINISH**
7.	Open the SolutionViewer to review ANSYS DesignXplorer VT results.	• **Main Menu> DesignXplorer VT> Postprocessing> SolutionViewer** • [OK]	**SXPOST**
8.	Create Design Curve.	• Highlight **file.rsx** in **Objects list**. • **Add curve** • **Add parameter** then pick **Thick1** and **Thick2** • [OK] • **Add criterion** then pick **Max VonMises Stress** • [OK] • Review **Property** list, then **Evaluate** to produce design curve.	

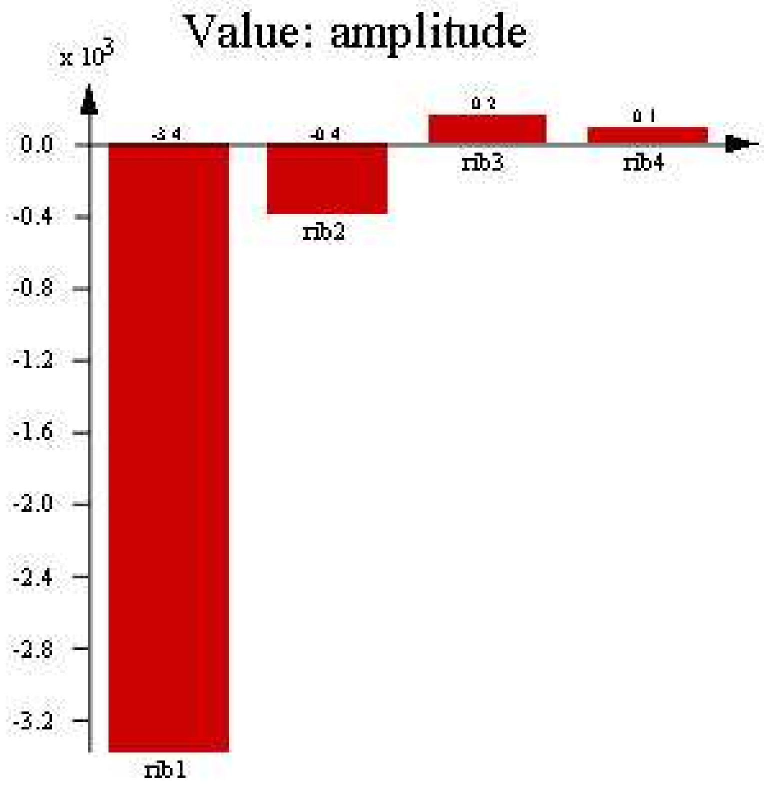
9.	Create Histo-gram.	• Highlight **file.rsx** in **Objects list**. • **Add histogram** • then pick **Rib1**, **Rib2**, **Rib3**and **Rib4** • [OK] • **Add criterion** then pick **Max VonMises Stress** • [OK] • Review **Property** list, then **Evaluate** to produce histogram.	
10.	Create Response Surface.	• Highlight **file.rsx** in **Objects list**. • **Add surface** • **Add parameter** then pick **Thick1** and **Thick2** • [OK] • **Add criterion** then pick **Max VonMises Stress** • [OK] • Review **Property** list, then **Evaluate** to produce response surface.	

Task		GUI Input	Command Input
11.	Create Contour Plot	• Highlight **file.rsx** in **Objects list**. • **Add contour plot** • **Add parameter** then pick **Thick1** • [OK] • **Add criterion** then pick **displacement** • [OK] • Review **Property** list, then **Evaluate** to produce contour plot.	

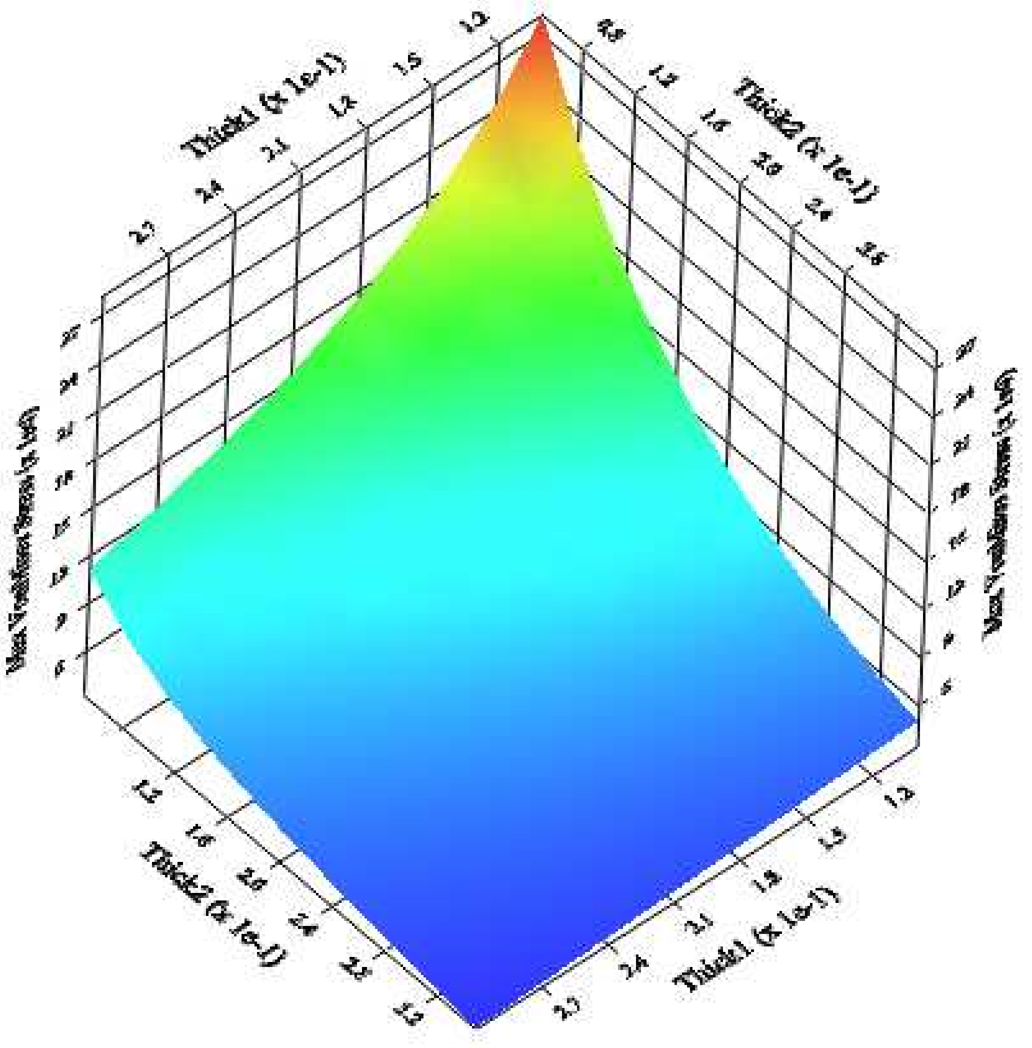
Design Curve



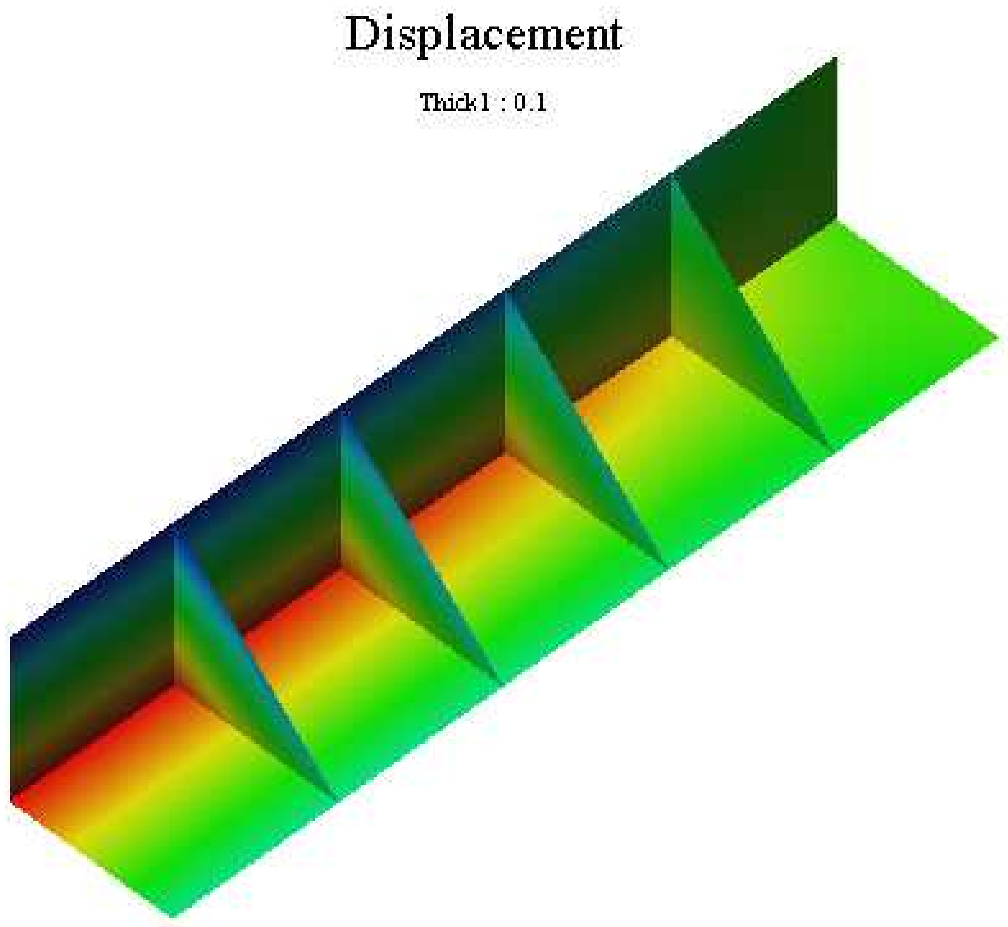
Histogram


Value: amplitude
x 10^3
0.0
-0.4
-0.8
-1.2
-1.6
-2.0
-2.4
-2.8
-3.2
-3.4
-0.4
0.2
0.1
rib1
rib2
rib3
rib4

Response Surface



Contour Plot



Discrete Example Input Listing

```
/PREP7
/TITLE, SXREAL & SXDISC for SHELL181

/PREP7

/COM, Model generation
*SET,a , 1   !Cell length
*SET,b , 1   !Cell Height
*SET,c,1     !Cell Depth
*SET,d,5     !Number of Cells

k,1,0,0
k,2,a,0
k,3,a,b
k,4,0,b
k,5,a,0,c
k,6,0,0,c
a,1,2,3,4
a,1,2,5,6
a,2,3,5
aplot
agen,5,1,2,1,a
agen,d-1,3,3,,a
aplot
aglue,all

/COM, Material generation
MP, EX,        1,  2.068000E+08
MP, NUXY,      1,  2.900000E-01
```

```
MP, DENS,      1,  7.820000E-01
MP, GXY,       1,  8.015504E+07
MP, ALPX,      1,  1.170000E-05
MP, KXX,       1,  4.500000E+04
MP, MU,        1,  0.000000E+00
MP, HF,        1,  0.000000E+00
MP, EMIS,      1,  1.000000E+00
MP, QRATE,     1,  0.000000E+00
MP, VISC,      1,  0.000000E+00

/COM, Element generation
et,1,181
r,1,0.2
et,2,181
r,2,0.2
et,3,181
r,3,0.2
asel,s,loc,z
type,1
real,1

esize,,8

amesh,all
asel,s,loc,y
type,2
real,2
amesh,all
asel,all
type,3
real,3
amesh,all

nsel,s,loc,x,a[1]
esln,,1
cm,rib1,elem

nsel,s,loc,x,2*a
esln,,1
cm,rib2,elem

nsel,s,loc,x,3*a
esln,,1
cm,rib3,elem

nsel,s,loc,x,4*a
esln,,1
cm,rib4,elem

allsel
finish


/solu

FLST,2,5,5,ORDE,5
FITEM,2,1
FITEM,2,18
FITEM,2,20
FITEM,2,22
FITEM,2,24
SFA,P51X,1,PRES,200, ,
allsel,

nsel,s,loc,x,
d,all,all
nsel,s,loc,x,5*a
nsel,r,loc,y,0
nsel,r,loc,z,0

! f,all,mx,10000
! d,all,rotx,10000
```

```
allsel
solve
finish

!  /post1
!  set,last,
!  pldisp,1
!  finish


/sx[2]
SXRFIL,file,rsx[3]
SXREAL,Thick 1,0.10,0.30,CONT,TK,1,,VAL[4]
SXREAL,Thick 2,0.05,0.35,CONT,TK,2,,VAL

SXDISC,Rib1,rib1[5]
SXDISC,Rib2,rib2
SXDISC,Rib3,rib3
SXDISC,Rib4,rib4

SXRSLT,defl,NODE,U,ALL,,[6]
SXRSLT,stress,ELEM,S,ALL,,
SXRSLT,mass,ELEM,MASS,ALL,,
finish

/solution
STAOPT,sx[7]
solve
finish
```

1 - This begins the set of commands to create the element components. Each rib must be defined as an element component for later reference by the **SXDISC** command.

2 - Enter the VT processor with the **/SX** command.

3 - The ANSYS DesignXplorer VT results file is named **file.rsx**.

4 - Input variables for shell thickness are defined.

5 - Each of the previously defined element components is now set as a “discrete” input variable using the **SXDISC** command.

6 - The results variables are defined with the **SXRSLT** command.

7 - The ANSYS DesignXplorer VT solution is selected with the **STAOPT** command.

4.1.6. Shell Thickness Example

In this example, seven real constant sets (1 through 7) defining shell thickness have previously been defined. The following is an example of that code that defines each, in this case real constant set 1.

```
R,1,4.,,,,0.,
RMORE,0.,0.,0.,0.,0.,0.
ET,   1, 181,   0,   0,   0,   0,   0,   0
KEYOPT,     1,  7,     0
KEYOPT,     1,  8,     0
KEYOPT,     1,  9,     0
KEYOPT,     1, 10,     0
```

Material properties were also previously defined, for example:

```
/COM MATERIAL DEFINITION
/COM
MP, EX,        1,  2.068000E+08
MP, NUXY,      1,  2.900000E-01
```

```
MP, DENS,       1,  7.820000E-06
MP, GXY,        1,  8.015504E+07
MP, ALPX,       1,  1.170000E-05
MP, KXX,        1,  4.500000E+04
MP, MU,         1,  0.000000E+00
MP, HF,         1,  0.000000E+00
MP, EMIS,       1,  1.000000E+00
MP, QRATE,      1,  0.000000E+00
MP, VISC,       1,  0.000000E+00
```

Assume also that the model is adequately constrained and a force applied. The following shows the commands used to set up the input variables based on the previously defined shell thicknesses (Thick1 through Thick7) and a material property input variable based on Young's Modulus. The results variables must be defined as well, for use by the results viewer.

```
/SX[1]
SXRFIL,test2,rsx[2]
SXREAL,Thick1,0.5,7.5,CONT,TK,1,,VAL[3]
SXREAL,Thick2,0.5,7.5,CONT,TK,2,,VAL
SXREAL,Thick3,0.5,7.5,CONT,TK,3,,VAL
SXREAL,Thick4,0.5,7.5,CONT,TK,4,,VAL
SXREAL,Thick5,0.5,7.5,CONT,TK,5,,VAL
SXREAL,Thick6,0.5,7.5,CONT,TK,6,,VAL
SXREAL,Thick7,0.5,7.5,CONT,TK,7,,VAL
SXMP,Young0,1.034E+08,3.102E+08,CONT,EX,2,,VAL[4]

SXRSLT,deplacement,NODE,U,ALL,,ALL[5]
SXRSLT,reactions,NODE,RF,ALL,,ALL
SXRSLT,mass,ELEM,MASS,ALL,
SXRSLT,sigma,ELEM,S,ALL,,ALL
FINI


/SOLUTION
STAOPT,SX[6]
```

1 - Enter the VT processor with the **/SX** command.

2 - Specify the results file name with the **SXRFIL** command.

3 - Specify the input variables based on thickness with the **SXREAL** command.

4 - Specify the input variable based on material property with the **SXMP** command.

5 -Specify the results variables with **SXRSLT** command.

6 -Specify a Variational Technology solution with the **STAOPT** command.

4.1.7. Troubleshooting

If you encounter problems running the ANSYS DesignXplorer VT:

1. Make certain that you are using supported elements. Refer to Table 4.1: “Elements For Use With ANSYS DesignXplorer VT” for a list of elements and supported features.
2. Make sure that the **CADOE_LIBDIR81** environment variable has been set. It should point to the **n\Program Files\Ansys Inc\V81\CommonFiles\Language\en-us** directory in Windows and the **/ansys_inc/v81/commonfiles/language/en-us** directory in UNIX.

If you encounter problems running the SolutionViewer help, make sure that the **CADOE_DOCDIR81** environment variable has been set. It should point to **n:\Program Files\Ansys Inc\V81\CommonFiles\help\en-us\solviewer** on Windows and **/ansys_inc/v81/commonfiles/help/en-us/solviewer** on UNIX/Linux.

4.2. ANSYS Frequency Sweep VT

The ANSYS Frequency Sweep VT (FS Module) provides a high-performance solution for forced-frequency simulations in high-frequency electromagnetic problems. This type of analysis uses ANSYS high-frequency elements 119 or 120 only.

The FS Module provides a high-frequency analysis over a range of user-defined frequencies. The module computes S-parameters over the entire frequency range. (For more information, see the **SPSWP** command description.)

In practice, the FS Module completes one normal ANSYS run at the mid-frequency of the specified frequency range. It then performs accurate approximations of the results across the frequency range (in user-specified steps). In addition to controlling the steps and the frequency range, you can also specify the accuracy of the approximations.

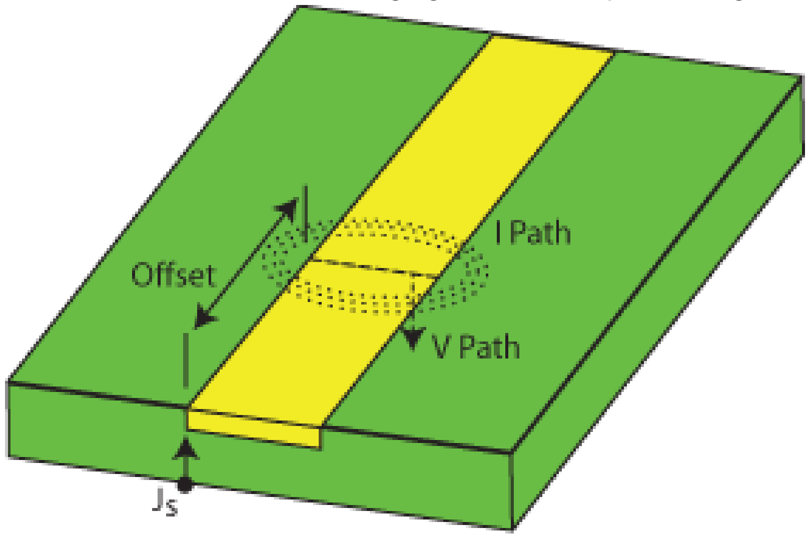
S-parameters are stored in a Touchstone format file, called ***`jobname`*.`s`*`n`*`p`**, where *n* is the number of ports.

To compute a matrix of S-parameters for waveguide ports

1. Define the port regions (flags) and boundary conditions. Use the
   - **BFA**, **BFL**, or **BF** commands for interior waveguide.
   - **SF** or **SFA** commands for exterior waveguide ports.
2. Define the waveguide port type and excitation using the **HFPORT** command.
3. Compute the matrix of S-parameters using the **SPSWP** command.
4. Display the S-parameters using the **PLSP** command.

To compute a matrix of S-parameters for transmission line ports

1. Define the transmission line port (flags) and options.
2. Define a port excitation using the current density (J_s) on the port nodes using the **BF**, **BFL**, or **BFA** commands.
3. Define voltage and current paths with the **PATH** command. The paths must be offset from the port nodes, as shown in the following figure. Save the paths using the **PASAVE** command.



4. Specify transmission line inputs using the **HFPORT** command.
5. Compute the matrix of S-parameters using the **SPSWP** command.
6. Display the S-parameters using the **PLSP** command.

Note — A large number of frequency steps will product a very large database file.

4.2.1. Transmission Line Example Problem

```
/batch,list
/TITLE, Example for SPSWP macro with transmission line ports
/com,
/com, Brick structure: 3 x 1.5 x 7 cm^3 with material step
/prep7
! ---------- geometry parametrs ------------
ch=0.015   $ cw=0.03   $ cl=0.0375
!
! ---------- define elements and material properties -------
et,11,200,7  ! temporary element
et,1,120,1         ! brick 1st order
et,2,120,1,,,1          ! PML elemnts
mp,murx,1,1
mp,perx,1,1
mp,murx,2,1
mp,perx,2,4
! ---------- define geometry and mesh the model ----------
numel=6
rect,-cw/2,cw/2,-ch/2,ch/2
type,11
esize,,cw/numel
amesh,1
type,1                  ! create brick mesh by extrusion
esize,,8
mat,1
asel,s,loc,z,0.
vext,all,,,0,0,-0.8*cl
mat,2
asel,s,loc,z,0.
vext,all,,,0,0,0.8*cl
type,2                  ! create PML elements
esize,,4
mat,1
asel,s,loc,z,-0.8*cl
vext,all,,,0,0,-0.4*cl
mat,2
asel,s,loc,z,0.8*cl
vext,all,,,0,0,0.4*cl
asel,s,loc,z,0.
aclear,all
alls
! ---------- assign PEC boundary conditions ----------
nsel,s,loc,y,-ch/2
nsel,a,loc,y,ch/2
d,all,ax,0.
! ---------- assign PEC BC on PML exterior -----------
nsel,s,loc,z,-cl*1.2
nsel,a,loc,z,cl*1.2
d,all,ax,0.
nsel,all
! ---------- create transmission line port #1 -------------
nsel,s,loc,z,-0.6*cl     ! select port nodes
bf,all,port,1            ! flag port nodes
bf,all,js,0,1,0          ! define excitation
hfport,1,tran,,v1,c1,188.36,0      ! specify port options
! ---------- create transmission line port #2 -------------
nsel,s,loc,z,0.6*cl      ! select port nodes
bf,all,port,2            ! flag port nodes
bf,all,js,0,1,0          ! define excitation
hfport,2,tran,,v2,c2,94.18,0       ! soecify port options
```

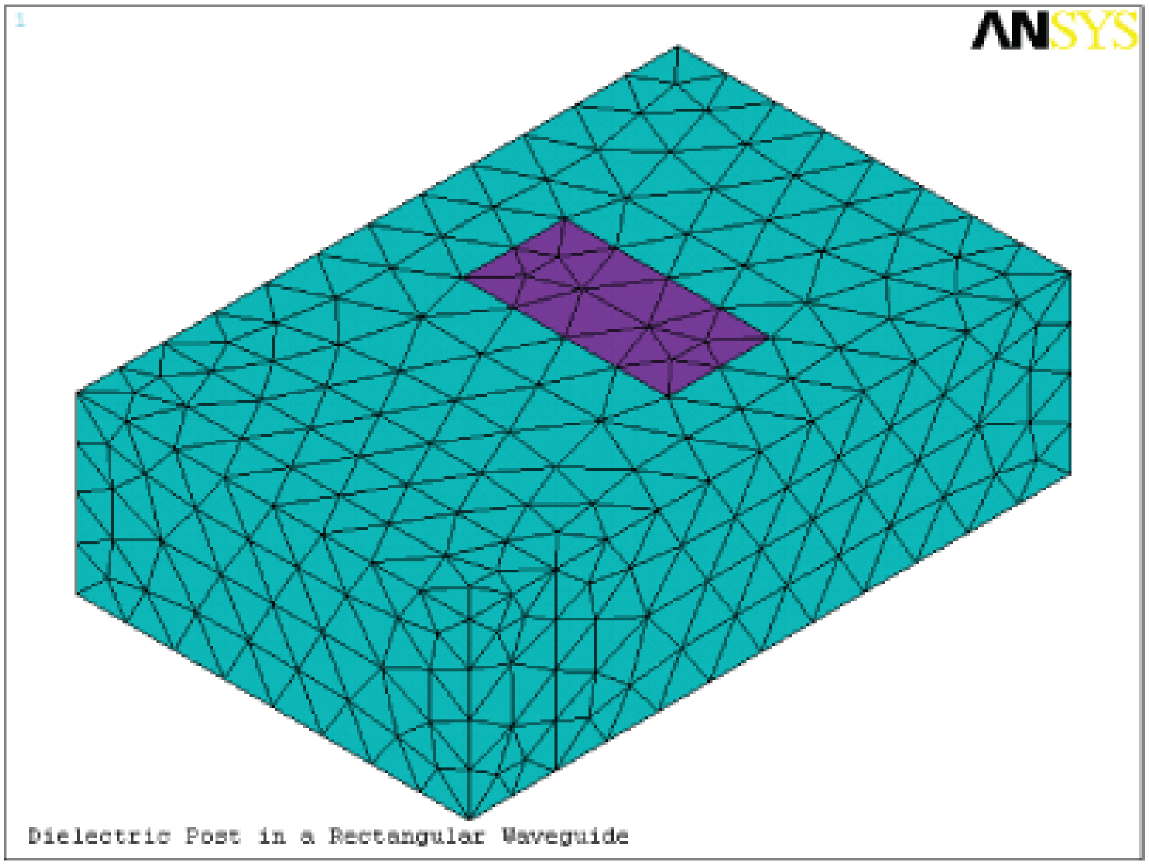
```
! ---------- define paths for transmission line ports ------
x1=-cw/2 $x2=cw/2 $z1=-0.5*cl $z2=0.5*cl $y1=-ch/2 $y2=ch/2
path,v1,2                ! voltage path for port 1
ppath,1,,0,y2,z1
ppath,2,,0,y1,z1
path,c1,2                ! current path for port 1
ppath,1,,x2,0,z1
ppath,2,,x1,0,z1
path,v2,2                ! voltage path for port 2
ppath,1,,0,y2,z2
ppath,2,,0,y1,z2
path,c2,2                ! current path for port 2
ppath,1,,x2,0,z2
ppath,2,,x1,0,z2
pasave,all               ! save paths in file.path
fini
/solu
alls
spswp,1.e9,5.e9,1.e9,0   ! run frequency sweep using Variational Technology option
fini
```

Results in Touchstone Format

```
!  ANSYS S-parameter Data for 2 Ports (Transmission Line).
# GHz S MA R 188.36 94.18.
! Freq     |S11|   <S11     |S21|    <S21     |S12|    <S12     |S22|   <S22
1.0000    0.336  133.538    0.941  -64.494    0.958  -64.522    0.293  -83.051
2.0000    0.374   95.148    0.926 -128.229    0.949 -129.714    0.335  175.810
3.0000    0.370   56.276    0.931  169.804    0.914  168.257    0.435   98.459
4.0000    0.353   24.869    0.939  107.591    0.940  109.070    0.406   43.248
5.0000    0.297  -14.730    0.956   46.633    1.000   47.259    0.220  -16.884
```

4.2.2. Waveguide Example Problem

Figure 4.1 Element Plot for Waveguide Example



```
/batch,list
/title, Dielectric Post in a Rectangular Waveguide
/com, Waveguide Dimension: 22.86 x 10.16 mm^2 (Cutoff Frequency: 6.56 GHz)
/com, Dielectric Post: 12 x 10.16 x 6 mm^3 at the center of waveguide
/com,                  epsr = 8.2
/com, Frequency Range: 8 - 12 GHz for TE10 mode
/com, PML Parameter: 5 Layers with -50 dB
/com, Mesh Size: Free Space Wavelength/15 at 12 GHz
/com, Numerical Model: IMPD Driven Port; PML Output Port
/com, Solution Target: S11 over frequency range using series expansion method

/prep7
ch=10.16e-3    ! waveguide height
cw=22.86e-3    ! waveguide width
c=12e-3         ! post width
d=6e-3      ! post length
epsr=8.2     ! Dielectric constant

freq=10.e9     ! Analysis frequency
fmesh=12e9     ! Mesh frequency
lamda=3.e8/fmesh ! wavelength
h1=lamda/8     ! mesh parameter
cl=5*d      ! waveguide length

et,1,HF119,1
et,2,HF119,1,,,1 ! PML option
mp,murx,1,1.
mp,perx,1,1.
mp,murx,2,1.
mp,perx,2,epsr  ! Dielectric
```

```
block,-cw/2,cw/2,-ch/2,ch/2,-cl/2,cl/2
block,-c/2,c/2,-ch/2,ch/2,-d/2,d/2
vsbv,1,2,,delete,keep
block,-cw/2,cw/2,-ch/2,ch/2,cl/2,cl/2+lamda/5
vglue,all

esize,h1
type,1
mat,2      ! Dielectric region
vmesh,2
mat,1      ! Air region
vmesh,4
type,2     ! PML region
vmesh,1

! Tangential E is zero on all side walls

nsel,s,loc,y,-ch/2
nsel,a,loc,y,ch/2
nsel,a,loc,x,-cw/2
nsel,a,loc,x,cw/2
nsel,a,loc,z,cl/2+lamda/5
d,all,ax,0.

nsel,s,loc,z,-cl/2 ! locate rectangular port
sf,all,port,1
hfport,1,rect,,te10,impd,cw,ch,1
alls
fini

/solu
alls
spswp,8.e9,12.e9,0.1e9,0 ! run frequency sweep using Variational Technology option
FINISH
/post1
plsp,1,1,1  ! plot S11 (dB)
plsp,2,1,1  ! Plot Phase Angle (Deg.)
finish
```

Figure 4.2 Graph of Phase Angle

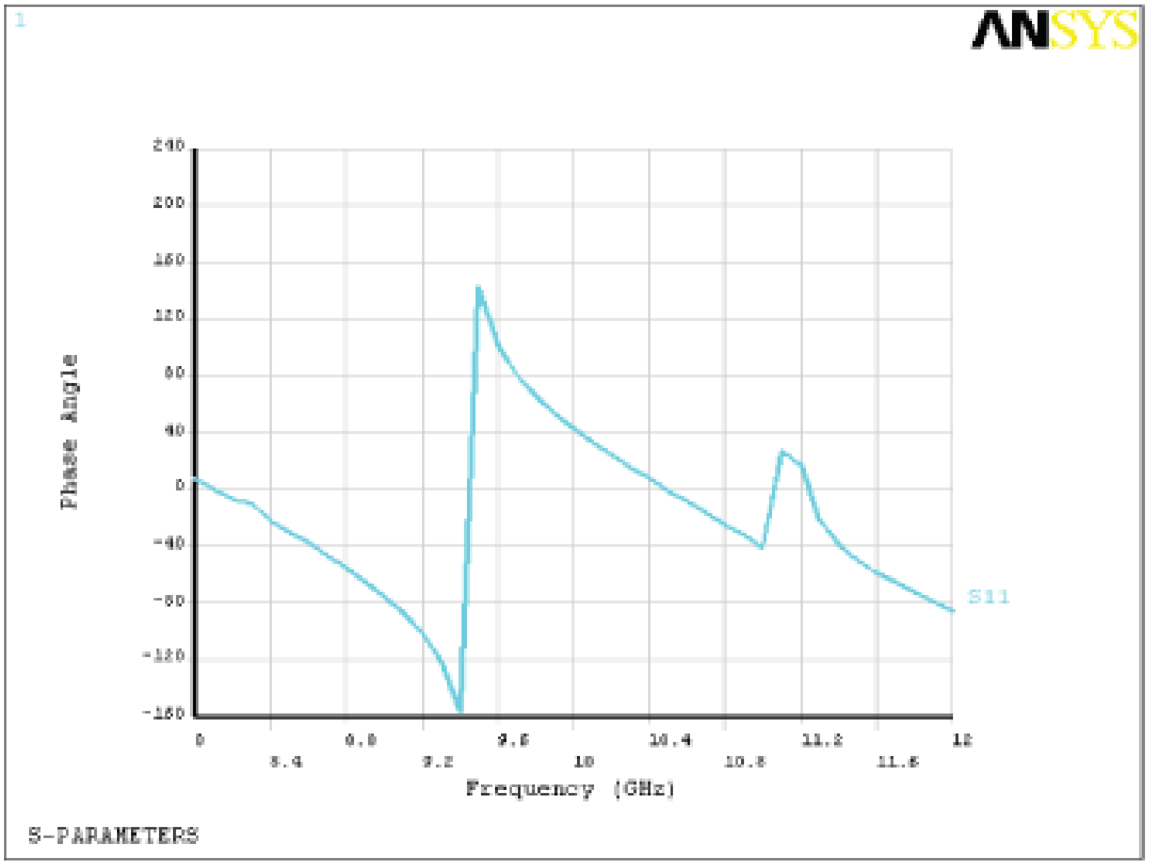
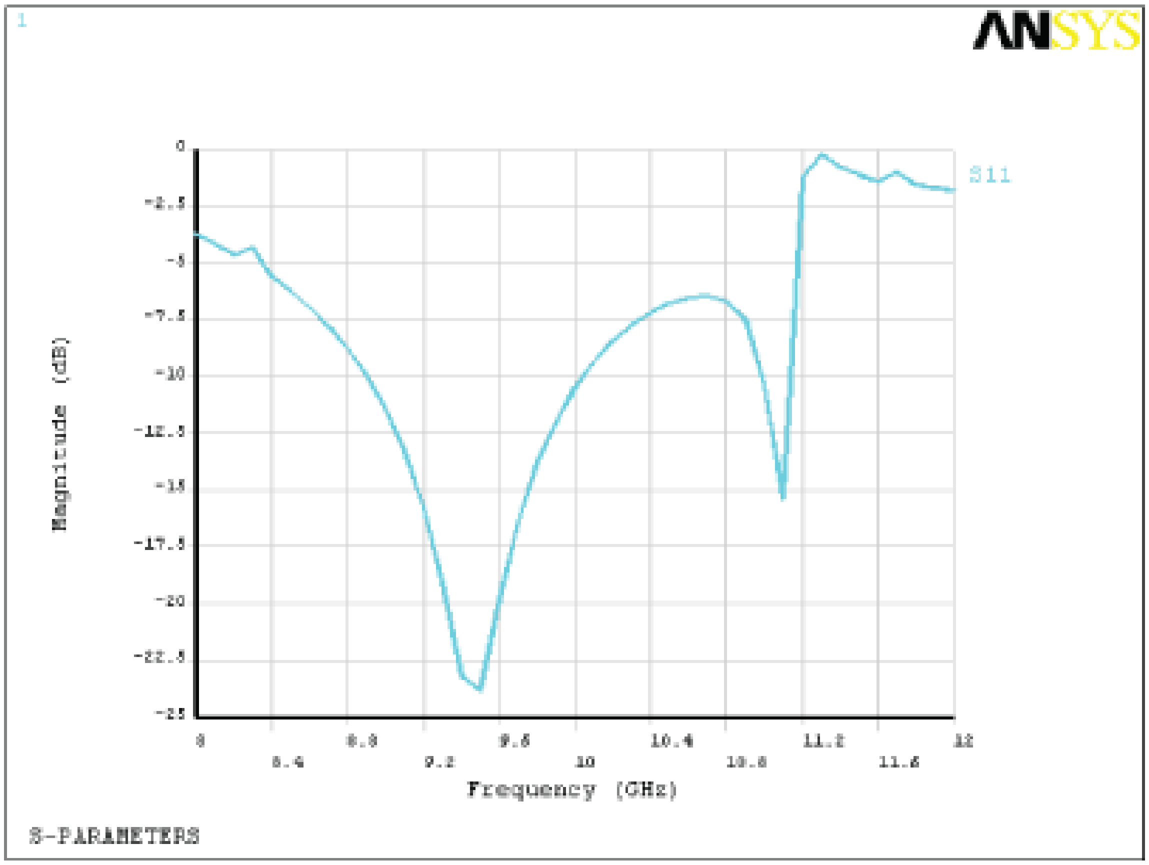


Figure 4.3 Graph of Magnitude



Chapter 5: Adaptive Meshing

The ANSYS program provides approximate techniques for estimating mesh discretization error for certain types of analyses. Using this measure of mesh discretization error, the program can then determine if a particular mesh is fine enough. If it is not, the program will automatically refine the mesh so that the measured error will decrease. This process of automatically evaluating mesh discretization error and refining the mesh, called *adaptive meshing*, continues through a series of solutions until the measured error drops below some user-defined value (or until a user-defined limit on the number of solutions is reached).

For more information about error-value approximation, see The General Postprocessor (POST1) in the *ANSYS Basic Analysis Guide*.

5.1. Prerequisites for Adaptive Meshing

The ANSYS program includes a prewritten macro, **ADAPT.MAC**, to perform adaptive meshing. Your model must satisfy certain preconditions before you can successfully activate the **ADAPT** macro. (In some cases, models that do not conform to these criteria can be adaptively meshed using a modified procedure, as discussed below.) These requirements include the following:

- The standard ADAPT procedure is valid only for single-solution linear static structural and linear steady-state thermal analyses.
- Use only *one* material type, as the error calculations are based in part on *average* nodal stresses, and would thus often be invalid at the material interfaces. Also, an element's error energy is affected by its elastic modulus; therefore, even if the stress discontinuity is the same in two adjoining elements, their error energy will be different if they have different material properties. You should also avoid abrupt changes in shell thickness, as such discontinuities will cause similar stress-averaging problems.
- Use element types that support error calculations.
- Use *meshable* solid model entities. (Avoid characteristics that will cause meshing failure.)

Table 5.1 Element Types That Can Be Used in Adaptive Meshing

Type	Element	Description
2-D Structural Solids	PLANE2	2-D 6-Node Triangular Structural Solid
	PLANE25	Axisymmetric Harmonic 4-Node Structural Solid
	PLANE42	2-D Structural Solid
	PLANE82	2-D 8-Node Structural Solid
	PLANE83	Axisymmetric Harmonic 8-Node Structural Solid
3-D Structural Solids	SOLID45	3-D Structural Solid
	SOLID64	3-D Anisotropic Structural Solid
	SOLID92	3-D 10-Node Tetrahedral Structural Solid
	SOLID95	3-D 20-Node Structural Solid
3-D Structural Shells	SHELL43	4-Node Plastic Large Strain Shell
	SHELL63	Elastic Shell
	SHELL93	8-Node Structural Shell
2-D Thermal Solids	PLANE35	2-D 6-Node Triangular Thermal Solid
	PLANE75	Axisymmetric-Harmonic 4-Node Thermal Solid
	PLANE55	2-D Thermal Solid
	PLANE77	2-D 8-Node Thermal Solid
	PLANE78	Axisymmetric-Harmonic 8-Node Thermal Solid

Type	**Element**	**Description**
3-D Thermal Solids	SOLID70 SOLID87 SOLID90	3-D Thermal Solid 3-D 10-Node Tetrahedral Thermal Solid 3-D 20-Node Thermal Solid
3-D Thermal Shells	SHELL57	Thermal Shell

5.2. Employing Adaptive Meshing

The general process for running the adaptive meshing macro follows:

1. As in any linear static structural or steady state thermal analysis, enter the preprocessor (**/PREP7** command or menu path **Main Menu> Preprocessor**) and specify the element type, real constants, and material properties (according to the prerequisites).
2. Model your system using solid modeling procedures, creating meshable areas or volumes describing the geometry of your system. It is not necessary to specify element sizes, nor do you need to mesh these areas and volumes; the **ADAPT** macro initiates meshing automatically. (If you need to mesh your model with *both* area and volume elements, create an **ADAPTMSH.MAC** subroutine.
3. You can either proceed to SOLUTION (**/SOLU** or menu path **Main Menu> Solution**) or remain in PREP7 to specify analysis type, analysis options, loads, and load step options. Apply only solid model loads and inertia loads (accelerations, rotational accelerations, and rotational velocities) in a *single* load step. (Finite element loads, coupling, and constraint equations can be introduced through the **ADAPTBC.MAC** subroutine. Multiple load steps can be introduced through the **ADAPTSOL.MAC** subroutine.)
4. If in **/PREP7**, exit the preprocessor (**FINISH**). (You can invoke the **ADAPT** macro from either SOLUTION or the Begin level.)
5. Invoke the adaptive solution. To do so, use one of these methods:

   Command(s): ADAPT
   GUI: Main Menu> Solution> Solve> Adaptive Mesh

   You can use the **ADAPT** macro in either a thermal or a structural analysis, but you cannot combine the two disciplines in one adaptive solution. As the adaptive meshing iterations proceed, element sizes are adjusted (within the limits set by *FACMN* and *FACMX*) to decrease and increase the elemental error energies until the error in energy norm matches the target value (or until the specified maximum number of solutions has been used).

   After you have invoked the adaptive solution, this macro controls all program operations until the solution is completed. The **ADAPT** macro will define element sizes, generate the mesh, solve, evaluate errors, and iterate as necessary till the target value of error in energy norm is met. All these steps are performed automatically, with no further input required from you.
6. When adaptive meshing has converged, the program automatically turns element shape checking on (**SHPP**,ON). It then returns to the SOLUTION phase or to the Begin level, depending on which phase you were in when you invoked **ADAPT**. You can then enter **/POST1** and postprocess as desired, using standard techniques.

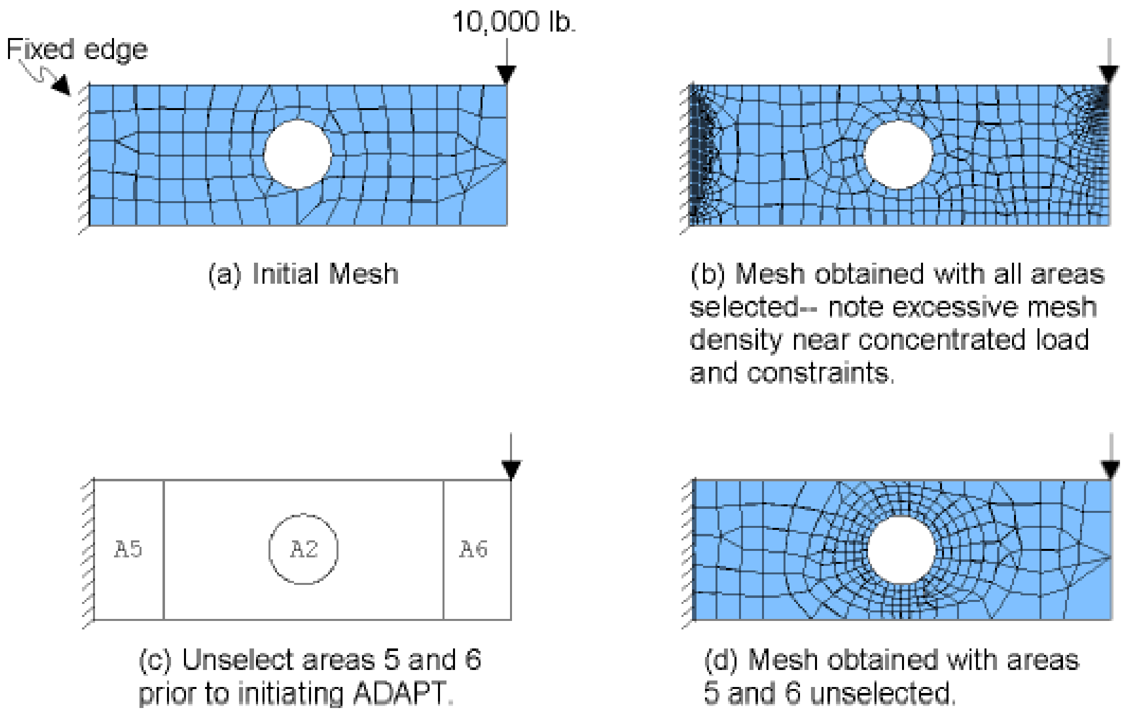
5.3. Modifying the Adaptive Meshing Process

This section describes how to modify adaptive meshing by employing selective adaptivity, enhancing the ADAPT macro's functionality via user subrountines, and customizing the ADAPT macro itself via the UADAPT macro.

5.3.1. Selective Adaptivity

If you know that mesh discretization error (measured as a percentage) is relatively unimportant in some regions of your model (for instance, in a region of low, slowly-changing stress), you can speed up your analysis by excluding such regions from the adaptive meshing operations. Also, you might want to exclude regions near singularities caused by concentrated loads. Selecting logic provides a way of handling such situations.

Figure 5.1 Selective Adaptivity


10,000 lb.
Fixed edge
(a) Initial Mesh
(b) Mesh obtained with all areas selected-- note excessive mesh density near concentrated load and constraints.
A5
A2
A6
(c) Unselect areas 5 and 6 prior to initiating ADAPT.
(d) Mesh obtained with areas 5 and 6 unselected.

Selective adaptivity can improve the performance of models having concentrated loads.

If you select a set of keypoints, the **ADAPT** macro will still include *all* your keypoints (that is, the **ADAPT** macro will modify the mesh at both selected and non-selected keypoints), unless you also set *KYKPS* = 1 in the **ADAPT** command (**Main Menu> Solution> Solve> Adaptive Mesh**).

If you select a set of areas or volumes, the **ADAPT** macro will adjust element sizes only in those regions that are in the selected set. You will have to mesh your *entire* model in PREP7 before executing **ADAPT**.

5.3.2. Customizing the ADAPT Macro with User Subroutines

The standard **ADAPT** macro might not always be applicable to your particular analysis needs. For instance, you might need to mesh both areas and volumes, which is not possible with the standard macro. For this and other such situations, you can modify the **ADAPT** macro to suit your analysis needs. By using a macro to perform the adaptive meshing task, we have intentionally given you access to the logic involved, and have thereby furnished you with the capability for modifying the technique as you might desire.

Fortunately, you do not always need to change the coding within the **ADAPT** macro to customize it. Three specific portions of the macro can be readily modified by means of *user subroutines*, which are separate user files that you can create and that will be called by the **ADAPT** macro. The three features that can be modified by user subroutines are:

- the meshing command sequence

- the boundary condition command sequence,
- the solution command sequence

The corresponding user subroutine files must be named **ADAPTMSH.MAC**, **ADAPTBC.MAC**, and **ADAPTSOL.MAC**, respectively.

5.3.2.1. Creating a Custom Meshing Subroutine (ADAPTMSH.MAC)

By default, if your model contains one or more selected volumes, the ADAPT macro will mesh only volumes (no area meshing will be done). If you have no selected volumes, then the ADAPT macro will mesh only areas. If you desire to mesh both volumes and areas, you can create a user subroutine, **ADAPTMSH.MAC**, to perform all the desired operations. You will need to clear any specially-meshed entities before remeshing. Such a subroutine might look like this:

```
C*** Subroutine ADAPTMSH.MAC - Your name - Job Name - Date Created
TYPE,1         ! Set element TYPE attribute for area meshing
ACLEAR,3,5,2   ! Clear areas and volumes to be meshed by this subroutine
VCLEAR,ALL
AMESH,3,5,2    ! Mesh areas 3 and 5 (no other areas will be meshed by ADAPT)
TYPE,2         ! Change element type for volume mesh
VMESH,ALL      ! Mesh all volumes
```

Please see the **TYPE**, **ACLEAR**, **VCLEAR**, **AMESH**, and **VMESH** command descriptions for more information.

We strongly recommend that you include a comment line (C***) to identify your macro uniquely. This comment line will be echoed in the job printout, and will provide assurance that the **ADAPT** macro has used the correct user subroutine.

5.3.2.2. Creating a Custom Subroutine for Boundary Conditions (ADAPTBC.MAC)

The **ADAPT** macro clears and remeshes with every new solution loop. As a result, your model's nodes and elements will be repeatedly changing. This situation generally precludes the use of finite element loads, DOF coupling, and constraint equations, all of which must be defined in terms of specific nodes and elements. If you need to include any of these finite-element-supported items, you can do so by writing a user subroutine, **ADAPTBC.MAC**. In this subroutine, you can select nodes by their location, and can then define finite element loads, DOF coupling, and constraint equations for the selected nodes. A sample **ADAPTBC.MAC** subroutine follows:

```
C*** Subroutine ADAPTBC.MAC - Your name - Job Name - Date Created
NSEL,S,LOC,X,0   ! Select nodes @ X=0.0
D,ALL,UX,0       ! Specify UX=0.0 for all selected nodes
NSEL,S,LOC,Y,0   ! Select nodes @ Y=0.0
D,ALL,UY,0       ! Specify UY=0.0 for all selected nodes
NSEL,ALL         ! Select all nodes
```

5.3.2.3. Creating a Custom Solution Subroutine (ADAPTSOL.MAC)

The default solution command sequence included in the **ADAPT** macro is simply:

```
/SOLU
SOLVE
FINISH
```

This default sequence will solve only a single load step. You might be able to implement other solution sequences by incorporating them into the user subroutine **ADAPTSOL.MAC**.

5.3.2.4. Some Further Comments on Custom Subroutines

You can create the **ADAPTMSH.MAC**, **ADAPTBC.MAC** and **ADAPTSOL.MAC** subroutine files as you would any user files.

You can use either the APDL command ***CREATE** (**Utility Menu> Macro> Create Macro**), the APDL command ***END**, or an third-party text editor. When the **ADAPT** macro calls the subroutines, the program will search first through the ANSYS root directory, then through your root directory, and last through the current directory. Thus, you should take care to ensure that no identically-named files from another directory will be read in place of your intended files. The comment lines (C***) shown in the example subroutines above would be echoed in your printout, and would provide one means of checking that the proper files were used. Additionally, executing **/PSEARCH**,OFF (**Utility Menu> Macro> Macro Search Path**) before running the **ADAPT** macro would restrict the file search to the ANSYS root directory and your current directory, reducing somewhat the possibility that the wrong file will be used.

No matter where they reside, the subroutines will be accessed only if you set *KYMAC* = 1 on the **ADAPT** command (**Main Menu> Solution> Solve> Adaptive Mesh**). See the *Guide to ANSYS User Programmable Features* for more information on user subroutines and the *ANSYS APDL Programmer's Guide* for more information on APDL.

5.3.3. Customizing the ADAPT Macro (UADAPT.MAC)

For those cases when you need to modify the **ADAPT** macro but are unable to do so via separate user subroutines, you can modify the main body of the macro.

To maintain the integrity of the **ADAPT** macro, ANSYS does *not* recommend modifying the **ADAPT.MAC** file directly. Instead, use the **UADAPT.MAC** file (an identical copy of **ADAPT.MAC**), provided on the ANSYS installation media.

If you modify the **UADAPT.MAC** file, it is a good idea to rename the modified file (especially to denote the specific version that you have created). Afterwards, instead of issuing the command **ADAPT**, call the modified adaptive meshing procedure by issuing the new file name.

Be aware that if you use the new file name as an "unknown command," the program will first search through the high-level directory, then through the login directory, and finally through the working directory, until the macro is found. If a modified adaptive procedure is to be accessible by a single user, it makes sense to store the file in a directory no higher than the user's login directory. If the macro file is stored in such a low-level directory, the file search can be streamlined by calling the macro using the ***USE** command (**Utility Menu> Macro> Execute Data Block**) in place of the unknown command format.

5.4. Adaptive Meshing Hints and Comments

Use the following hints to enhance your implementation of adaptive meshing:

- No initial element sizes are required, but they may be specified to aid the adaptive convergence when desired. If you specify initial element sizes at keypoints, the **ADAPT** macro will use these sizes in its first loop, and will adjust these sizes as appropriate in subsequent loops. To specify initial element sizes, use one of these methods:

  Command(s): KESIZE
  GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> ManualSize> Keypoints> All KPs
  Main Menu> Preprocessor> Meshing> Size Cntrls> ManualSize> Keypoints> Picked KPs

If you specify line divisions or spacing ratios, the **ADAPT** macro will use these values in *every* loop. (That is, line divisions or spacing ratios that you specify will *not* be changed by the **ADAPT** macro.) If you do not specify mesh divisions of any kind, default element sizing (**SMRTSIZE** and **DESIZE**) will be used for the initial mesh. To specify line divisions or spacing ratios, use one of these methods:

Command(s): LESIZE
GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> ManualSize> Lines> All Lines
Main Menu> Preprocessor> Meshing> Size Cntrls> ManualSize> Lines> Picked Lines

- Mapped meshing (all quadrilaterals) is available for 2-D solid and 3-D shell analyses. The benefits of using mapped area meshing are usually minimal, however.
- Mapped meshing (all hexahedral bricks) is available for 3-D solid analyses. Mapped volume meshing, if possible for a given model, will probably give superior performance, compared to tetrahedral meshing.
- In general, midside-node elements will perform better than linear elements in an adaptive meshing operation.
- Do not use concentrated loads or other singularities (such as sharp re-entrant corners), because the **ADAPT** procedure cannot show energy value convergence with mesh refinement in the vicinity of these singularities. If a concentrated loading condition is present in your model, replace it with an equivalent pressure load, applied over a small area. (Or, exclude the region of the concentrated load from adaptive modification using the select options discussed previously.)
- For many problems, it could be preferable to use a number of relatively small, simple regions in place of fewer large, complicated regions, for best meshing performance.
- If the location of maximum response is known or suspected beforehand, then a keypoint should be placed near that location.
- If the **ADAPT** procedure is used in an interactive run and the ANSYS program aborts abruptly without issuing the proper error message, then the output file for the adaptive meshing portion of your run (**Jobname.ADPT**) should be reviewed to determine the cause.
- Similarly, if the **ADAPT** procedure is used in a batch run, then **Jobname.ADPT** should be saved and examined in case of an abrupt abort.
- If a model has an excessive curvature in some region, then your model might experience mesh failure. In this case, use the *SIZE* field on the **KESIZE** command (**Main Menu> Preprocessor> Meshing> Size Cntrls> ManualSize> Keypoints> Picked KPs**) to define the maximum element edge length settings at keypoints near the excessive curvature region. *FACMX* should also be set equal to 1 (in the **ADAPT** command) so that element size will not be permitted to grow in the vicinity of the excessive curvature.
- You should save the results file (**Jobname.RST** or **Jobname.RTH**). In case of a program abort in the middle of the **ADAPT** procedure, the results file will still have the entire analysis data from the previous solution completed by the **ADAPT** procedure.
- You should issue a **SAVE** (**Utility Menu> File> Save as Jobname.db**) before starting the adaptive meshing operation. In case of system abort, **Jobname.DB** can then be used to restart the analysis.

5.5. Where to Find Examples

The *ANSYS Verification Manual* describes several analyses that demonstrate adaptive meshing.

The *ANSYS Verification Manual* consists of test case analyses demonstrating the analysis capabilities of the ANSYS program. While these test cases demonstrate solutions to realistic analysis problems, the *ANSYS Verification Manual* does not present them as step-by-step examples with lengthy data input instructions and printouts. However, most ANSYS users who have at least limited finite element experience should be able to fill in the missing details by reviewing each test case's finite element model and input data with accompanying comments.

The *ANSYS Verification Manual* contains the following adaptive meshing test cases:

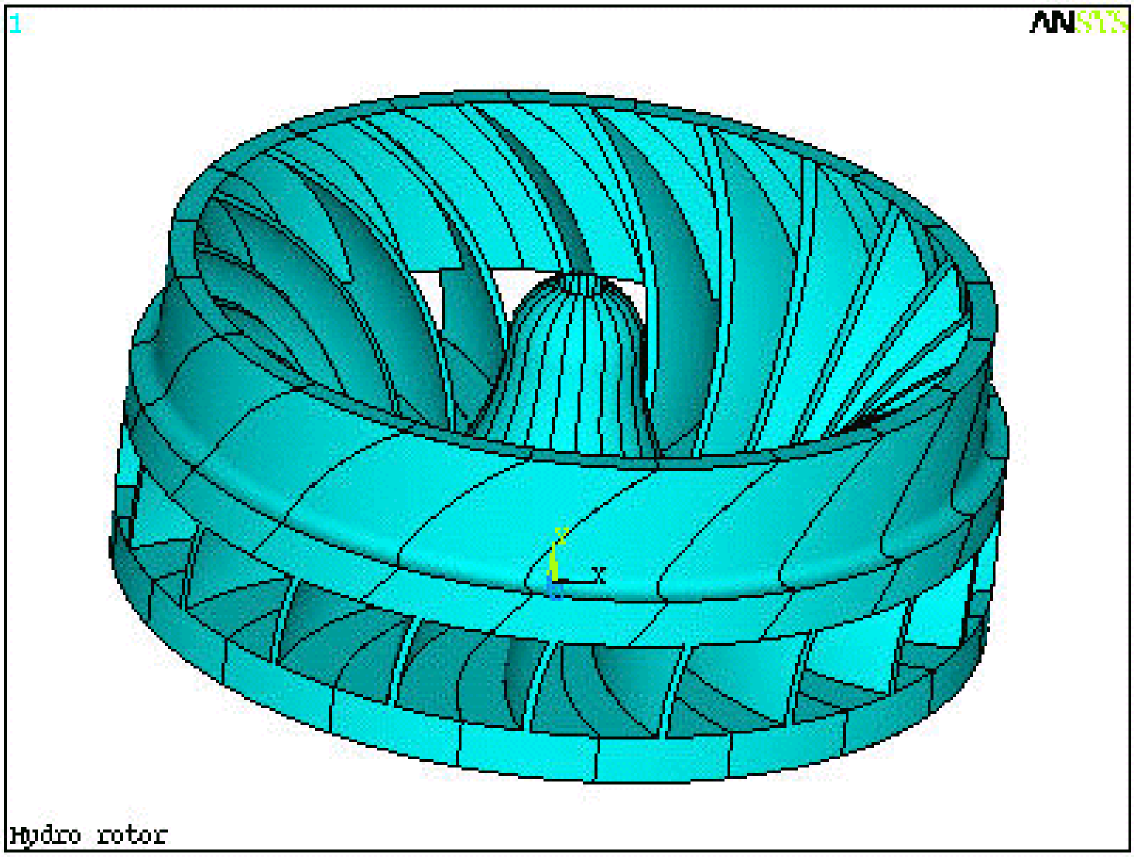
VM193 - Adaptive Analysis of 2-D Heat Transfer with Convection
VM205 - Adaptive Analysis of an Elliptic Membrane Under a Uniformly Distributed Load

Chapter 6: Cyclic Symmetry Analysis

A component or assembly is cyclically symmetric if it has a correspondence in form or arrangement of parts (that is, repetitive patterns) centered around an axis.

A fan wheel is a typical example of a cyclically symmetric structure, as is a spur gear. Another example is this model of a hydro rotor created in ANSYS:

Figure 6.1 Hydro Rotor -- ANSYS Model of a Cyclically Symmetric Structure



This chapter describes cyclic symmetry concepts and the process involved in a cyclic symmetry analysis. The following pages cover these topics:

- Section 6.1: Understanding Cyclic Symmetry Analysis
- Section 6.2: Cyclic Modeling
- Section 6.3: Solving a Cyclic Symmetry Analysis
- Section 6.4: Postprocessing a Cyclic Symmetry Analysis
- Section 6.5: Sample Modal Cyclic Symmetry Analysis
- Section 6.6: Sample Buckling Cyclic Symmetry Analysis

6.1. Understanding Cyclic Symmetry Analysis

If a structure exhibits cyclic symmetry, you can perform an ANSYS-automated static, modal or buckling analysis on the structure. Automated cyclic symmetry analysis is available in the ANSYS Multiphysics, ANSYS Mechanical and ANSYS Structural products.

This section presents the following topics:

- Section 6.1.1: How ANSYS Automates a Cyclic Symmetry Analysis
- Section 6.1.2: Commands Used in a Cyclic Symmetry Analysis

6.1.1. How ANSYS Automates a Cyclic Symmetry Analysis

An ANSYS-automated cyclic symmetry analysis conserves time and CPU resources and allows you to view analysis results on the entire structure. ANSYS automates cyclic symmetry analysis by:

- Solving for the behavior of a *single* symmetric sector (part of a circular component or assembly)
- Using the single-sector solution to construct the response behavior of the full circular component or assembly (as a postprocessing step).

For example, by analyzing a single 10° sector of a 36-blade turbine wheel assembly, you can obtain the complete 360° model solution via simple postprocessing calculations. Using twice the usual number of degrees of freedom (DOFs) in this case, the single sector represents a 1/18th part of the model.

6.1.2. Commands Used in a Cyclic Symmetry Analysis

The most important command in an automated cyclic symmetry analysis is **CYCLIC**, which initiates a cyclic analysis and configures the database accordingly. The command automatically detects cyclic symmetry model information such as edge components, the number of sectors, the sector angles, and the corresponding cyclic coordinate system.

The **ANTYPE** command specifies the analysis type (for example, static, modal or buckling), and the **SOLVE** command obtains the cyclic solution.

Other cyclic-specific commands include:

- **CYCOPT** for specifying solution options
- **/CYCEXPAND** for graphically expanding displacements, stresses and strains of a cyclically symmetric model
- **CYCPHASE** for determining minimum and maximum possible result values from frequency couplets during postprocessing (**/POST1**).

Depending upon the type of cyclic symmetry analysis that you want to perform and your specific needs, it may be necessary to issue other ANSYS commands. For example:

- In a prestressed modal cyclic symmetry analysis, you must issue the **PSTRES**,ON command during the static portion of the analysis to apply the prestress effects.
- During postprocessing, you may want to issue the **ANCYC** command to apply a traveling wave animation to your cyclic model.

The sections of this document describing various cyclic symmetry analyses mention such commands as necessary. For more information, see Section 6.3: Solving a Cyclic Symmetry Analysis.

6.2. Cyclic Modeling

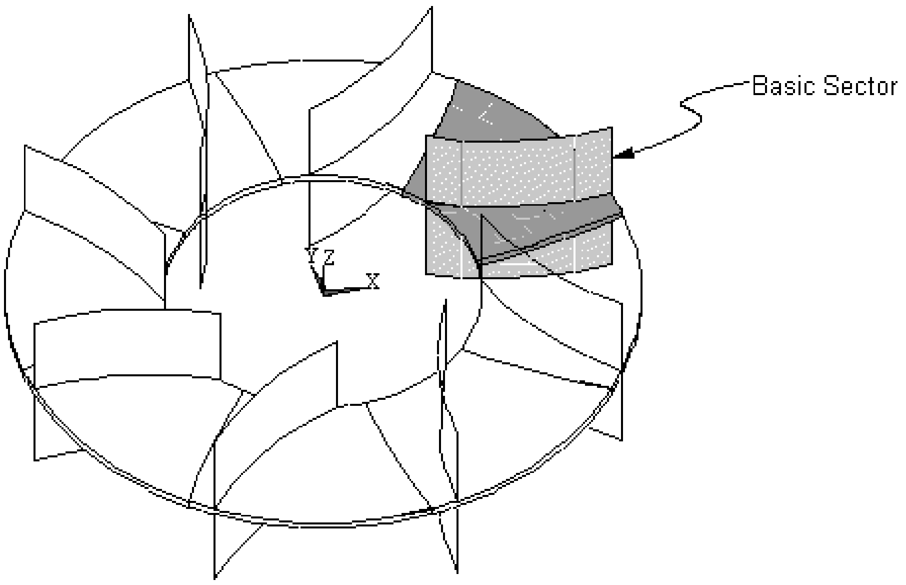
This section describes how to set up a cyclic sector model, discusses important considerations for edge component pairs, and shows how to verify the cyclically symmetric model. The following pages cover these topics:

- Section 6.2.1: The Basic Sector
- Section 6.2.2: Edge Component Pairs
- Section 6.2.3: Model Verification (Preprocessing)

6.2.1. The Basic Sector

A cyclic symmetry analysis requires that you model a single sector, called the *basic sector*. A proper basic sector represents one part of a pattern that, if repeated *N* times in cylindrical coordinate space, yields the complete model, as shown:

Figure 6.2 A Basic Sector in a Cyclically Symmetric Structure

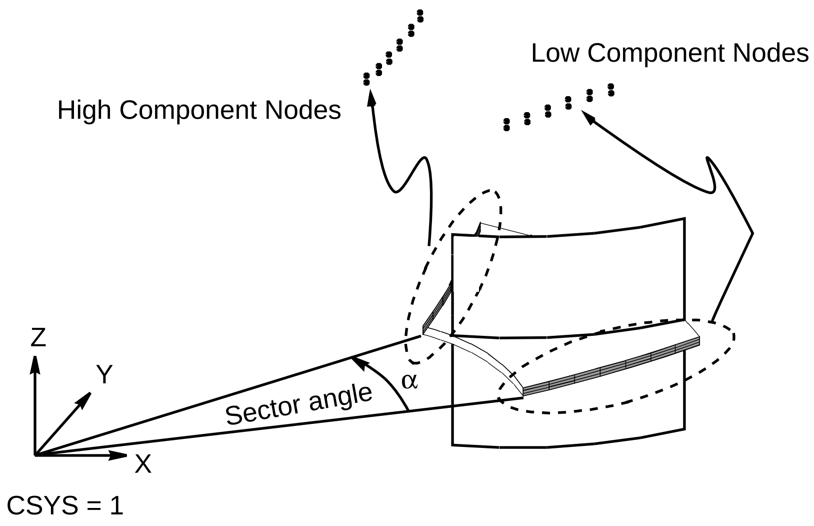


Define a basic sector model that is cyclically symmetric in any global or user-defined cylindrical coordinate system. (For information about creating a model, see the *ANSYS Modeling and Meshing Guide*.)

The angle θ (in degrees) spanned by the basic sector should be such that $N\theta = 360$, where *N* is an integer. The basic sector can consist of meshed or unmeshed geometry. ANSYS allows user-defined coupling and constraint equations on nodes that are not on the low or high edges of the cyclic sector. (For more information about the cyclic sector's low and high edges, see Section 6.2.2: Edge Component Pairs.)

If meshed, the basic sector may have matching (as shown in Figure 6.3: “Basic Sector Definition”) or unmatched lower and higher angle edges. *Matching* means that corresponding nodes exist on each edge, offset geometrically by the sector angle θ. The edges may be of any shape and need not be "flat" in cylindrical coordinate space. For more information, see Section 6.2.2.1: Identical vs. Dissimilar Edge Node Patterns.

Figure 6.3 Basic Sector Definition



6.2.2. Edge Component Pairs

The cyclic sector has two edges that align along the surfaces of cyclic symmetry. The edge having the algebraically lower θ in the R-Theta (cylindrical) coordinate system is called the *low edge* and the one having the higher θ is called the *high edge*. The angle α between the two successive surfaces of cyclic symmetry is called the *sector angle*.

When setting up a cyclic symmetry analysis, the **CYCLIC** command defines edge components automatically, assigning them a default root name of "CYCLIC."

Optionally, you can use the **CYCLIC** command to define the edges and the component names manually. If you do so, you must specify a root name for the sector low- and high-edge components (line, area, or node components). A root name that you specify can contain up to 11 characters. The naming convention for each low- and high-edge component pair is either of the following:

*name*_m*xx*l, *name*_m*xx*h
(potentially matched node patterns)

*name*_u*xx*l, *name*_u*xx*h
(potentially unmatched node patterns)

The *name* value is the default ("CYCLIC") or specified root name, and *xx* is the component pair ID number (sequential, starting at 01).

6.2.2.1. Identical vs. Dissimilar Edge Node Patterns

Automated Matching

The **AMESH** and **VMESH** commands perform automated matching. Other meshing-operation commands (for example, **VSWEEP**) do not.

If you employ a meshing operation other than **AMESH** or **VMESH**, ensure that node and element face patterns match, if desired. The **CYCLIC** command output indicates whether each edge-component pair has or can produce a matching node pair.

To ensure the most accurate solution, it is preferable to have identical node and element face patterns on the low and high edges of the cyclic sector. If you issue the **CYCLIC** command *before* meshing the cyclic sector (via the **AMESH** or **VMESH** command only), the mesh will, if possible, have identical node and element face patterns on the low and high edges.

ANSYS allows dissimilar node patterns on the low and high edges of the cyclic sector, useful when you have only finite-element meshes for your model but not the geometry data necessary to remesh it to obtain identical node patterns. In such cases, it is possible to obtain solution (**SOLVE**) results, although perhaps at the expense of accuracy. A warning message appears because results may be degraded near the cyclic sector edges.

6.2.2.2. Unmatched Nodes on Edge-Component Pairs

Unmatched nodes on the low- and high-edge components produce *approximate* cyclic symmetry solutions (as compared to matched-node cases). ANSYS employs an unmatched-node algorithm (similar to that of the **CEINTF** command) to connect dissimilar meshes.

In unmatched cases, the eigenvector shapes exhibit discontinuity across segment boundaries when expanded (via the **/CYCEXPAND** command). The discontinuity is an expected behavior; in the expansion process, the low edge of sector 2 lies adjacent to the high edge of sector 1, and so on throughout the full 360°.

For information about expanding the solution results of a cyclic symmetry analysis, see Section 6.4.2: Expanding the Cyclic Symmetry Solution.

6.2.2.3. Identifying Matching Node Pairs

To identify matching node pairs, you can issue a ***STATUS** command to list the cyclic parameter array ***Name*_xref_*n*** (where *Name* is the root name of the low- and high-edge components specified via the **CYCLIC** command).

The cyclic parameter array is generated internally during element plotting with cyclic expansion activated (**/CYCEXPAND**,,ON).

In the cyclic parameter array listing, the matching node pairs appear as a pair of node numbers with the low-edge node number having a negative value.

6.2.3. Model Verification (Preprocessing)

If the **CYCLIC** command's default automatic detection capability accepts your model for cyclic analysis, ANSYS will have already verified the following two essential conditions for a cyclic analysis:

- When your model rotates by the cyclic angle about the local Z axis of the cyclic coordinate system, the edges identified as "low" occupy the same space as those identified by "high" prior to the rotation.
- The cyclic angle divides evenly into 360°.

If you specify edge components and cyclic quantities manually, you must verify the two conditions yourself.

6.3. Solving a Cyclic Symmetry Analysis

ANSYS solves for the full cyclically symmetric model using the basic sector model that you have set up during preprocessing with the appropriate boundary conditions, loading, and any coupling and constraint equations. (For more information, see Section 6.2: Cyclic Modeling.) This section provides specific information for obtaining the solution to various types of cyclic symmetry analyses and covers the following topics:

- Section 6.3.1: Understanding the Solution Architecture

6.3.1. Understanding the Solution Architecture

At the solution (**SOLVE**) stage of a cyclic symmetry analysis, ANSYS applies the appropriate cyclic symmetry boundary conditions for each nodal-diameter solution requested (via the **CYCOPT** command) and solves. ANSYS performs each nodal-diameter solution as a separate load step.

6.3.1.1. The Duplicate Sector

The architecture of the cyclic symmetry solution process depends upon how the compatibility and equilibrium conditions of the cyclic sector are enforced in the matrix-solution process. The two most commonly employed solution methods are Duplicate Sector and Complex Hermitian. For faster performance, ANSYS employs the Duplicate Sector method.

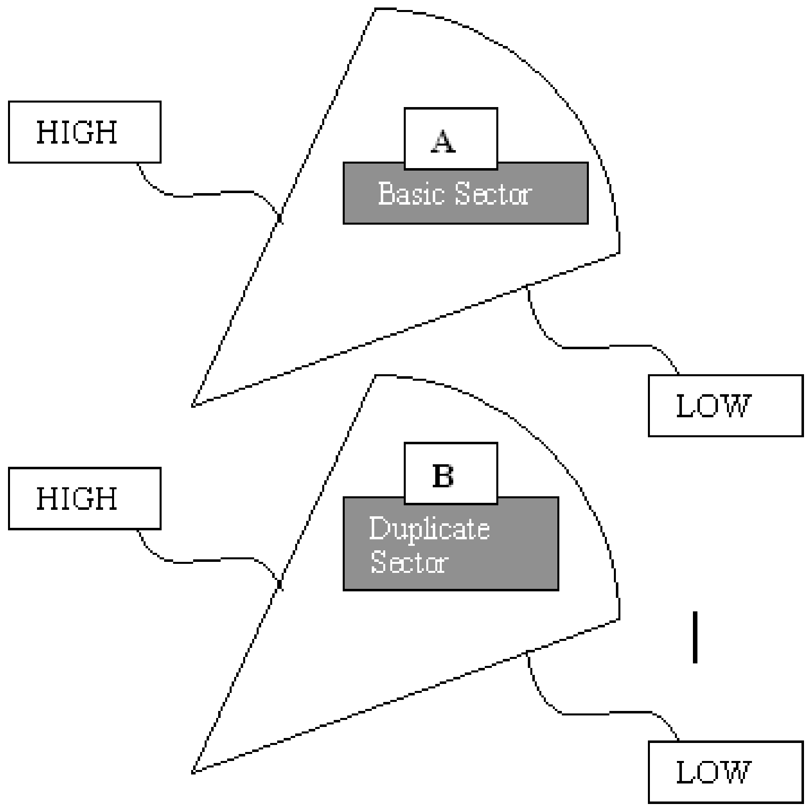
During the solution stage, ANSYS generates a duplicate sector of elements at the same geometric location as the basic sector. (Duplicate sector creation occurs automatically and transparently.) ANSYS applies all loading, boundary conditions, and coupling and constraint equations present on the basic sector to the duplicate sector.

6.3.1.2. Coupling and Constraint Equations (CEs)

ANSYS enforces cyclic symmetry compatibility conditions for each nodal-diameter solution via coupling and/or constraint equations (CEs) connecting the nodes on the low- and high-edge components on the basic and duplicate sectors. ANSYS deletes the coupling and/or constraint equations after each nodal-diameter solution, preserving any internal coupling and constraint equations that you may have defined on the basic sector. The constraint equations for edge-component nodes have the form shown in Equation 6–1.

Note — During the solution stage of a cyclic symmetry analysis, ANSYS automatically rotates the nodal coordinate systems of all nodes on the low and high sector edges to be parallel with the cyclic coordinate system.

Figure 6.4 Connecting Low and High Edges of Basic and Duplicate Sectors



$$\begin{Bmatrix} U^A High \\ U^B High \end{Bmatrix} = \begin{bmatrix} \cos k\alpha & \sin k\alpha \\ -\sin k\alpha & \cos k\alpha \end{bmatrix} \begin{Bmatrix} U^A Low \\ U^B Low \end{Bmatrix} \quad \textbf{(6–1)}$$

where,

k = Harmonic index -- (0,1,2,…,N / 2) when N is even, (0,1,2,…,(N-1) / 2) when N is odd. (N is an integer representing the number of sectors in 360°.)
α = Sector angle (2π / N)
U = Vector of displacement and rotational degrees of freedom
U^A*Low* represents the basic sector low side edge
U^A*High* represents the basic sector high side edge
U^B*Low* represents the duplicate sector low side edge
U^B*High* represents the duplicate sector high side edge

The equation is a function of harmonic index *k* generating different sets of constraint equations for each harmonic index. Therefore, for each harmonic index solution requested, ANSYS creates the appropriate constraint equations automatically, connects the edge-component nodes on basic sector A and duplicate sector B, and solves.

Constraint equations that tie together the low and high edges of your model are generated from the low- and high-edge components, and nowhere else. You should verify that automatically detected components are in the correct locations and that you are able to account for all components; to do so, you can list (**CMLIST**) or plot (**CMPLOT**) the components.

6.3.1.3. Non-Cyclically Symmetric Loading

A load is non-cyclic when it varies between sectors and involves at least one harmonic index greater than zero. ANSYS supports linear static cyclic symmetry analyses with non-cyclic loading. Support is also available for a cyclic analysis having some *combination* of cyclic and non-cyclic loading.

For non-cyclic loading, ANSYS considers the arbitrary forces acting on the full system as the sum of a finite number of spatial Fourier harmonics. The program analyzes the structure for each spatial harmonic index by applying constraint equations between the basic sector and duplicate sector. For each spatial harmonic of force, the program solves a corresponding equation, then expands and sums the calculated harmonics of the response to give the response for each substructure. For more information, see Section 15.15.3: Cyclic Symmetry Transformations in the *ANSYS, Inc. Theory Reference*.

Table 6.1 Valid Non-Cyclically Symmetric Loads

Non-Cyclic Load Type	Commands	Constraints	Comments
Nodal DOF Constraints	**D**, **DA**, **DK**, **DL**	UX, UY, UZ, ROTX, ROTY, and ROTZ constraints follow sector-restricted loading (**CYCOPT**,LDESCT,*n* where *n* > 0).	Always in *sector* coordinates. Same as nodal coordinates for sector 1, rotated by sector angle * (*N* - 1) for sector *N*. Constraint always acts on given DOF for all sectors.
Nodal Loads	**F**, **K**	FX, FY, and FZ constraints follow sector-restricted loading (**CYCOPT**,LDESCT,*n* where *n* > 0).	HFLOW is blocked for sector-restricted loading but can follow afterwards. All other nodal loads are blocked for sector-restriced loading.
Surface Loads	**SF**, **SFA**, **SFE**, **SFL**	PRES follows sector-restricted loading (**CYCOPT**,LDESCT,*n* where *n* > 0).	CONV is blocked for sector-restriced loading but can follow afterwards. All other surface loads are blocked for sector-restriced loading.
Inertia Loads	**ACEL**, **DOMEGA**, **CMDOMEGA**, **CMOMEGA**, **OMEGA**	Apply to all sectors. (Not affected by sector-restriction via **CYCOPT**,LDSECT,*n* where *n* > 0). Default action in global X, Y, and Z on all sectors.	May require harmonic index 0 and/or 1 only.

6.3.1.3.1. Specifying Non-Cyclic Loading

Specify non-cyclically symmetric loading via the the **CYCOPT** command' s LDSECT (load-on-sector) value. A value greater than 0 (the default, indicating that the loads are identical on all sectors) restricts subsequently defined DOF constraint values, force loads, and surface loads to the specified sector. The restriction remains in effect until you change or reset it.

When non-cyclic loading applies, ANSYS creates or modifies the required SECTOR tabular boundary condition (BC) data to apply on the appropriate sector. Therefore, it is not necessary to manipulate tables for situations where the applied BC is not a function of other tabular BC variables such as TIME, X, Y, Z, and so on.

If a SECTOR-varying table exists on an entity-BC combination (for example, node 17 FZ) and you enter another value for the same entity-BC combination (perhaps specifying a different sector on which to apply the load), the following conditions occur:

- ANSYS modifies the existing table to accommodate the new specification.
- The table cannot reference any *other* independent variable (for example, TEMP). You must manually define any BC table requiring more than one independent variable.

If a table exists for an entity-BC combination and you enter another table for the same entity-BC combination, but the table does not reference SECTOR, the new table reference *replaces* the existing one.

During preprocessing, all tabular BCs referencing SECTOR list table names only. During solution or postprocessing, all tabular BCs referencing SECTOR list per sector as they would be applied when solving (**SOLVE**).

Any tabular data X, Y, or Z variation applied to a cyclic model may not be applied in the same manner in which such a variation would occur for an equivalent full model (the exception being a variation in the axial direction). For example, if a tabular value of a nodal force is applied as function of the tabular variable Y, ANSYS applies it to the designated cyclic sectors using values based upon the Y values of the basic sector only.

A given high-edge node is usually the same location in the structure as the corresponding low-edge node of the adjacent sector; therefore, it is necessary to apply constraints consistently. ANSYS can trap inconsistent constraints only for matched-node cases; however, for tabular BCs, you must perform the consistency verification yourself. Because inconsistent constraints are impossible to satisfy if the solution remains cyclic, the results can be unpredictable.

Because edge nodes are rotated into the cyclic coordinate system during solution, any applied displacements or forces on sector edges will be in the cyclic coordinate system.

Example 6.1 Non-Cyclic Loading via Automatically-Defined Tabular Load

```
CYCOPT,LDSECT,1 ! LOADS ON SECTOR 1 ONLY

SFL,ALL,PRES,10000
```

For DOF constraints, force loads, and body forces, any non-tabular load is cyclic. Any tabular load that does not reference the variable SECTOR is cyclic. ANSYS assumes any tabular load referencing SECTOR to be non-cyclic (although it *could* be identical on all sectors).

Example 6.2 Non-Cyclic Loading via User-Defined Tabular Load

```
*DIM,S1PRES,TABLE,5,1,1,SECTOR

*SET,S1PRES(1,0,1),1,2,3,4,5

*SET,S1PRES(1,1,1),10000,0,0,0,0  ! PRESSURE ON SECTOR 1 ONLY

SFL,ALL,PRES,%S1PRES%
```

When combined with other independent variables, SECTOR can be in positions 1, 2, or 3 only. Other independent variables operate as they do for non-cyclic data. (Think of X, Y, and Z as “ghost” coordinates, behaving as though all sectors have been modeled with actual nodes and elements.)

Example 6.3 Deleting a Sector Load

```
CYCOPT,LDSECT,3

F,10,UX,value   ! Apply a load (value) on node 10 at sector 3

...

FDELE,10,UX    ! Delete the load on node 10 at sector 3
```

To delete a previously applied load on a specified sector, issue an **FDELE** command.

6.3.1.3.2. Commands Affected by Non-Cyclic Loading

When non-cyclic loading applies, the following ANSYS commands are affected: **D**, **DA**, **DK**, **DL**, **FK**, **SF**, **SFA**, **SFE**, **SFL**.

The commands do not allow a table name in any *value* position. You must manually define any boundary condition or loading table requiring more than one independent variable (for example, SECTOR and some other variable).

The **SF** and **SFE** commands do not allow nodal pressure values to vary over an element face.

6.3.1.3.3. Plotting and Listing Non-Cyclic Boundary Conditions

You can plot non-cyclic boundary conditions (BCs) on the sector on which the BC (**F**, **D**, **SF**) is applied. By expanding the cyclic sector model plot to the full 360 degrees (via the **/CYCEXPAND** command), you can view a BC on the sector on which it is applied.

Issue BC-listing commands **FLIST**, **DLIST**, and **SFLIST** to list non-cyclic BCs. The list indicates the value of the BC and the sector on which it is applied.

6.3.2. Supported Analysis Types

ANSYS supports static, modal and buckling analysis types for a cyclic symmetry solution, as follows:

- **Static analysis**

  For cyclically symmetric loading, support is available for linear static and large-deflection nonlinear static solution options. For non-cyclically symmetric loading, ANSYS supports linear static analysis only. For more information, see Section 6.3.3: Solving a Static Cyclic Symmetry Analysis.

  ANSYS also supports static analyses involving magnetic cyclic symmetry. For more information, see Section 6.3.6: Solving a Magnetic Cyclic Symmetry Analysis.

- **Modal analysis**

  ANSYS supports modal analyses with or without prestress effects. In the case of a prestressed modal analysis, the prestress state of the sector may be from a linear static or large-deflection nonlinear static analysis. For more information, see Section 6.3.4.2: Solving a Stress-Free Modal Analysis and Section 6.3.4.3: Solving a Prestressed Modal Analysis.

  For a large-deflection prestressed modal cyclic symmetry analysis, partial solution steps employing the **PSOLVE** command are necessary; however, no separate expansion (via **PSOLVE**,EIGEXP) is needed because ANSYS expands the eigenvectors out to the results file automatically. Alternatively, you can simply issue the **SOLVE** command; in this case, ANSYS performs the partial solution step (via **PSOLVE**,EIGLANB) *automatically* and expands the eigenvectors out to the results file. For more information, see Section 6.3.4.4: Solving a Large-Deflection Prestressed Modal Analysis.

- **Linear buckling analysis**

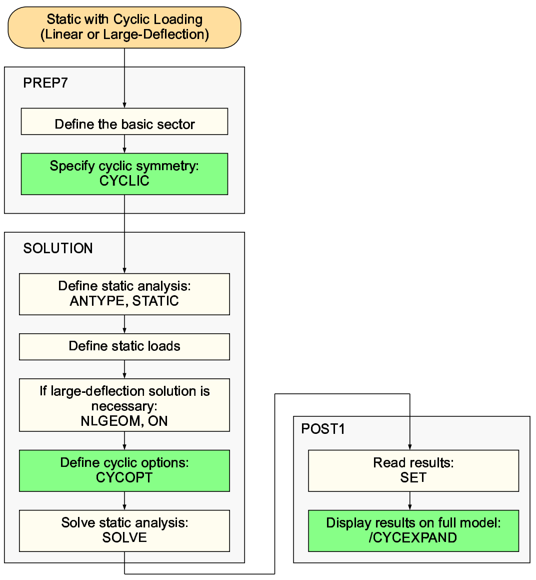
  Support is available for linear eigenvalue buckling analyses with prestress effects. ANSYS recommends the Block Lanczos mode-extraction method (via **BUCOPT**,LANB). You can also employ the subspace option (via **BUCOPT**,SUBS); however, extracting negative buckling loads may be more difficult. For more information, see Section 6.3.5: Solving a Linear Buckling Cyclic Symmetry Analysis, and Buckling Analysis in the *ANSYS Structural Analysis Guide*.

6.3.3. Solving a Static Cyclic Symmetry Analysis

For cyclically symmetric loading, support is available for linear static and large-deflection nonlinear static solution options. (Cyclically symmetric loading implies any load applied on the cyclic sector representing a loading pattern that is repetitive at sector angle increments around the 360° structure.)

The following flowchart illustrates the process involved in a static (linear or large-deflection) cyclic symmetry analysis with cyclic loading.

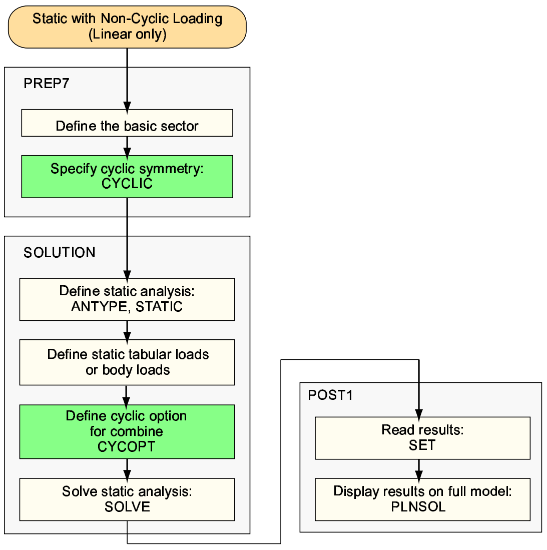
Figure 6.5 Process Flow for a Static Cyclic Symmetry Analysis (Cyclic Loading)


Static with Cyclic Loading (Linear or Large-Deflection)
PREP7
Define the basic sector
Specify cyclic symmetry: CYCLIC
SOLUTION
Define static analysis: ANTYPE, STATIC
Define static loads
If large-deflection solution is necessary: NLGEOM, ON
Define cyclic options: CYCOPT
Solve static analysis: SOLVE
POST1
Read results: SET
Display results on full model: /CYCEXPAND

Only a nodal-diameter zero solution is valid for a static solution with cyclic loading.

For non-cyclically symmetric loading, ANSYS supports linear static analysis only. The following flowchart illustrates the process involved in a static cyclic analysis with non-cyclic loading.

Figure 6.6 Process Flow for a Static Cyclic Symmetry Analysis (Non-Cyclic Loading)


Static with Non-Cyclic Loading
(Linear only)
PREP7
Define the basic sector
Specify cyclic symmetry: CYCLIC
SOLUTION
Define static analysis: ANTYPE, STATIC
Define static tabular loads or body loads
Define cyclic option for combine CYCOPT
Solve static analysis: SOLVE
POST1
Read results: SET
Display results on full model: PLNSOL

For more information, see Section 6.3.1.3: Non-Cyclically Symmetric Loading.

6.3.4. Solving a Modal Cyclic Symmetry Analysis

This section describes harmonic indices in relation to modal cyclic symmetry analyses and provides information necessary for solving several types of modal analyses. The following pages cover these topics:

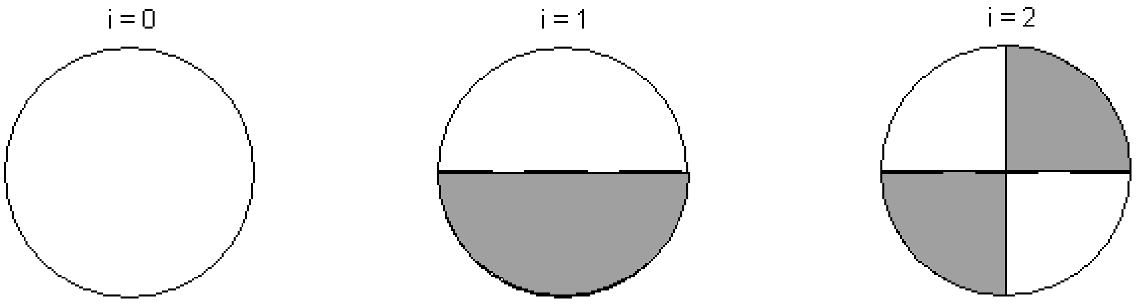
- Section 6.3.4.1: Understanding Harmonic Index and Nodal Diameter
- Section 6.3.4.2: Solving a Stress-Free Modal Analysis
- Section 6.3.4.3: Solving a Prestressed Modal Analysis
- Section 6.3.4.4: Solving a Large-Deflection Prestressed Modal Analysis

6.3.4.1. Understanding Harmonic Index and Nodal Diameter

To understand the process involved in a modal cyclic symmetry analysis, it is necessary to understand the concepts of harmonic indices and nodal diameters.

The *nodal diameter* refers to the appearance of a simple geometry (for example, a disk) vibrating in a certain mode. Most mode shapes contain lines of zero out-of-plane displacement which cross the entire disk, as shown in these examples:

Figure 6.7 Examples of Nodal Diameters (i)



For a complicated structure exhibiting cyclic symmetry (for example, a turbine wheel), lines of zero displacement may not be observable in a mode shape.

The *harmonic index* is an integer that determines the variation in the value of a single DOF at points spaced at a circumferential angle equal to the sector angle. For a harmonic index equal to nodal diameter *d*, the function cos(*d**θ) describes the variation. This definition allows a varying number of waves to exist around the circumference for a given harmonic index, provided that the DOF at points separated by the sector angle vary by cos(*d**θ). For example, a harmonic index of 0 and a 60° sector produce modes with 0, 6, 12, ... , 6*N* waves around the circumference.

The nodal diameter is the same as the harmonic index in only some cases. The solution of a given harmonic index may contain modes of more than one nodal diameter.

The following equation represents the relationship between the harmonic index *k* and nodal diameter *d* for a model consisting of *N* sectors:

$$d = m \cdot N \pm k \tag{6–2}$$

where $m = 0, 1, 2, 3, ..., \infty$

For example, if a model has seven sectors (*N* = 7) and the specified harmonic index *k* = 2, ANSYS solves for nodal diameters 2, 5, 9, 12, 16, 19, 23, ...

The following table illustrates Equation 6–2, showing how the harmonic index, nodal diameter and number of sectors relate to one another:

Harmonic Index (k)	**Nodal Diameter (d)**					
0	0	*N*	*N*	2*N*	2*N*	...
1	1	*N*-1	*N* + 1	2*N* - 1	2*N* + 1	...
2	2	*N* - 2	*N* + 2	2*N* - 2	2*N* + 2	...
3	3	*N* - 3	*N* + 3	2*N* - 3	2*N* + 3	...
4	4	*N* - 4	*N* + 4	2*N* - 4	2*N* + 4	...
...	...	...	...	...	...	...
N / 2 (*N* is even)	*N* / 2	*N* / 2	3*N* / 2	3*N* / 2	5*N* / 2	...
(*N* - 1) / 2 (*N* is odd)	(*N* - 1) / 2	(*N* + 1) / 2	(3*N* - 1) / 2	(3*N* + 1) / 2	(5*N* - 1) / 2	...

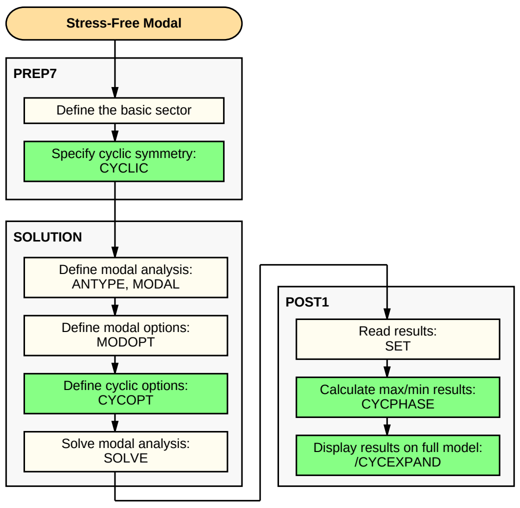
Note — To avoid confusion, be aware that in some references *mode* refers to harmonic index as defined here and *nodal diameter* describes the actual number of observable waves around the structure.

Harmonic Index in an Electromagnetic Analysis For electromagnetic analyses, only the EVEN and ODD harmonic index settings are valid (for symmetric and antisymmetric solutions, respectively).

6.3.4.2. Solving a Stress-Free Modal Analysis

The following flowchart illustrates the process involved in a stress-free modal cyclic symmetry analysis.

Figure 6.8 Process Flow for a Stress-Free Modal Cyclic Symmetry Analysis


Stress-Free Modal
PREP7
Define the basic sector
Specify cyclic symmetry: CYCLIC
SOLUTION
Define modal analysis: ANTYPE, MODAL
Define modal options: MODOPT
Define cyclic options: CYCOPT
Solve modal analysis: SOLVE
POST1
Read results: SET
Calculate max/min results: CYCPHASE
Display results on full model: /CYCEXPAND

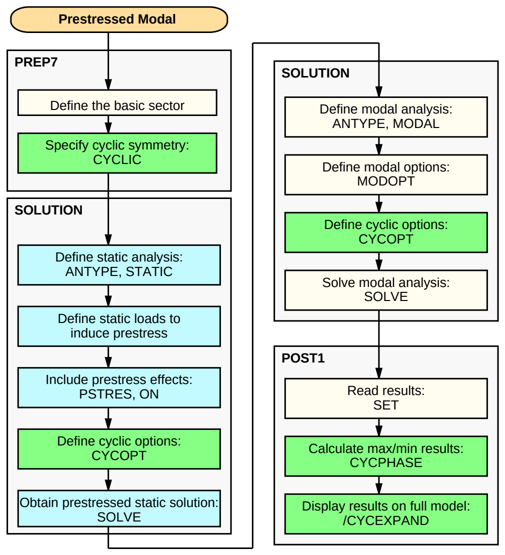
A modal cyclic symmetry analysis allows only cyclically symmetric applied boundary constraints and applied loads. Applied loads are not used in a modal analysis (which is a free-vibration problem). Eigensolutions are performed, looping on the number of harmonic indices specified (via the **CYCOPT** command) at each load step.

6.3.4.3. Solving a Prestressed Modal Analysis

The process for a prestressed modal cyclic symmetry analysis is essentially the same as that for a stress-free case, except that a static solution is necessary to calculate the prestress in the basic sector. The prestress state of the sector may be from a linear static or a large-deflection nonlinear static analysis. If the model is spinning, you can include spin-softening effects (via the **OMEGA** command's *KSPIN* option) in the modal solution if necessary.

The following flowchart illustrates the process involved in a prestressed modal cyclic symmetry analysis.

Figure 6.9 Process Flow for a Prestressed Modal Cyclic Symmetry Analysis


Prestressed Modal
PREP7
Define the basic sector
Specify cyclic symmetry: CYCLIC
SOLUTION
Define static analysis: ANTYPE, STATIC
Define static loads to induce prestress
Include prestress effects: PSTRES, ON
Define cyclic options: CYCOPT
Obtain prestressed static solution: SOLVE
SOLUTION
Define modal analysis: ANTYPE, MODAL
Define modal options: MODOPT
Define cyclic options: CYCOPT
Solve modal analysis: SOLVE
POST1
Read results: SET
Calculate max/min results: CYCPHASE
Display results on full model: /CYCEXPAND

The modal cyclic symmetry solution occurs *after* the static cyclic symmetry solution. The modal solution uses the same low- and high-edge components defined in the static cyclic analysis stage (via the **CYCLIC** command). The analysis yields the eigenvectors of the structure in the prestressed state.

6.3.4.4. Solving a Large-Deflection Prestressed Modal Analysis

Geometric nonlinearity occurs when the deflections are large enough to cause significant changes in the geometry of the structure. In such cases, the equations of equilibrium must account for the deformed configuration. (For more information, see the *ANSYS, Inc. Theory Reference*.) When a nonlinearity is present, ANSYS employs an iterative process to obtain the solution.

To calculate the frequencies and mode shapes of a highly deformed structure, you can perform a prestressed modal analysis of cyclic structures *after first performing* a large-deflection (**NLGEOM**,ON) static analysis. In the cyclic modal analysis that follows the large-deflection static analysis, the **PSTRES**,ON command applies the prestress effects. If the model is spinning, you can include spin-softening effects (via the **OMEGA** command's `KSPIN` option) in the modal solution if necessary.

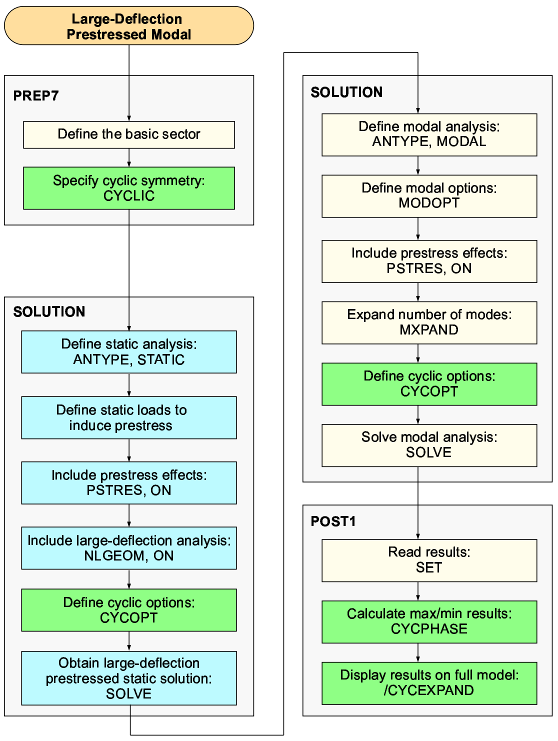
To obtain the modal solution, two options are available:

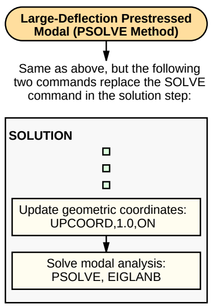
- Issue the **SOLVE** command to update the geometry and solve.

  In this case, ANSYS obtains the solution *automatically* (via calls to the **PSOLVE**,EIGLANB and **UPCOORD**,1,ON commands).
- Update the geometry (via the **UPCOORD** command) and issue the **PSOLVE**,EIGLANB command.

The following flowchart illustrates both methods for solving a large-deflection prestressed modal cyclic symmetry analysis.

Figure 6.10 Process Flow for a Large-Deflection Prestressed Modal Cyclic Symmetry Analysis


Large-Deflection Prestressed Modal
PREP7
Define the basic sector
Specify cyclic symmetry: CYCLIC
SOLUTION
Define static analysis: ANTYPE, STATIC
Define static loads to induce prestress
Include prestress effects: PSTRES, ON
Include large-deflection analysis: NLGEOM, ON
Define cyclic options: CYCOPT
Obtain large-deflection prestressed static solution: SOLVE
SOLUTION
Define modal analysis: ANTYPE, MODAL
Define modal options: MODOPT
Include prestress effects: PSTRES, ON
Expand number of modes: MXPAND
Define cyclic options: CYCOPT
Solve modal analysis: SOLVE
POST1
Read results: SET
Calculate max/min results: CYCPHASE
Display results on full model: /CYCEXPAND


Large-Deflection Prestressed Modal (PSOLVE Method)
Same as above, but the following two commands replace the SOLVE command in the solution step:
SOLUTION
Update geometric coordinates: UPCOORD,1.0,ON
Solve modal analysis: PSOLVE, EIGLANB

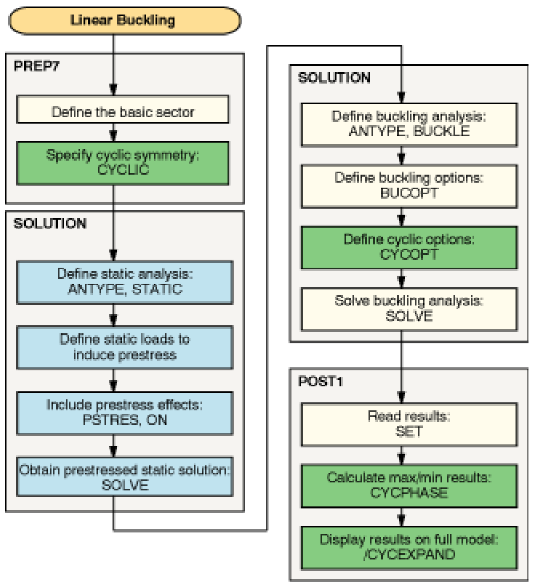
Solving a large-deflection prestressed modal cyclic symmetry analysis is similar to solving a prestressed modal cyclic analysis in that the modal solution requires a linear static cyclic solution. The differences are as follows:

- The large-deflection key (via the **NLGEOM** command) in the static cyclic analysis stage
- The partial solutions logic (via the **PSOLVE** command) in the modal cyclic analysis stage.

6.3.5. Solving a Linear Buckling Cyclic Symmetry Analysis

The process for a linear buckling analysis is essentially the same as that for a prestressed modal cyclic symmetry analysis, with the exception that buckling options (**ANTYPE**,BUCKLE and **BUCOPT**,LANB) are necessary to calculate buckling loads and the corresponding buckled mode shapes. The following flowchart illustrates the process involved in an eigenvalue buckling cyclic symmetry analysis.

Figure 6.11 Process Flow for a Linear Buckling Cyclic Symmetry Analysis


Linear Buckling
PREP7
Define the basic sector
Specify cyclic symmetry: CYCLIC
SOLUTION
Define static analysis: ANTYPE, STATIC
Define static loads to induce prestress
Include prestress effects: PSTRES, ON
Obtain prestressed static solution: SOLVE
SOLUTION
Define buckling analysis: ANTYPE, BUCKLE
Define buckling options: BUCOPT
Define cyclic options: CYCOPT
Solve buckling analysis: SOLVE
POST1
Read results: SET
Calculate max/min results: CYCPHASE
Display results on full model: /CYCEXPAND

You can specify the **SET** command's ORDER option to sort the harmonic index results in ascending order of buckling load multipliers.

6.3.6. Solving a Magnetic Cyclic Symmetry Analysis

Most magnetic analysis problems can be defined with flux parallel and/or flux normal boundary conditions. With problems such as electrical machines, however, cyclic boundary conditions best represent the periodic nature of the structure and excitation, and have the advantage of being able to use a less computation-intensive partial model, rather than a full model.

You can analyze only one sector of the full model to take advantage of this kind of symmetry. The full model consists of as many sectors as the number of poles. In Section 6.7: Sample Magnetic Cyclic Symmetry Analysis, the number of sectors is two; the analysis can be done on a half model.

The cyclic boundary condition is between matching degrees of freedom on corresponding symmetry faces. The studied sector is bounded by two faces called the low edge and high edge, respectively. In Figure 6.18: "Two-Phase Electric Machine – Half Model", the low edge face is the y = 0, x >= 0 plane; the high edge is the y = 0, x <= 0 plane.

The simplest case is when the node-matching interface of the low edge is the same as the high edge. In this case, for every node, there is one and only one matching node on the high edge; moreover the pertinent geometry and connectivity are the same. In this case, the cyclic boundary condition for the edge formulation could be formulated as

Az(low entity) = - Az(high entity)

This is an anti-symmetric condition which is called ODD symmetry.

The analysis could be carried out on a 360/p sector (where p is the number of poles), in which case the cyclic condition would be:

Az(low entity) = + Az(high entity)

which is a symmetric condition called EVEN symmetry.

In Section 6.7: Sample Magnetic Cyclic Symmetry Analysis, the ODD model is smaller and thus more practical. For some problems, depending on geometry and excitation, EVEN symmetry may be more practical. ANSYS supports both ODD and EVEN cyclic symmetry.

In a more general case, the mesh on the low and high end may be different. In this case, more general cyclic symmetry conditions can be established by interpolation on the pertinent faces. ANSYS handles this process automatically via the **CYCLIC** command.

ANSYS requires node-matching only for SOLID117 magnetic edge elements; however, you will typically obtain greater accuracy using node-matching for any element type.

The geometry of the low- and high-end cyclic faces may be more general than a simple plane surface. Thus, for example, a skewed slot of an electric machine may constitute the cyclic sector modeled.

Section 6.2: Cyclic Modeling discusses cyclic modeling in detail.

The following restrictions apply to a magnetic cyclic symmetry analysis:

- For SOLID117 elements, the low and high edges must have node-matching conditions.
- Cyclic conditions can be restricted to specific degrees of freedom (DOFs) via the DOF option. DOF restrictions may be useful, for example, in cases involving circuit-/voltage-fed solenoidal edge elements (SOLID117 with KEYOPT(1) = 5 or 6).
- Multiphysics coupling must use the same EVEN/ODD condition.
- Circuit coupling is not supported for cyclic symmetry except solenoidal SOLID97 and SOLID117.
- Harmonic and transient analyses are not supported.
- By default, plotting displays the partial solution only. To see the full model solution, issue the **/CYCEXPAND** command. For magnetic cyclic symmetry, the **/CYCEXPAND** command produces contour plots but not vector plots.

Figure 6.5: "Process Flow for a Static Cyclic Symmetry Analysis (Cyclic Loading)" shows the process for a cyclic symmetry analysis. The process is virtually identical for a magnetic cyclic symmetry analysis; simply disregard the step for a large-deflection solution.

Magnetic cyclic boundary conditions can be applied to the following element types:

- Nodal magnetic vector potential elements: PLANE13, PLANE53, SOLID97
- Magnetic edge elements, classical or solenoidal: SOLID117
- Magnetic scalar potential elements: SOLID5, SOLID96, SOLID98
- Electrostatic elements: PLANE121, SOLID122, SOLID123

6.3.7. Database Considerations After Obtaining the Solution

At the conclusion of the cyclic symmetry solution, exit the solution processor via the **FINISH** command.

If you intend to exit the ANSYS program at this point (before postprocessing), save the database (**file.db**). The saved database allows you to perform postprocessing on the analysis results at a later time.

6.3.8. Model Verification (Solution)

The cyclic solution reports the number of constraint equations generated for each harmonic index solution and information about how they were created. The information should match what you already know about the analysis model; if not, try to determine the reason for the discrepancy. The following extracts (from a batch output file or an interactive output window) are typical:

```
NUMBER OF CONSTRAINT EQUATIONS GENERATED =   124
(USING THE MATCHED NODES ALGORITHM --
                  MAX NODE LOCATION ERROR NEAR ZERO)
```

Meaning: 124 constraint equations are created, used, and then deleted to enforce cyclic symmetry conditions between the low- and high-edge nodes. Every node on the low edge is precisely matched to a corresponding node on the high edge, representing the best possible situation.

```
NUMBER OF CONSTRAINT EQUATIONS GENERATED =   124
(USING THE MATCHED NODES ALGORITHM --
                  MAX NODE LOCATION ERROR = 0.73906E-02)
```

Meaning: 124 constraint equations are created, used, and then deleted to enforce cyclic symmetry conditions between the low- and high-edge nodes. Every node on the low edge is matched to a corresponding node on the high edge within the current tolerance setting, but not all matches are precise. The largest position mismatch is 0.0073906.

```
NUMBER OF CONSTRAINT EQUATIONS GENERATED =   504
(USING THE UNMATCHED NODES ALGORITHM)
```

Meaning: 504 constraint equations are created, used, and then deleted to enforce cyclic symmetry conditions between the low- and high-edge nodes. At least one node on the low edge does not match any node on the high edge within the current tolerance setting, so ANSYS uses the unmatched nodes algorithm.

6.4. Postprocessing a Cyclic Symmetry Analysis

This section describes how to perform postprocessing on the solution results obtained from a cyclic symmetry analysis. The following pages cover these topics:

- Section 6.4.1: Real and Imaginary Solution Components

- Section 6.4.2: Expanding the Cyclic Symmetry Solution
- Section 6.4.3: Phase Sweep of Repeated Eigenvector Shapes

If you exited the ANSYS program after obtaining the cyclic symmetry solution, use the database (**file.db**) that you saved for postprocessing. For more information, see Section 6.3.7: Database Considerations After Obtaining the Solution.

6.4.1. Real and Imaginary Solution Components

A cyclic symmetry solution typically has multiple load step results depending upon the harmonic index solutions requested.

The real (basic sector) and imaginary (duplicate sector) parts of the solution reside in the results file. However, the solution does not yet represent the actual displacements, stresses, or reaction forces for *any part* of the actual structure.

> *Note* — Listing or plotting the sector results causes ANSYS to issue a warning message such as *PLNSOL is displaying the unprocessed real and imaginary parts of this cyclic symmetry solution*. Furthermore, the basic and duplicate sectors overplot each other if displayed, providing yet another indication that a problem exists.

To transform the real and imaginary cyclic symmetry solution results to the actual structure solution, three postprocessing (**/POST1**) commands are available:

- **/CYCEXPAND**
- **EXPAND**
- **CYCPHASE**

For information about **/CYCEXPAND** and **EXPAND** command usage, see Section 6.4.2: Expanding the Cyclic Symmetry Solution. For information about **CYCPHASE** command usage, see Section 6.4.3: Phase Sweep of Repeated Eigenvector Shapes.

6.4.2. Expanding the Cyclic Symmetry Solution

This section describes the capabilities of the **/CYCEXPAND** and **EXPAND** commands and explains their differences. Use the commands to expand the solution results of your cyclic symmetry analysis to the full model.

6.4.2.1. Using the /CYCEXPAND Command

The **/CYCEXPAND** command is the preferred operation for viewing expanded cyclic symmetry results. The command does not modify the geometry, nodal displacements or element stresses stored in the database. Issue the command to expand your basic sector model and obtain the full 360° model displacement, stress, or strain response.

After the expansion, you can plot (**PLESOL** or **PLNSOL**) or print (**PRNSOL**) the results. Other commands (such as **NSEL** and **NSORT**) continue to operate on the *unprocessed* real and imaginary parts of the solution.

Using the cyclic symmetry solution of the basic and duplicate sectors (illustrated in Figure 6.4: "Connecting Low and High Edges of Basic and Duplicate Sectors"), the **/CYCEXPAND** command *combines* the solutions from the two sectors by performing computations on the selected load step (specified via the **SET** command) to combine the results of the two sectors. ANSYS employs the following response equation for the full structure or assembly:

$$U_j = U^A \cos(j-1)k\alpha - U^B \sin(j-1)k\alpha \tag{6–3}$$

where,

U_j = Response of the full structure or assembly (displacement, stress, or strain) for sector number j
U^A = Basic sector solution
U^B = Duplicate sector solution
j = Sector number for response expansion -- j = 1,2,3,…,*N*
k = Harmonic index -- (0,1,2,…,*N* / 2) when *N* is even, (0,1,2,…,(*N*-1) / 2) when *N* is odd. (*N* is an integer representing the number of sectors in 360°.)
α = Sector angle ($2\pi/N$)

The equation applies when expanding the static cyclic solution as well as the modal cyclic eigenvector solution.

If cyclic expansion via the **/CYCEXPAND** command is active for a static cyclic symmetry analysis, the **PLNSOL** and **PRNSOL** commands have summation of *all* required harmonic index solutions by default. (You can override the default behavior if necessary.)

A **SET**,LIST command lists the range of load step numbers in the group containing each solution. Each load step post data header contains the first, last, and count of load steps from the given **SOLVE** command, as shown:

```
        *****  INDEX OF DATA SETS ON RESULTS FILE  *****
SET  TIME/FREQ  LOAD STEP  SUBSTEP  CUMULATIVE  HRM-INDEX  GROUP
1    1.0000     1          1        1           0          1-3
2    2.0000     2          1        2           1          1-3
3    3.0000     3          1        3           2          1-3
4    1.0000     1          1        1           0          4-6
5    2.0000     2          1        2           1          4-6
6    3.0000     2          1        3           2          4-6
...
```

The **SET** command establishes which **SOLVE** load step group should display. Summation via **/CYCEXPAND** is automatic (although you can override the default behavior). Plots and printed output show the summation status.

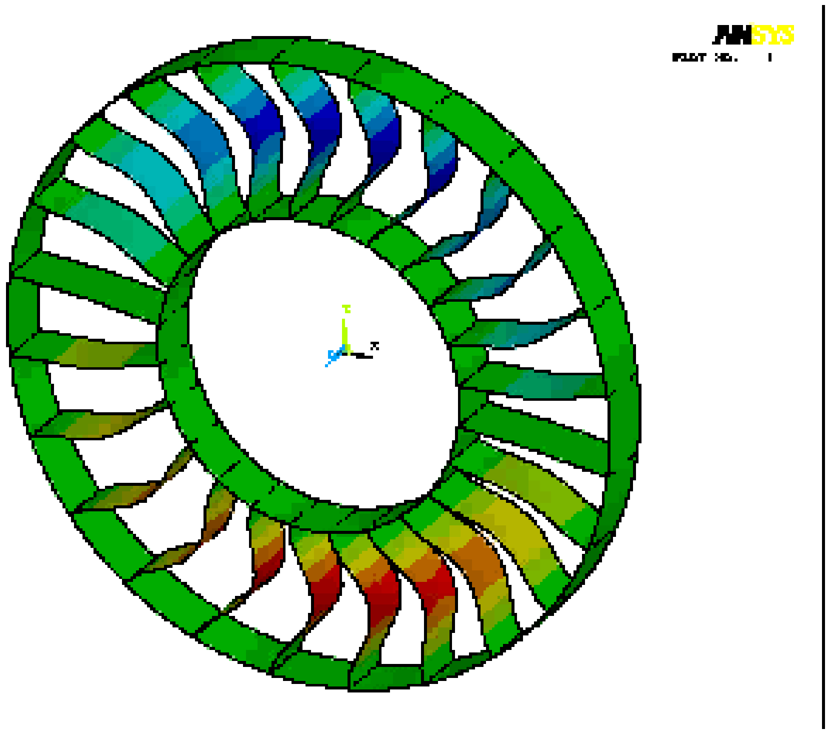
Accumulation occurs at the first applicable **PLNSOL** or **PRNSOL** command. After accumulation, the last load step number of the current group becomes the *new* current load step number.

6.4.2.1.1. Applying a Traveling Wave Animation to the Cyclic Model

After you have completed a modal cyclic symmetry analysis, you can apply an animated traveling wave to the cyclic model by issuing the **ANCYC** command (which employs **/CYCEXPAND** functionality). The traveling wave capability applies only to modal cyclic symmetry analyses. For more information, see the description of the **ANCYC** command in the *ANSYS Commands Reference*.

Figure 6.12: “Traveling Wave Animation Example” illustrates the **ANCYC** command's effect. To view the input file used to create the model shown, see Section 6.5: Sample Modal Cyclic Symmetry Analysis.

Figure 6.12 Traveling Wave Animation Example



6.4.2.2. Using the EXPAND Command

The **EXPAND** command offers an alternate method for displaying the results of a modal cyclic symmetry analysis. It is a specification command that causes a **SET** operation to transform and expand the data it is reading before storing it in the database. If you request two or more sector repetitions, the command creates additional nodes and elements to provide space for the extra results.

After the real-space results are stored in the database, you can plot (**PLESOL** or **PLNSOL**), print (**PRNSOL**) or process them as you would those for a non-cyclic model, a convenience if you want to process the results in a manner unsupported by the **/CYCEXPAND** command. The database can become very large, however, negating the inherent model-size advantage of a cyclic symmetry analysis.

Caution: Do not confuse the **EXPAND** command with **/EXPAND**.

6.4.3. Phase Sweep of Repeated Eigenvector Shapes

In a modal cyclic symmetry analysis, repeated eigenfrequencies are obtained at solutions corresponding to harmonic indices *k*, greater than 0 and less than *N*/2. The repeated modes are a consequence of the cyclically symmetric geometry of the structure or assembly being modeled by the cyclic sector.

The eigenvector shapes corresponding to the repeated eigenfrequencies are non-unique. That is, for the repeated eigenfrequencies $f_i = f_{i+1}$, the mode shapes corresponding to f_i and f_{i+1} can be linearly combined to obtain a mode shape that is also a valid mode shape solution for the frequencies f_i and f_{i+1}. A valid linear combination of the eigenvectors is:

$$U = c1 * U_i + c2 * U_{i+1} \tag{6–4}$$

where,

c1 and c2 = Arbitrary constants
U_i and U_{i+1} = Eigenvectors corresponding to f_i and f_{i+1}, respectively

The orientation of the combined mode shape U will be along a nodal diametral line that is neither along that of U_i nor U_{i+1}. Because the full structure may have stress-raising features (such as bolt holes), determining the eigenvector orientation that causes the most severe stresses, strains, or displacements on the structure or assembly is critical.

To determine the peak value of stress, strain or displacement in the full structure or assembly, it is necessary to calculate U at all possible angular orientations ϕ in the range of 0 through 360°. In the general postprocessor, the **CYCPHASE** command performs the computational task.

Because c1 and c2 are arbitrary constants, the **CYCPHASE** calculation rewrites Equation 6–4 as follows:

$$U = U_i \cos\phi - U_{i+1} \sin\phi \tag{6–5}$$

Using the cyclic symmetry expansion of Equation 6–3 in Equation 6–5, the simplified phase-sweep equation that operates on the cyclic sector solution (rather on the computation-intensive full-structure expression in Equation 6–5) is:

$$U_i(\phi) = U_i^A \cos\phi - U_i^B \sin\phi \tag{6–6}$$

A phase sweep using the **CYCPHASE** command provides information about the peak values of stress, strain and/or displacement components and the corresponding phase angle values. Using the phase angle value further, you can expand the mode shape at that phase angle to construct the eigenvector shape that produces the peak stress, strain and/or displacement. The expansion expression with the phase angle used by the **/CYCEXPAND** command is:

$$U_i = U^A \cos\{(j-1)k\alpha + \phi\} - U^B \sin\{(j-1)k\alpha + \phi\} \tag{6–7}$$

where,

j = 1,2,3,...,*N*

Example:

To determine the eigenvector orientation that causes the highest equivalent stress, perform a phase sweep on the stress via the **CYCPHASE**,STRESS command. Obtain a summary of the phase sweep via the **CYCPHASE**,STAT command to determine the value of ϕ at which maximum equivalent stress occurred. You can shift the mode shape to that angle via the **/CYCEXPAND**,,PHASEANG command and plot the expanded mode shape via the **PLNSOL**,S,EQV command.

6.5. Sample Modal Cyclic Symmetry Analysis

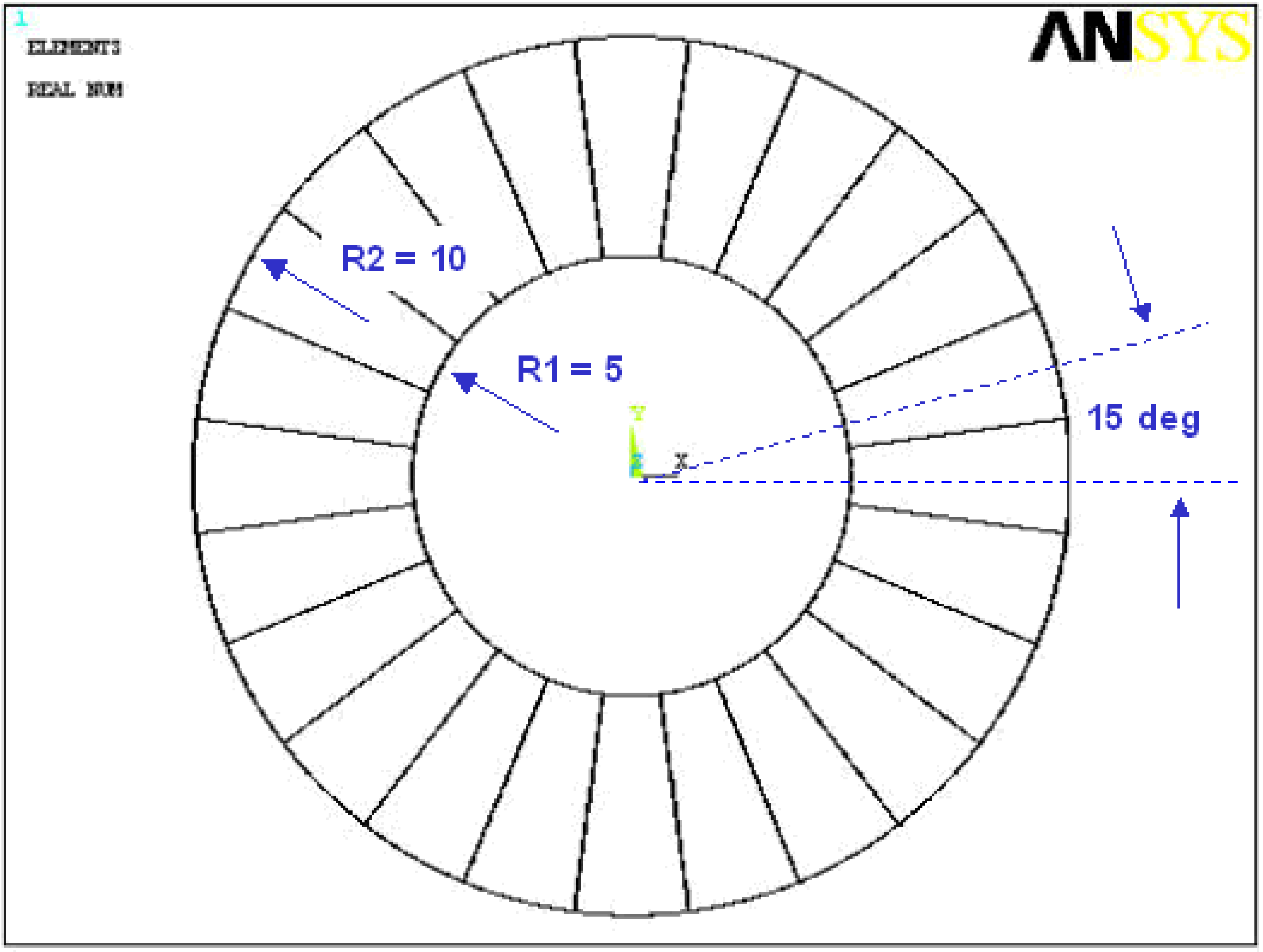
This section introduces you to the ANSYS product's automated cyclic symmetry analysis capabilities by way of example. The sample modal cyclic symmetry analysis presents a simplified “ring-strut-ring” structure used in many rotating-machinery applications.

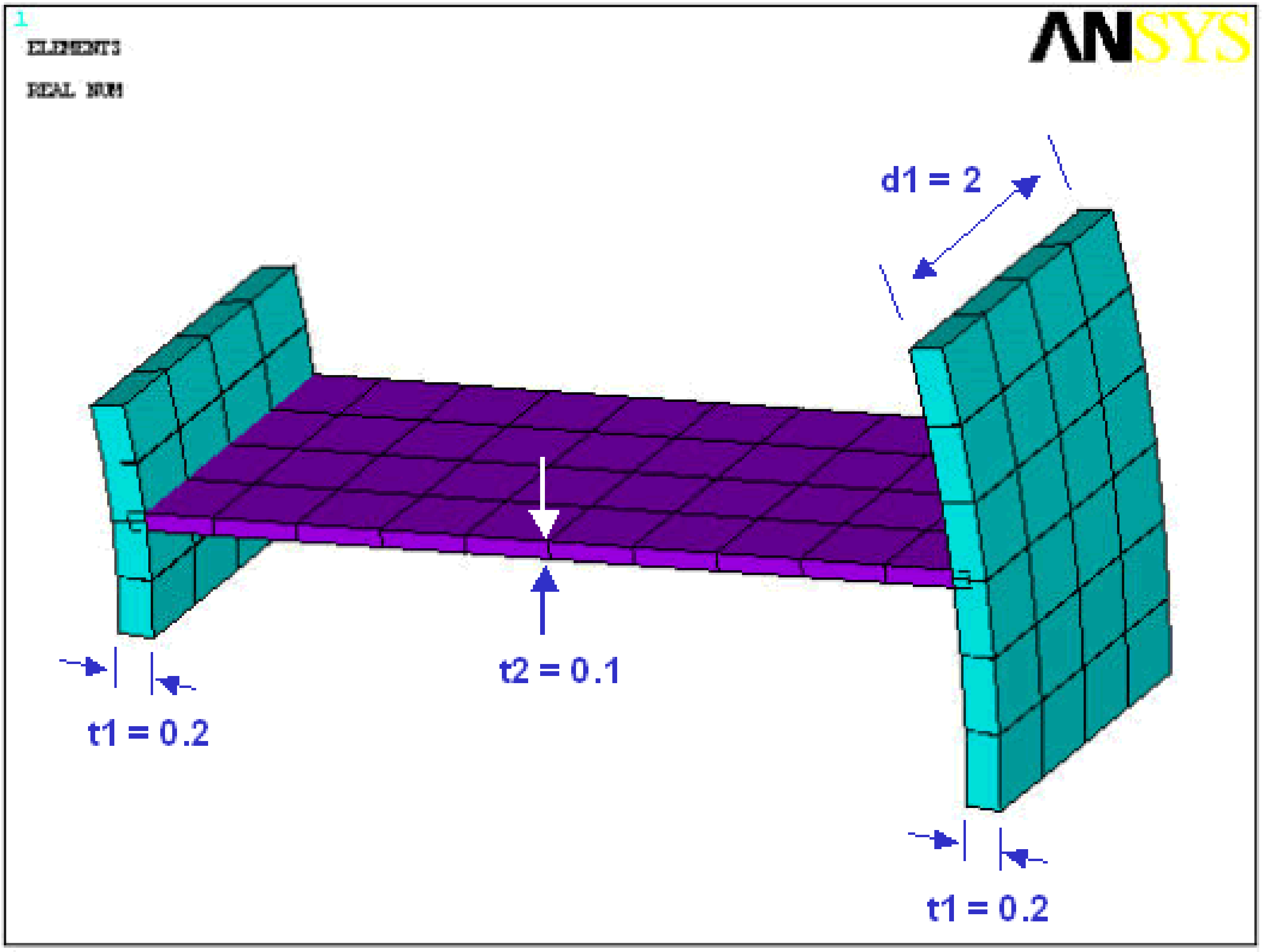
6.5.1. Problem Description

The component is a simplified fan inlet case for a military aircraft engine. As part of the design process for the assembly, you must determine the vibration characteristics (natural frequencies and mode shapes) of the inlet case.

6.5.2. Problem Specifications

The geometric properties for this analysis are as follows:





The material properties for this analysis are as follows:

Young's modulus (E) = 10e6
Poisson's ratio (υ) = 0.3
Density = 1e-4

All applicable degrees of freedom are used for the cyclic symmetry edge-component pairs. The first six mode shapes for all applicable harmonic indices are requested.

6.5.3. Input File for the Analysis

Use this input file (named **cyc_symm.inp**) to perform the example modal cyclic symmetry analysis. The file contains the complete geometry, material properties and solution options for the finite element model.

```
! Modal Cyclic Symmetry Analysis Example
! Ring-Strut-Ring Configuration

! STEP #1
! Start an ANSYS interactive session

! STEP #2
! Read in this input file: cyc_symm.inp

finish
/clear

r1=5
r2=10
```

```
d1=2
nsect=24
alpha_deg=360/nsect
alpha_rad=2*acos(-1)/nsect

/view,1,1,1,2
/plopts,minm,0
/plopts,date,0
/pnum,real,1
/number,1

/prep7
csys,1
k,1,0,0,0
k,2,0,0,d1
k,3,r1,0,0
k,4,r1,0,d1
l,3,4
arotat,1,,,,,,1,2,alpha_deg/2
k,7,r2,0,0
k,8,r2,0,d1
l,7,8
arotat,5,,,,,,1,2,alpha_deg/2
arotat,2,,,,,,1,2,alpha_deg/2
arotat,6,,,,,,1,2,alpha_deg/2
a,5,6,10,9
mshkey,1
et,1,181
r,1,0.20
r,2,0.1
mp,ex,1,10e6
mp,prxy,1,0.3
mp,dens,1,1e-4
esize,0.5
asel,,,,1,4
aatt,,1
asel,,,,5
aatt,,2
allsel
finish

/solution
antype,modal
modopt,lanb,6
mxpand,6,,,yes
dk,5,uz,0
finish

aplot
/prep7

/eof

! STEP #3
! Configure the database for a cyclic symmetry analysis

cyclic

! STEP #4
! Mesh the areas

amesh,all

! STEP #5
! Turn on cyclic symmetry graphical expansion

/cycexpand,,on

! STEP #6
! Plot the elements

eplot
```

```
! STEP #7
! List the cyclic status

cyclic,status

! STEP #8
! List the cyclic solution option settings

cycopt,status

! STEP #9
! Solve the modal cyclic symmetry analysis

/solution
solve

! STEP #10
! Specify global cylindrical as the results coordinate system

/post1
rsys,1

! STEP #11
! Read results for "load step 1 - substep 4 - harmonic index 0"

set,1,4

! STEP #12
! Plot the tangential displacement contour

plns,u,y

! STEP #13
! Read results for "load step 13 - substep 1 - harmonic index 12"

set,13,1

! STEP #14
! Plot the tangential displacement contour

plns,u,y

! STEP #15
! Read results for "load step 2 - substep 5 - harmonic index 1"

set,2,5

! STEP #16
! Plot the tangential displacement contour

plns,u,y
```

6.5.4. Analysis Steps

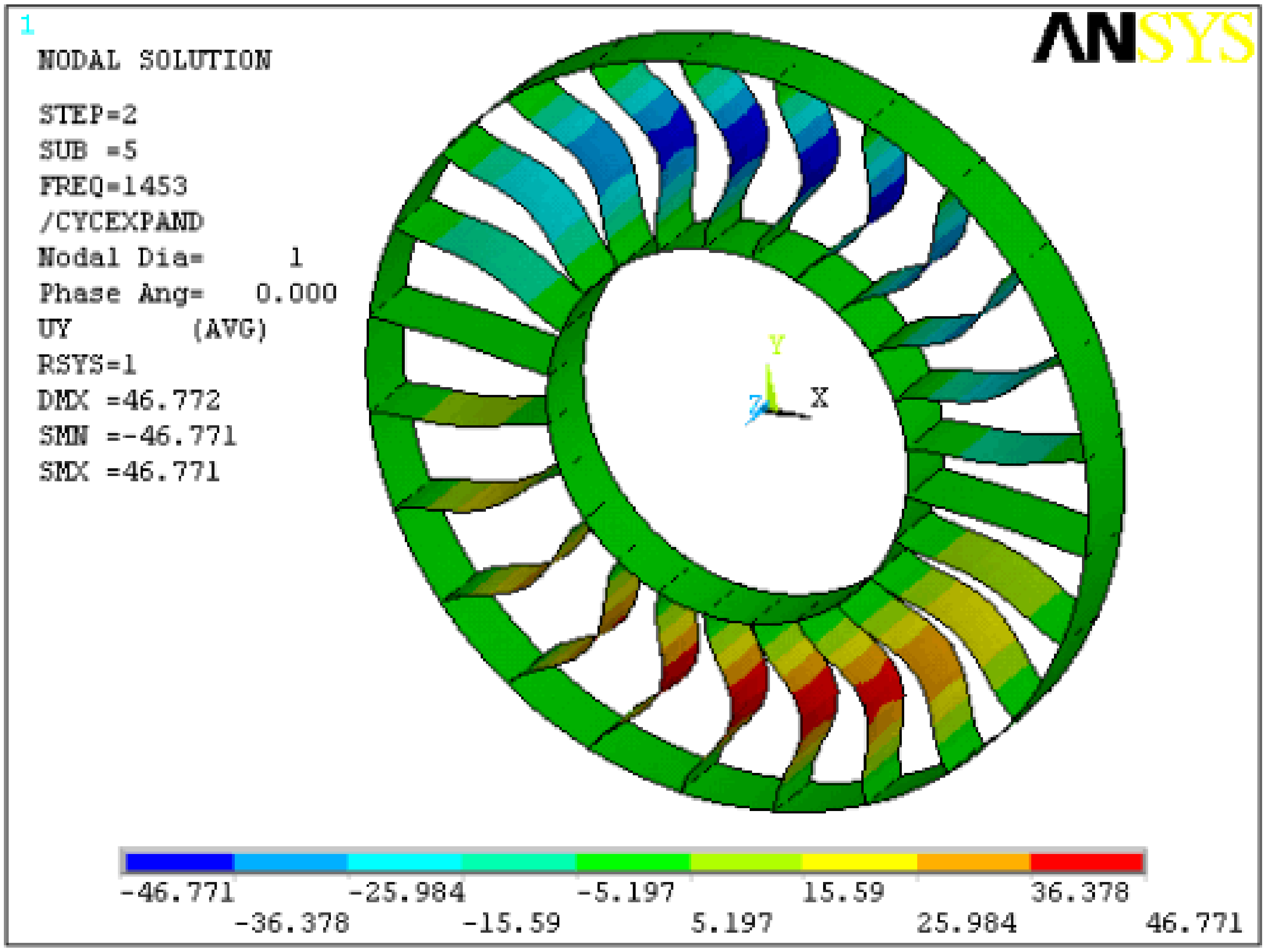
The following table describes the input listing and the steps involved in the sample modal cyclic symmetry analysis in more detail.

Step	Description	ANSYS Command
1.	Start an ANSYS interactive session.	---
2.	Read the input file: **cyc_symm.inp**	**/INPUT**,CYC_SYMM.INP
3.	Specify a cyclic symmetry analysis and configure the database accordingly.	**CYCLIC**
4.	Mesh the areas.	**AMESH**,ALL
5.	Activate cyclic symmetry graphical expansion.	**/CYCEXPAND**,,ON
6.	Plot the elements.	**EPLOT**

Step	Description	ANSYS Command
7.	List the cyclic status.	**CYCLIC**,STATUS
8.	List the cyclic solution option settings.	**CYCOPT**,STATUS
9.	Solve the modal cyclic symmetry analysis.	**/SOLU** **SOLVE**
10.	Specify the global cylindrical coordinate system.	**/POST1** **RSYS**,1
11.	Read results for "load step 1 - substep 4 - harmonic index 0."	**SET**,1,4
12.	Plot the tangential displacement contour. *Executing this step causes the struts of the assembly to bend "in phase."*	**PLNSOL**,U,Y
13.	Read results for "load step 13 - substep 1 - harmonic index 12."	**SET**,13,1
14.	Plot the tangential displacement contour. *Executing this step causes the struts of the assembly to bend "out of phase."*	**PLNSOL**,U,Y
15.	Read results for "load step 2 - substep 5 - harmonic index 1."	**SET**,2,5
16.	Plot the tangential displacement contour. ---- *This step completes the sample modal cyclic symmetry analysis. Your results should match those shown in Figure 6.13: "Sample Modal Cyclic Symmetry Analysis Results".*	**PLNSOL**,U,Y

The results of your analysis should match those shown here:

Figure 6.13 Sample Modal Cyclic Symmetry Analysis Results



To view a traveling wave animation of your model, issue the **ANCYC**,24,,0.1 command. For more information, see Section 6.4.2.1.1: Applying a Traveling Wave Animation to the Cyclic Model.

6.6. Sample Buckling Cyclic Symmetry Analysis

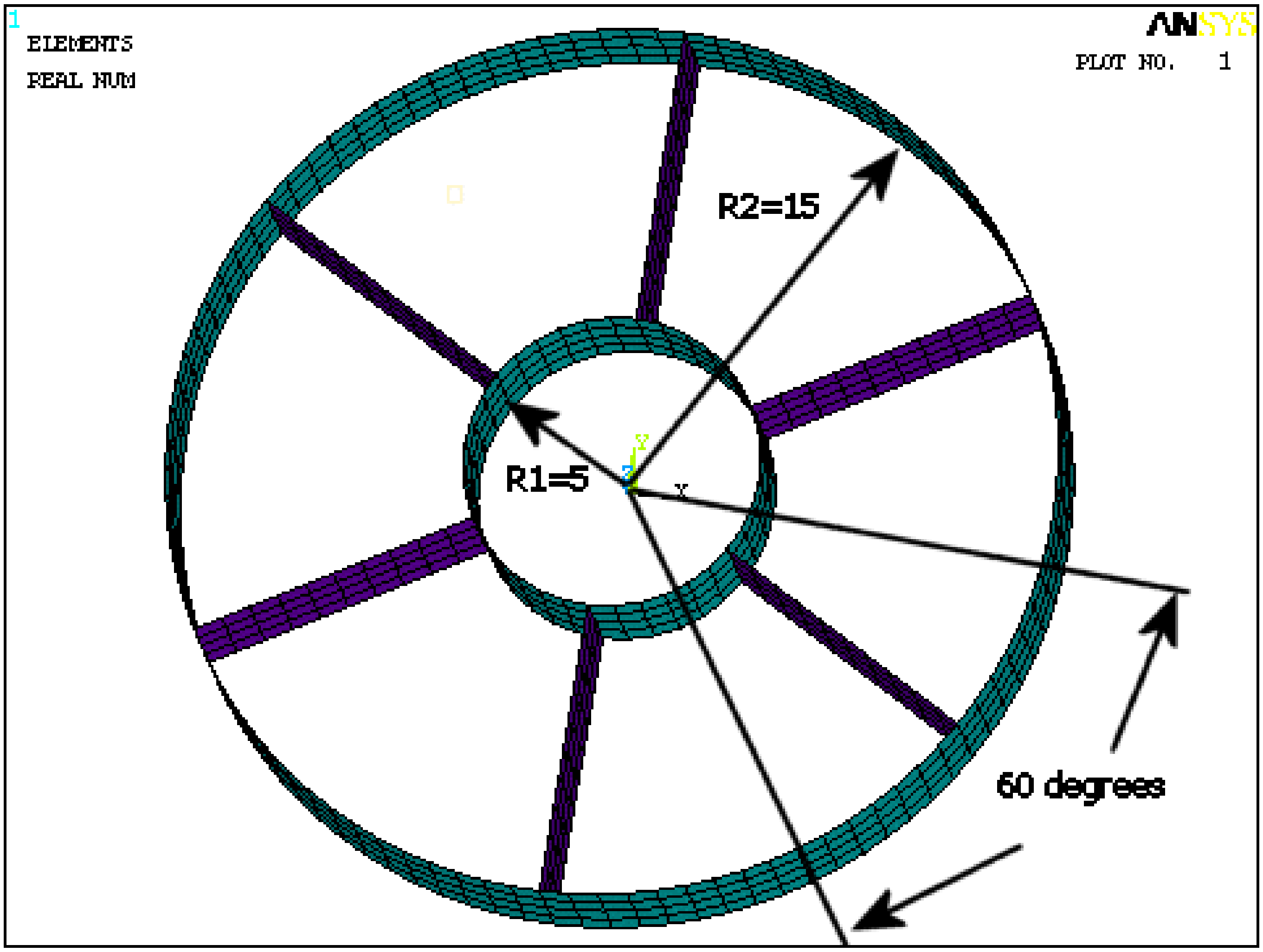
This section introduces you to the ANSYS product's automated cyclic symmetry analysis capabilities by way of example. The sample buckling cyclic symmetry analysis presents a simplified “ring-strut-ring” structure used in many rotating-machinery applications.

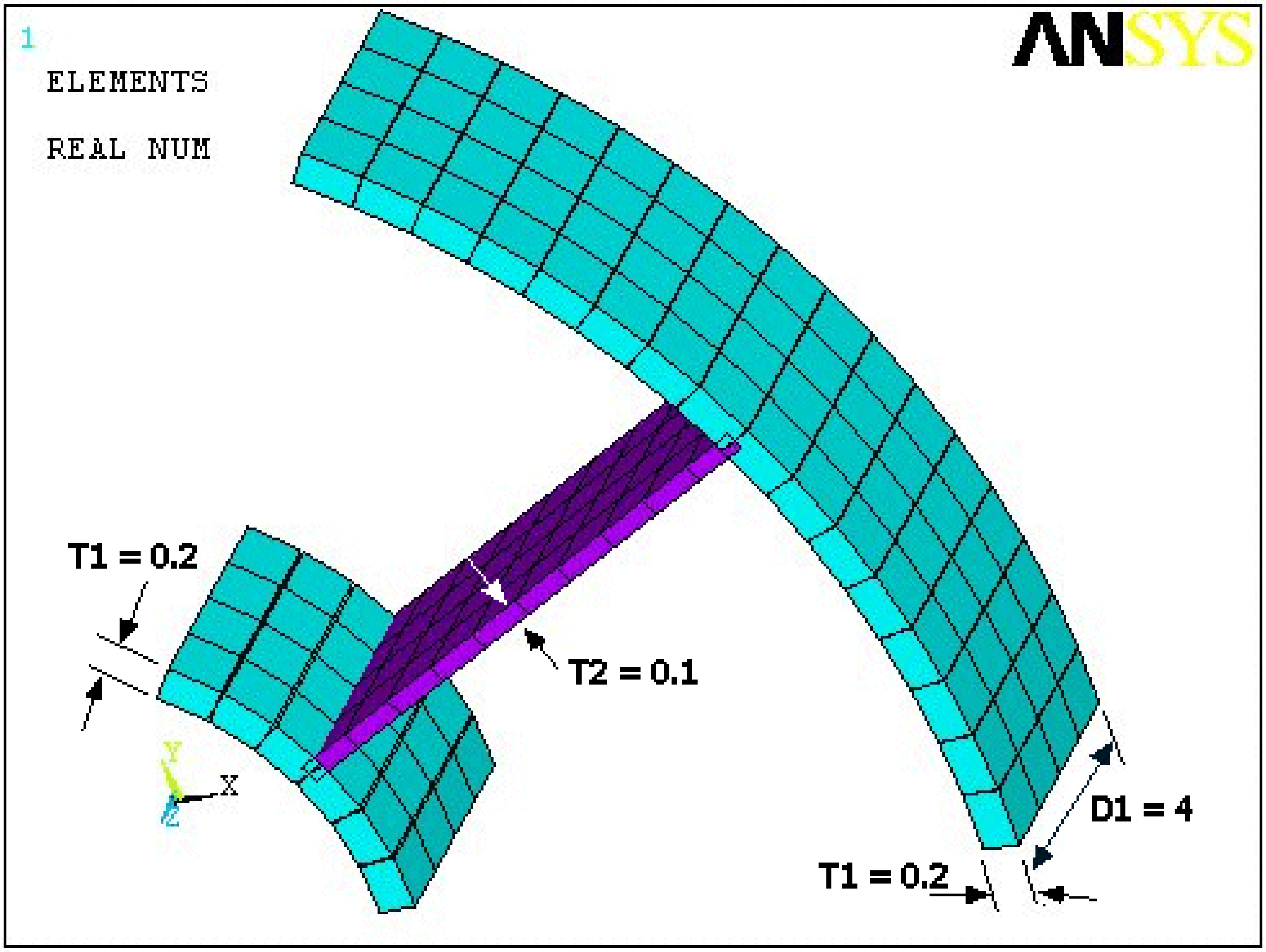
6.6.1. Problem Description

The object is a simplified structure that experiences a thermal load emanating outward from the center. The inner ring is kept at a constant 600° F temperature and the outer ring is kept at a constant 0° F. A linear eigenvalue buckling analysis determines when the struts will buckle as the temperature in the struts increases from 100° F to 500° F.

6.6.2. Problem Specifications

The geometric properties for this analysis are as follows:


1
ELEMENTS
REAL NUM
ANSYS
PLOT NO. 1
R2=15
R1=5
Y
X
Z
60 degrees



The material properties for this analysis are as follows:

Poisson's ratio (υ) = 0.3
Density = 1e-4
Coefficient of thermal expansion (α) = 5e-5
Young's modulus (E) = 10e6 (at 0° F)
Young’s modulus (E) = 4e6 (at 600° F)

The Young’s modulus value varies linearly between 0 and 600° F. All applicable degrees of freedom (DOFs) are used for the cyclic symmetry edge-component pairs. The first six mode shapes for all applicable harmonic indices are requested.

6.6.3. Input File for the Analysis

Use this input file (named **buck_cyc_sym.inp**) to perform the buckling cyclic symmetry analysis example. The file contains the complete geometry, material properties and solution options for the finite element model.

```
! Cyclic Symmetry Buckling Example
! Ring-Strut-Ring Configuration

! STEP #1
! Start an ANSYS interactive session

! STEP #2
! Read in the input file: buck_cyc_sym.inp

r1=5
r2=15
d1=4
nsect=6
alpha_deg=360/nsect
alpha_rad=2*acos(-1)/nsect

/view,1,1,1,2
/plopts,minm,0
/plopts,date,0
/pnum,real,1
/number,1

/prep7
csys,1
k,1,0,0,0
k,2,0,0,d1
k,3,r1,0,0
k,4,r1,0,d1
l,3,4
arotat,1,,,,,,1,2,alpha_deg/2
k,7,r2,0,0
k,8,r2,0,d1
l,7,8
arotat,5,,,,,,1,2,alpha_deg/2
arotat,2,,,,,,1,2,alpha_deg/2
arotat,6,,,,,,1,2,alpha_deg/2
a,5,6,10,9
mshkey,1
et,1,181
r,1,0.20
r,2,0.1
mptemp,1,0
mptemp,2,600
mpdata,ex,1,1,10e6
mpdata,ex,1,2,4e6
mp,prxy,1,0.3,0.0
mp,dens,1,1e-4
mp,alpx,1,5e-5
tref,0
esize,1.0
asel,,loc,x,r1
bfa,all,temp,600
asel,a,loc,x,r2
aatt,,1
asel,inve
bfa,all,temp,100
aatt,,2
allsel
amesh,all
lsel,,loc,z,d1/2
lsel,r,loc,y,alpha_deg/2
ksll
nslk
nrotate,all
dk,all,uz,0
dk,all,uy,0
allsel
finish
aplot
/prep7

/eof

! STEP #3
! Configure the database for a cyclic symmetry analysis

cyclic

! STEP #4
! Turn on cyclic symmetry graphical expansion

/cycexpand,,on

! STEP #5
! Plot the elements

eplot

! STEP #6
! List the cyclic status

cyclic,status

! STEP #7
! List the cyclic solution option settings

cycopt,status

! STEP #8
! Specify static analysis type with prestress effects

/solution
antype,static
pstres,on

! STEP #9
! Solve the prestress static analysis

solve

! STEP #10
! Specify buckling analysis type

finish
/solution
antype,buckle

! STEP #11
! Specify buckling analysis options

bucopt,lanb,3

! STEP #12
! Specify mode expansion options

mxpand,3,,,yes

! STEP #13
! Solve the buckling analysis

solve

! STEP #14
! Read results for the smallest load factor

finish
/post1
set,first,,,,,,,order

! STEP #15
! Plot the buckled mode shape

plnsol,u,sum
```

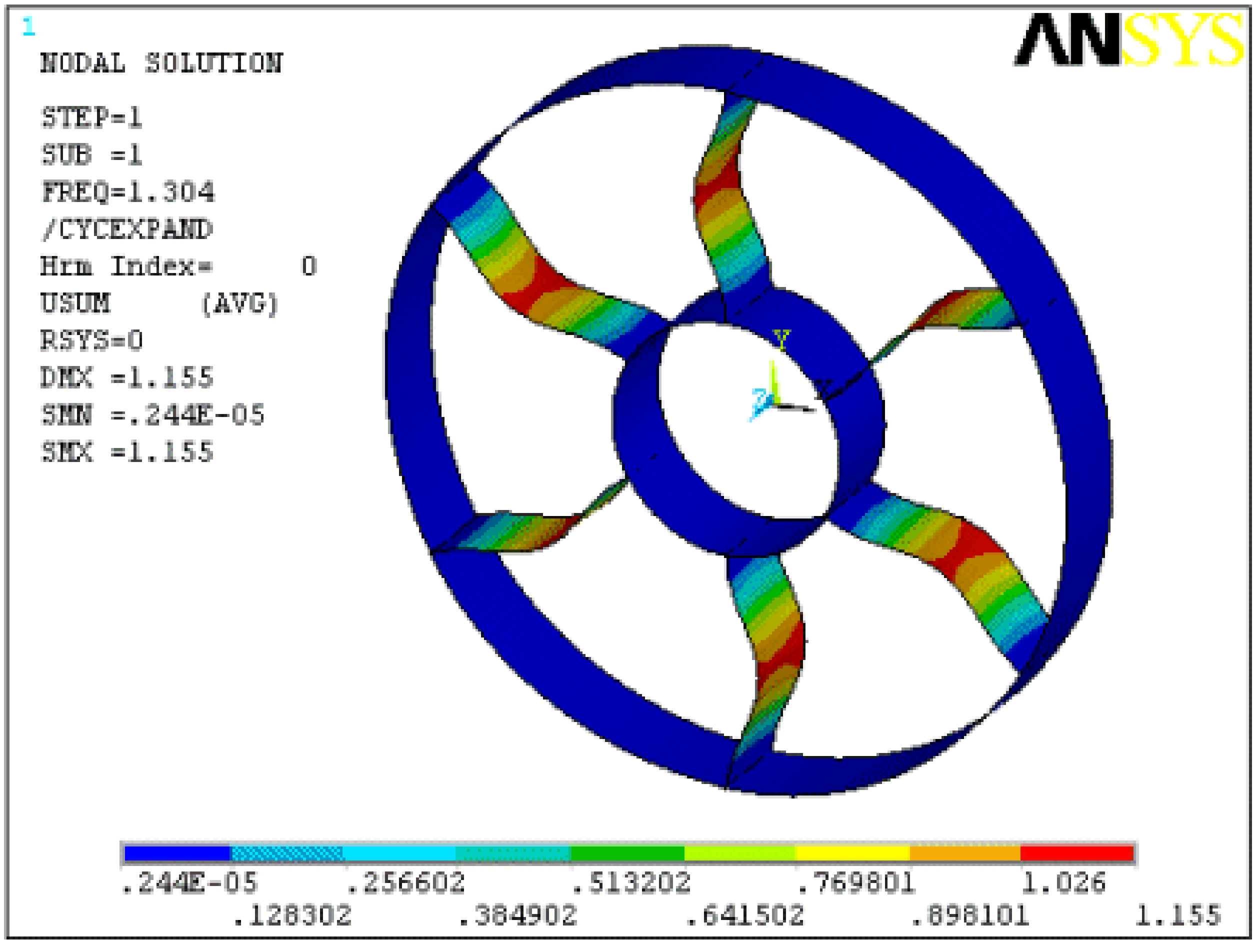
6.6.4. Analysis Steps

The following table describes the input listing and the steps involved in the sample buckling cyclic symmetry analysis in more detail.

Step	**Description**	**ANSYS Command**
1.	Start an ANSYS interactive session.	---
2.	Read in the input file: **buck_cyc_sym.inp**	**/IN-PUT**,buck_cyc_sym.inp
3.	Specify a cyclic symmetry analysis and configure the database accordingly.	**CYCLIC**
4.	Activate cyclic symmetry graphical expansion.	**/CYCEXPAND**,,ON
5.	Plot the elements.	**EPLOT**
6.	List the cyclic status.	**CYCLIC**,STATUS
7.	List the cyclic solution option settings.	**CYCOPT**,STATUS
8.	Specify a static analysis type with prestress effects.	**/SOLU** **ANTYPE**,STATIC **PSTRES**, ON
9.	Solve the prestress static analysis.	**SOLVE**
10.	Specify a buckling analysis type.	**FINISH** **/SOLU** **ANTYPE**,BUCKLE
11.	Specify buckling analysis options	**BUCOPT**, LANB, 3
12.	Specify mode expansion options.	**MXPAND**, 3, , , YES
13.	Solve the buckling analysis.	**SOLVE**
14.	Read the results from the smallest load factor. (This should correspond to the smallest frequency.)	**FINISH** **/POST1** **SET**, FIRST , , , , , , , ORDER
15.	Plot the buckled mode shape. ---- *This step completes the sample buckling cyclic symmetry analysis. Your results should match those shown in Figure 6.14: "Sample Buckling Cyclic Symmetry Analysis Results".*	**PLNSOL**, U, SUM

The results of your analysis should match those shown here:

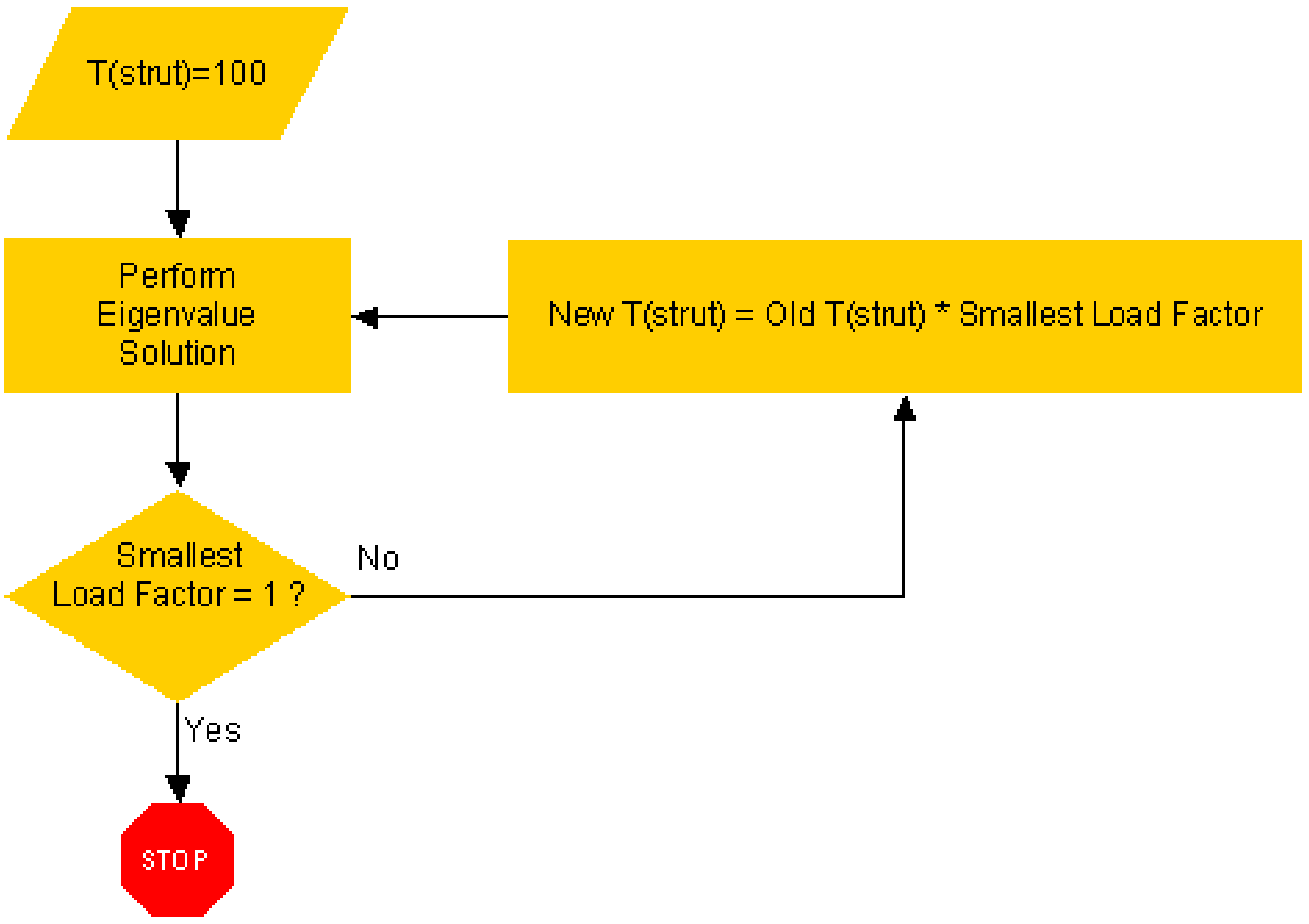
Figure 6.14 Sample Buckling Cyclic Symmetry Analysis Results



6.6.5. Solve For Critical Strut Temperature at Load Factor = 1.0

You can automatically solve for the critical strut temperature by iterating on the variable loads until the eigenvalue becomes 1.0 (or nearly 1.0 within some tolerance). The iterations ensure that the eigenvalue solution does not factor the stress stiffness matrix from the constant loads. The following flowchart illustrates the process:

Figure 6.15 Buckling Cyclic Symmetry Results: Load Factor Iterations


T(strut)=100
Perform Eigenvalue Solution
New T(strut) = Old T(strut) * Smallest Load Factor
Smallest Load Factor = 1 ?
No
Yes
STOP

Use the **/PREP7** portion of the previous input file (**buck_cyc_sym.inp**) to construct your model. After defining the model parameters--but before activating cyclic symmetry--define the arrays and the programming operations, as follows:

```
*dim,Tstrut,array,10
*dim,Tfact,array,10

*do,I,1,10

  /prep7

  *if,I,eq,1,then
    Tstrut(I)=100
  *else
    Tstrut(I)=Tstrut(I-1)*Tfact(I-1)
    cyclic,undouble
  *endif

  asel,,real,,2
  bfa,all,temp,Tstrut(I)
  allsel
```

After you have defined the iterative parameters, proceed with the cyclic symmetry portion of the analysis:

```
  cyclic
  /cycexpand,,on
  eplot
```

```
      cyclic,status
      cycopt,status
      /solution
      antype,static
      pstres,on
      solve
      finish
      /solution
      antype,buckle
      bucopt,lanb,3
      mxpand,3,,,yes
      solve
      finish
      /post1
      set,first,,,,,,,order
      plnsol,u,sum

      *get,loadmult,active,,set,freq
      Tfact(I)=loadmult

    *enddo
```

ANSYS then plots the data to determine the critical strut temperature:

```
    *dim,data,table,10,2
    data(0,1)=1
    data(0,2)=2

    *do,I,1,10

      data(I,0)=I
      data(I,1)=Tstrut(I)
      data(I,2)=Tfact(I)

    *enddo

    /AXLAB,X,Strut Temperature
    /AXLAB,Y,Load Factor
    /GROPT,DIVX,5
    /GROPT,DIVY,5
    /XRANGE,100,200
    /YRANGE,0.9,1.4
    /GTHK,CURVE,1
    /GMARKER,1,3

    *VPLOT,data(1,1),data(1,2)
```

The eigenvalues (frequencies) calculated for the buckling analysis represent the buckling load factors. The eigenvalues represent load factors for *all* applied loads.

The iteration strategy yields the following results:

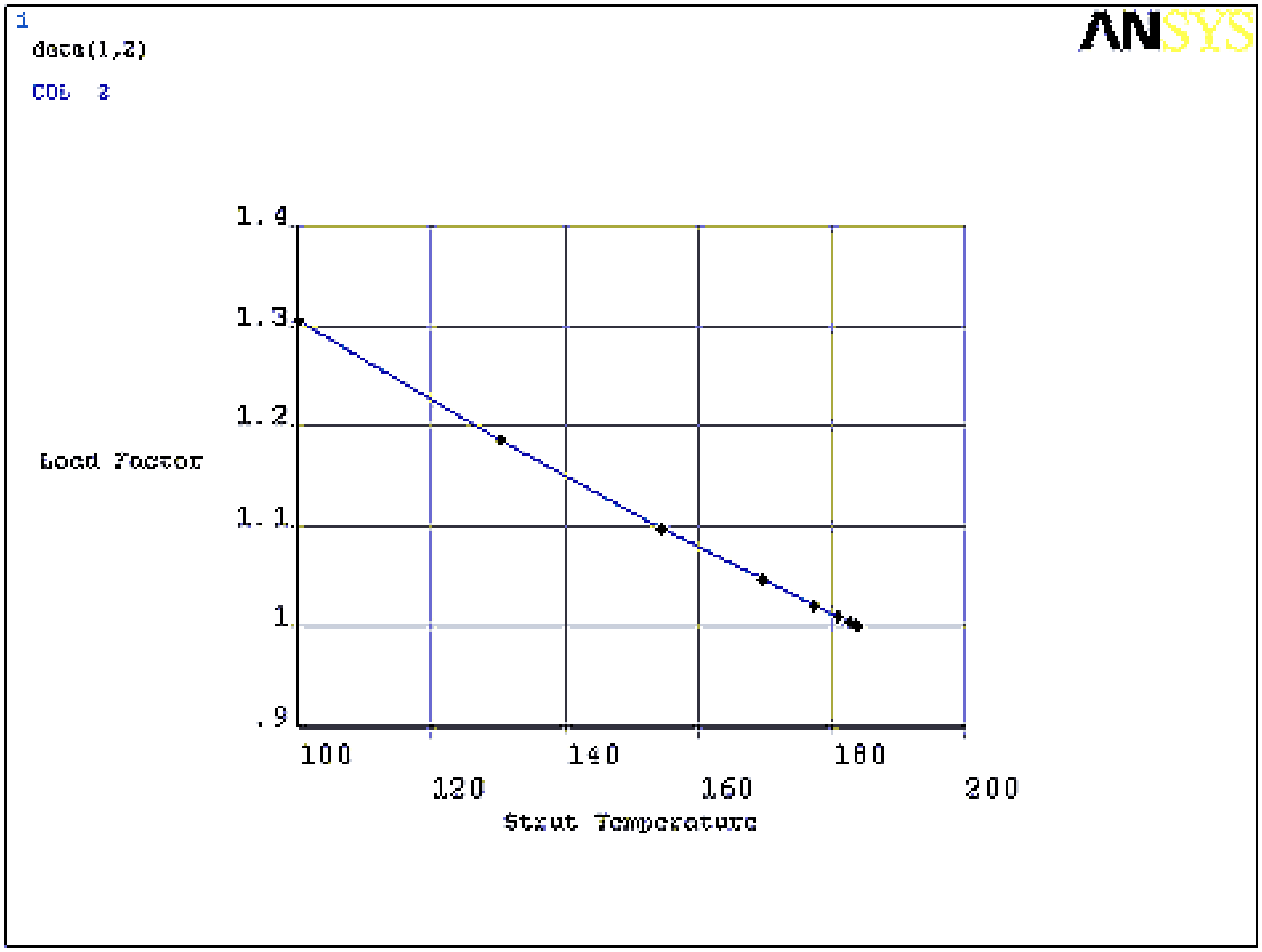
Table 6.2 Buckling Cyclic Symmetry: Load Factor Iteration Results

Iteration	T° (Strut)	Load Factor
1	100.00	1.3039
2	130.39	1.1845
3	154.44	1.0972
4	169.45	1.0461
5	177.27	1.0206
6	180.91	1.0089
7	182.52	1.0038

Iteration	T° (Strut)	Load Factor
8	183.21	1.0016
9	183.50	1.0007
10	183.62	1.0003

A graph of the results shows the convergence at Load Factor = 1.0:

Figure 6.16 Buckling Cyclic Symmetry Results: Load Factor Results Graph



6.7. Sample Magnetic Cyclic Symmetry Analysis

This section further explains ANSYS' magnetic cyclic symmetry capabilities via an example. This example presents a simplified electrical machine where the model size can be reduced with cyclic boundary conditions.

6.7.1. Problem Description

Figure 6.17: "Two-Phase Electric Machine – Full Model" shows a typical example, the full model of a 2-phase electrical machine.

In the full model, flux parallel boundary conditions can be formulated at the outer surface of the stator frame. If only phase A were excited, the magnetic flux would point in the y direction at x=0 plane; flux parallel condition could be formulated at the x=0 plane, allowing an analysis on a half model in the x>=0 space. Similarly, if only phase B were excited, the magnetic flux would have only x component on the y=0 plane; again, flux parallel could be applied to a half model in the y>=0 plane.

Typically, however, both coils are excited, and no flux parallel conditions could be formulated over the x=0 or y=0 planes. However, due to the cyclic nature of the field, the field pattern repeats itself after 180 degrees. In particular, on the y=0 plane:

By(x) = By(-x)

A similar pattern can be observed in Figure 6.18: "Two-Phase Electric Machine – Half Model", where the flux lines (equi vector potential lines) are plotted:

Az(x) = - Az(-x)

In this example, the field has a two pole pattern. In general, there are 2p poles; the repetition would take place after 180/p degrees.

Figure 6.17 Two-Phase Electric Machine – Full Model

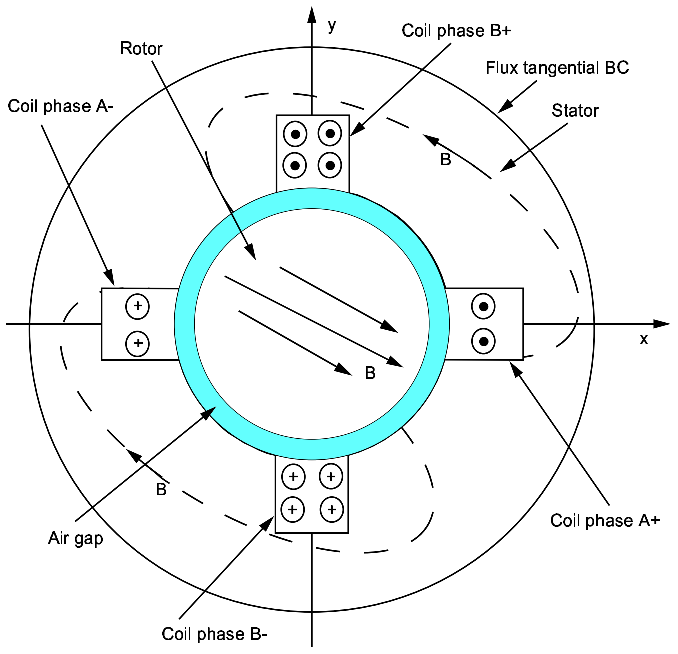
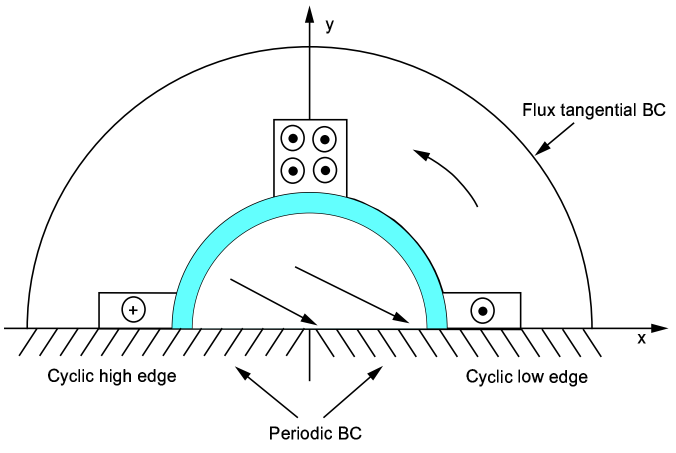


Figure 6.18 Two-Phase Electric Machine – Half Model



6.7.2. Problem Specifications

The material properties for this analysis are as follows:

Iron relative permeability: 1000

Iron electrical resistivity: 9.579E-8

Aluminum relative permeability: 1.0

Aluminum electrical resistivity: 2.65E-8

Copper relative permeability: 1.0

Copper electrical resistivity: 1.74E-8

6.7.3. Input file for the Analysis

Use this input file to perform the example magnetic cyclic symmetry analysis. This file contains the complete geometry, material properties, and solution options for the finite element model. Magnetic cyclic symmetry commands of particular interest are preceded by the comment:

```
!!!  Apply Cylic

/title,Cyclic Symmetry Model for EMAG Analysis (Dual Coils with Iron Yoke)
/com
/com ***** Quarter Symmetry Model Expanded to Half Then to Full *****
/com
/com
/com
/nopr
/out,scratch

!!! Setup Model Parameters


_geomgen=1
p=1                       ! Use for number of quarter sectors
                          !  (i.e. 1 = 1 90deg sector, 2 = 2 sectors in 90deg)
alpha=22.5/p              ! angle up to the end of first coil
beta=alpha+(45/p)         ! angle from coil1 to coil2
gamma=beta+(22.5/p)       ! angle from beginning of coil2 to end of sector
r1=3
r2=4.5
r3=5
```

```
r4=7
r5=11
ncoil=(4*p)
i1=1
i2=2

*dim,alpha1,,ncoil
*dim,alpha2,,ncoil
*dim,current,,ncoil
*dim,coilname,string,ncoil

coilname(1) = 'coil1'
coilname(2) = 'coil2'
coilname(3) = 'coil3'
coilname(4) = 'coil4'

*do,i,1,ncoil

alpha1(i) = -alpha + (i-1)*(90/p)
alpha2(i) = alpha + (i-1)*(90/p)

*enddo

ii=0

*do,i,1,p

 ii = ii + 1
 current(ii) = i2
 ii = ii + 1
 current(ii) = i1
 ii = ii + 1
 current(ii) = -i2
 ii = ii + 1
 current(ii) = -i1

*enddo


/prep7


ET,1,13,4                  ! Use PLANE13 Elements (DOFset = UX,UY,TEMP,AZ)

!!! Setup Model using Parameters

PCIRC,r1, ,0,alpha,
PCIRC,r1, ,0,beta
PCIRC,r1, ,0,gamma
PCIRC,r2, ,0,alpha
PCIRC,r2, ,0,beta
PCIRC,r2, ,0,gamma
PCIRC,r3, ,0,alpha
PCIRC,r3, ,0,beta
PCIRC,r3, ,0,gamma
PCIRC,r4, ,0,alpha
PCIRC,r4, ,0,beta
PCIRC,r4, ,0,gamma
PCIRC,r5, ,0,alpha
PCIRC,r5, ,0,beta
PCIRC,r5, ,0,gamma
AOVLAP,ALL

!!! Setup Material Properties

! IRON
MP,MURX,1,1000
MP,RSVX,1,9.579E-8

! AL
MP,MURX,2,1
MP,RSVX,2,2.65E-8
```

```
! Copper
MP,MURX,3,1
MP,RSVX,3,1.74E-8

! Air
MP,MURX,4,1
MP,RSVX,4,0

!!! Setup Components and Atributes

! Iron Core
CSYS,1                   ! Enter Cylindrical Mode
ASEL,S,LOC,X,0,r1
CM,Inner_Iron,AREA
AATT,1,,1,

! Al Core
ASEL,S,LOC,X,r1,r2
CM,Outer_AL,AREA
AATT,2,,1,

! Air Gap
ASEL,S,LOC,X,r2,r3
CM,AIR,AREA
AATT,4,,1

! Coil 1
ASEL,S,LOC,X,r3,r4
ASEL,R,LOC,Y,0,alpha
CM,COIL1,AREA
AATT,3,,1

! Coil 2
ASEL,S,LOC,X,r3,r4
ASEL,R,LOC,Y,beta,gamma
CM,COIL2,AREA
AATT,3,,1

! Iron Yoke
ASEL,S,LOC,X,r3,r4
ASEL,R,LOC,Y,alpha,beta
ASEL,A,LOC,X,r4,r5
CM,YOKE,AREA
AATT,1,,1
ALLSEL
CSYS,0                   ! Enter Cartesian Mode

!!! Setup and Mesh Model

MSHKEY,1
CSYS,1
LSEL,S,LOC,Y,0
LSEL,A,LOC,Y,gamma
LESIZE,ALL,,,6,,1,,,1,
CMSEL,S,Inner_Iron
AMESH,ALL
CMSEL,S,Outer_AL
AMESH,ALL
CMSEL,S,Air
AMESH,ALL
CMSEL,S,Coil1
AMESH,ALL
CMSEL,S,Coil2
AMESH,ALL
CMSEL,S,Yoke
AMESH,ALL
ALLSEL
CSYS,0

!!! Reflect Model across X-axis
!!  Create HALF model from QUARTER model
```

```
arsym,x,all


/prep7
save,magtest,db         ! save half model for cyclic

arsym,y,all               ! create full model reflecting on y axis
nummrg,all

csys,1
nsel,s,loc,x,r5
CM,extnode,NODE

! Apply BFE Current loads to each coil

*do,i,1,ncoil

 asel,s,loc,x,r3,r4
 asel,r,loc,y,alpha1(i),alpha2(i)
 esla,s
 cm,coilname(i),element
 bfe,all,js,,,,current(i)

*enddo

csys,0

allsel
cmsel,s,extnode
d,all,az,0
d,all,temp,25
allsel
FINISH

/solu
/out,scratch

antype,static
allsel
solve

FINISH

/post1


!!! Plot Out Result Plots
plvect,b,,,,VECT,ELEM,ON,0

FINISH
parsav,all
/clear,nostart
resume,magtest,db       ! Resume half Model
parres,new

!! Delete Bottom half of model and all loading attatched to bottom nodes

/prep7

allsel
nummrg,all

csys,1
nsel,s,loc,x,r5
D,all,az,0  ! AZ = 0 on outside nodes of arc
D,all,temp,25

!! Define Coils on Half Model

! Coil 1
ASEL,S,LOC,X,r3,r4
```

```
ASEL,R,LOC,Y,0,alpha
esla,s
CM,COIL1,ELEMENT
! Coil 2
ASEL,S,LOC,X,r3,r4
ASEL,R,LOC,Y,beta,(180-beta)
esla,s
CM,COIL2,ELEMENT
! Coil 3
ASEL,S,LOC,X,r3,r4
ASEL,R,LOC,Y,(180-alpha),180
esla,s
CM,COIL3,ELEMENT

!! Apply bfe loads to Half Model coils


cmsel,s,COIL1
bfe,all,js,,,,i2
cmsel,s,COIL2
bfe,all,js,,,,i1
cmsel,s,COIL3
bfe,all,js,,,,(-i2)

!!!  Apply cyclic - create cyclic model with two sectors

allsel
csys,0
cyclic,2

/solution

cycopt,hindex,odd         ! Odd Symmetry for half model
solve
FINISH

/post1
/out


!!! Plot Out Result Plots
/vscale,1,1,1
plvect,b,,,,VECT,ELEM,ON,0    ! See figure for B field plot.


!!! Plot Out Contour Line Plot of Equipotentials
plf2d

FINISH
```

Figure 6.19 Vector Plot of Cyclic Flux Density (B) - Half Model

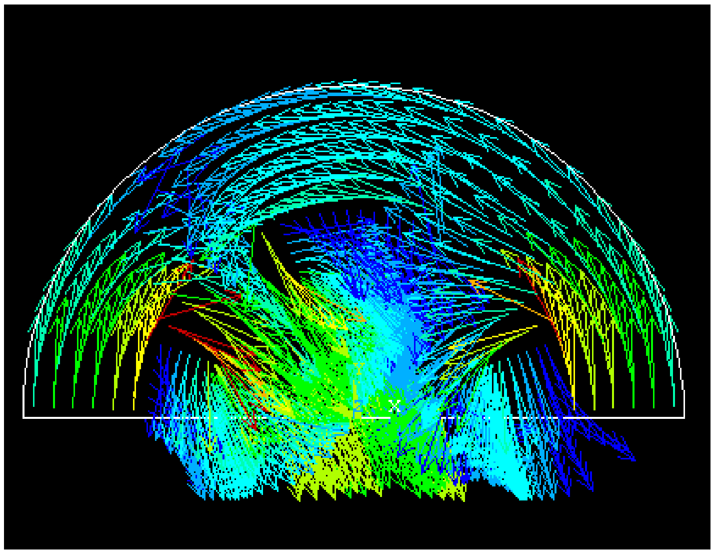
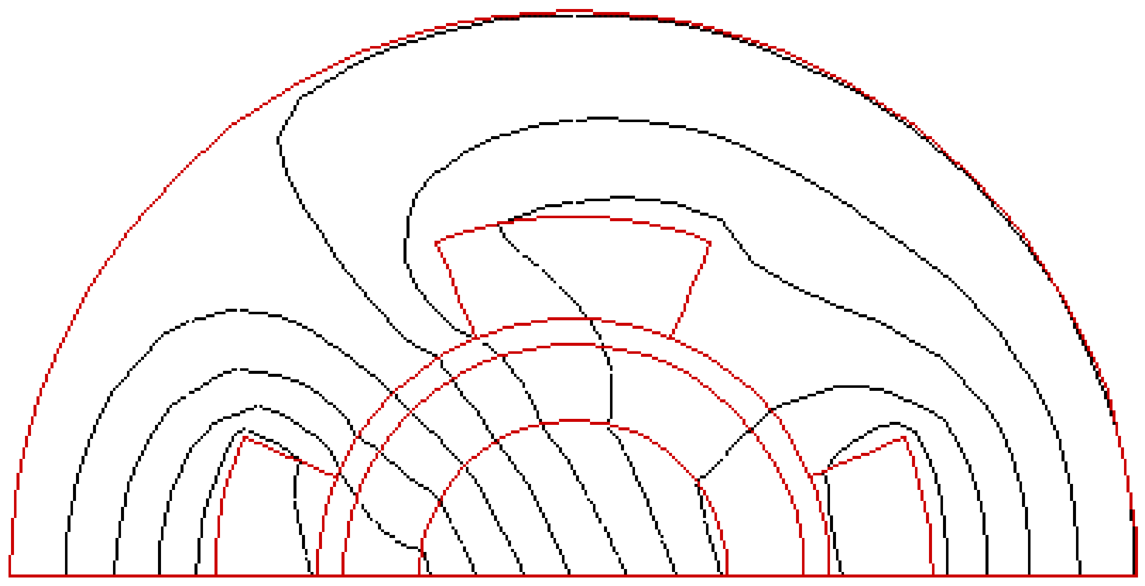


Figure 6.20 Contour Line Plot of Equipotentials

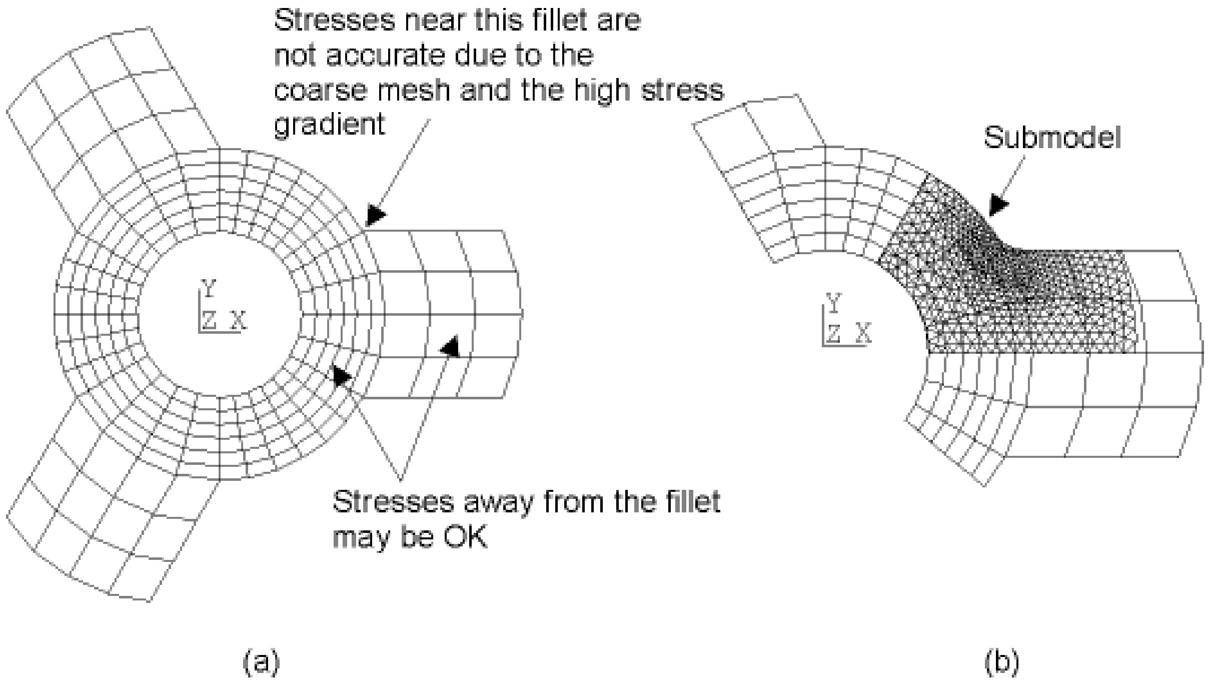


Chapter 7: Submodeling

Submodeling is a finite element technique used to get more accurate results in a region of your model. Often in finite element analysis, the finite element mesh may be too coarse to produce satisfactory results in a region of interest, such as a stress concentration region in a stress analysis (see Figure 7.1: "Submodeling of a Pulley" (a). The results away from this region, however, may be adequate.

To obtain more accurate results in such a region, you have two options: (a) reanalyze the entire model with greater mesh refinement, or (b) generate an independent, more finely meshed model of only the region of interest and analyze it. Obviously, option (a) can be time-consuming and costly (depending on the size of the overall model). Option (b) is the submodeling technique (see Figure 7.1: "Submodeling of a Pulley" (b).

Figure 7.1 Submodeling of a Pulley


Stresses near this fillet are not accurate due to the coarse mesh and the high stress gradient
Y
Z X
Stresses away from the fillet may be OK
(a)
Submodel
Y
Z X
(b)

Submodeling of a pulley hub and spokes: (a) coarse model, (b) submodel shown superimposed over coarse model

Submodeling is also known as the cut-boundary displacement method or the specified boundary displacement method. The cut boundary is the boundary of the submodel which represents a cut through the coarse model. Displacements calculated on the cut boundary of the coarse model are specified as boundary conditions for the submodel.

Submodeling is based on St. Venant's principle, which states that if an actual distribution of forces is replaced by a statically equivalent system, the distribution of stress and strain is altered only near the regions of load application. The principle implies that stress concentration effects are localized around the concentration; therefore, if the boundaries of the submodel are far enough away from the stress concentration, reasonably accurate results can be calculated in the submodel.

The ANSYS program does not restrict submodeling to structural (stress) analyses only. Submodeling can be used effectively in other disciplines as well. For example, in a magnetic field analysis, you can use submodeling to calculate more accurate magnetic forces in a region of interest.

Aside from the obvious benefit of giving you more accurate results in a region of your model, the submodeling technique has other advantages:

- It reduces, or even eliminates, the need for complicated transition regions in solid finite element models.
- It enables you to experiment with different designs for the region of interest (different fillet radii, for example).
- It helps you in demonstrating the adequacy of mesh refinements.

Some restrictions for the use of submodeling are:

- It is valid only for solid elements and shell elements.
- The principle behind submodeling assumes that the cut boundaries are far enough away from the stress concentration region. It is up to you to verify that this assumption is adequately satisfied.

7.1. Employing Submodeling

The process for using submodeling is as follows:

1. Create and analyze the coarse model.
2. Create the submodel.
3. Perform cut boundary interpolation.
4. Analyze the submodel.
5. Verify that the distance between the cut boundaries and the stress concentration is adequate.

The steps are explained in detail next.

7.1.1. Create and Analyze the Coarse Model

The first step is to model the entire structure and analyze it.

> *Note* — To easily identify this initial model, we will refer to it as the coarse model. This does not mean that the mesh refinement has to be coarse, only that it is relatively coarse compared to the submodel.

The analysis type may be static (steady-state) or transient and follows the same procedure as described in the individual analysis guides. Some additional points to keep in mind are listed below.

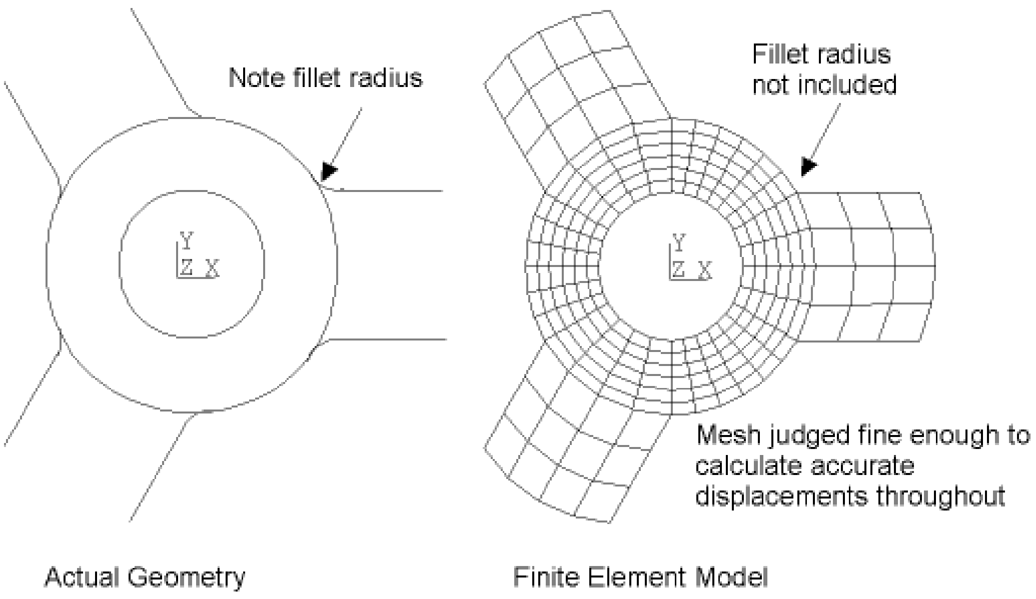
Jobname - You should use different jobnames for the coarse model and the submodel. This way, you can keep files from being overwritten. Also, you can easily refer to files from the coarse model during cut boundary interpolation. To specify a jobname, use one of these methods:
Command(s): /FILNAME
GUI: Utility Menu> File> Change Jobname

Element Types -- Only solid and shell elements support the submodeling technique. Your analysis may include other element types (such as beams added as stiffeners), but the cut boundary should only pass through the solids or shells.

A special submodeling technique called *shell-to-solid* submodeling allows you to build your coarse model with shell elements and your submodel with 3-D solid elements. This technique is discussed in Section 7.3: Shell-to-Solid Submodels.

Modeling -- In many cases, the coarse model need not include local details such as fillet radii, as shown in the following figure. However, the finite element mesh must be fine enough to produce a reasonably accurate degree of freedom solution. This is important because the results of the submodel are almost entirely based on interpolated degree of freedom results at the cut boundary.

Figure 7.2 Coarse Model


Note fillet radius
Fillet radius not included
Y
Z X
Y
Z X
Mesh judged fine enough to calculate accurate displacements throughout
Actual Geometry
Finite Element Model

Initial, coarse model may not need to include many details

Files - Both the results file (**Jobname.RST**, **Jobname.RMG**, etc.) and the database file (**Jobname.DB**, containing the model geometry) are required from the coarse-model analysis. Be sure to save the database before going on to create the submodel. To save the database, use one of these methods:

Command(s): SAVE
GUI: Utility Menu> File> Save as
Utility Menu> File> Save as Jobname.db

7.1.2. Create the Submodel

The submodel is completely independent of the coarse model. Therefore, the first step after the initial analysis is to clear the database at the Begin level. (Another way is to leave and re-enter the ANSYS program.) To clear the database at the Begin level, use one of these methods:

Command(s): /CLEAR
GUI: Utility Menu> File> Clear & Start New

Also, be sure to use a different jobname for the submodel so that the coarse-model files are not overwritten. To specify a jobname, use one of these methods:

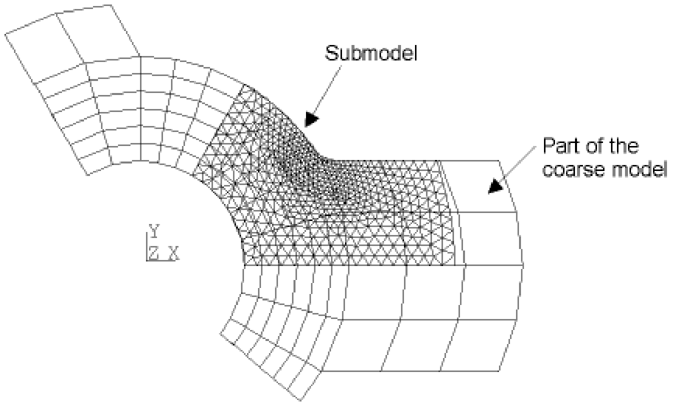
Command(s): /FILNAME
GUI: Utility Menu> File> Change Jobname

Then enter PREP7 and build the submodel. Some points to remember are:

- Use the same element type (solid or shell) that was used in the coarse model. Also, specify the same element real constants (such as shell thickness) and material properties. (Another type of submodeling - shell-to-solid submodeling - allows you to switch from shell elements in the coarse model to 3-D solid elements in the submodel; see Figure 7.9: "3-D Solid Submodel Superimposed on Coarse Shell Model".)

- The location of the submodel (with respect to the global origin) must be the same as the corresponding portion of the coarse model, as shown in Figure 7.3: "Submodel Superimposed Over Coarse Model".

Figure 7.3 Submodel Superimposed Over Coarse Model



- Specify appropriate node rotations. Node rotation angles on cut boundary nodes should *not* be changed after they have been written to the node file in interpolation step 1 (see Section 7.1.3: Perform Cut-Boundary Interpolation). To specify node rotations, use one of these methods:
  Command(s): NROTAT
  GUI: Main Menu> Preprocessor> Modeling> Create> Nodes> Rotate Node CS> To Active CS
  Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> To Active CS

Be aware that node rotation angles might be changed by application of nodal constraints [**DSYM**], by transfer of line constraints [**SFL**], or by transfer of area constraints [**SFA**], as well as by more obvious methods [**NROTAT** and **NMODIF**].

The presence or absence of node rotation angles in the coarse model has no effect upon the submodel.

Loads and boundary conditions for the submodel will be covered in the next two steps.

7.1.3. Perform Cut-Boundary Interpolation

This is the key step in submodeling. You identify the nodes along the cut boundaries, and the ANSYS program calculates the DOF values (displacements, potentials, etc.) at those nodes by interpolating results from the full (coarse) model. For each node of the submodel along the cut boundary, the ANSYS program uses the appropriate element from the coarse mesh to determine the DOF values. These values are then interpolated onto the cut boundary nodes using the element shape functions.

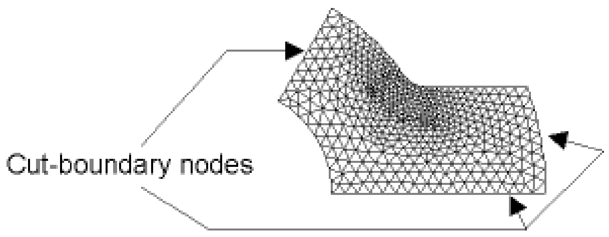
The following tasks are involved in performing the cut boundary interpolation:

1. Identify and write the cut-boundary nodes of the submodel to a file (**Jobname.NODE** by default). You can do this in PREP7 by selecting nodes along the cut boundaries and then using one of these methods to write the nodes to a file:
   Command(s): NWRITE
   GUI: Main Menu> Preprocessor> Modeling> Create> Nodes> Write Node File

   Here is an example of issuing the **NWRITE** command:

```
NSEL,...          ! Select nodes along cut boundaries
NWRITE            ! Writes all selected nodes to Jobname.NODE
```

Figure 7.4 Cut Boundaries on the Submodel



At this point, it is worthwhile to discuss *temperature interpolation*. In an analysis with temperature-dependent material properties, or in a thermal-stress analysis, the temperature distribution must be the same between the coarse model and the submodel. For such cases, you must also interpolate the temperatures from the coarse model to *all* nodes in the submodel. To do this, select *all* submodel nodes and write them to a different file using **NWRITE**,`Filename`,`Ext`. Be sure to specify a file name; otherwise, your file of cut boundary nodes will be overwritten! Step 7 shows the command to do temperature interpolation.

2. Restore the full set of nodes, write the database to **Jobname.DB**, and leave PREP7. You must write the database to **Jobname.DB** because you need to continue with the submodel later.

   To restore the full set of nodes, use one of these methods:
   Command(s): ALLSEL
   GUI: Utility Menu> Select> Everything

   To write the database to **Jobname.DB**, use one of these methods:
   Command(s): SAVE
   GUI: Utility Menu> File> Save as Jobname.db

3. To do the cut boundary interpolation (and the temperature interpolation), the database must contain the geometry for the coarse model. Therefore, you must resume the database using one of the methods shown below, making sure to identify the name of the coarse-model database file:
   Command(s): RESUME
   GUI: Utility Menu> File> Resume from

   For example, if the jobname for the coarse-model analysis was **COARSE**, issue the command **RESUME**,COARSE,DB.

4. Enter POST1, which is the general postprocessor (**/POST1** or menu path **Main Menu> General Postproc**). Interpolation can only be performed in POST1.

5. Point to the coarse results file (**FILE** or menu path **Main Menu> General Postproc> Data & File Opts**).

6. Read in the desired set of data from the results file (**SET** or menu path **Main Menu> General Postproc> Read Results>** ***`option`***).

7. Initiate cut-boundary interpolation. To do so, use one of these methods:
   Command(s): CBDOF
   GUI: Main Menu> General Postproc> Submodeling> Interpolate DOF

   By default, the **CBDOF** command assumes that the cut boundary nodes are on file **Jobname.NODE**. The ANSYS program will then calculate the cut boundary DOF values and write them in the form of **D** commands to the file **Jobname.CBDO**.

   To do temperature interpolation, use one of these methods, being sure to identify the name of the file containing *all* submodel nodes:
   Command(s): BFINT

GUI: Main Menu> General Postproc> Submodeling> Interp Body Forc

Interpolated temperatures are written in the form of **BF** commands to the file **Jobname.BFIN**.

Note — If real and imaginary data are involved, steps 6 and 7 will need to be done twice. First issue the **SET** command to get the real data, followed by the interpolation step [**CBDOF** and/or **BFINT**]. Then issue the **SET** command with the field set to 1 to get the imaginary data, and repeat the interpolation step, this time writing the interpolated imaginary data to a different file.

8. All interpolation work is now done, so leave POST1 [**FINISH**] and restore the submodel database (**RESUME** or menu path **Utility Menu> File> Resume from**). (Be sure to use the submodel database jobname.)

7.1.4. Analyze the Submodel

In this step, you define the analysis type and analysis options, apply the interpolated DOF values (and temperatures), define other loads and boundary conditions, specify load step options, and obtain the submodel solution.

The first step is to enter SOLUTION (**/SOLU** or menu path **Main Menu> Solution**).

Then define the appropriate analysis type (usually static) and analysis options.

To apply the cut boundary DOF constraints, simply read in the file of **D** commands (created by **CBDOF**) using one of these methods (for example, **/INPUT**,,CBDO):
Command(s): /INPUT
GUI: Utility Menu> File> Read Input from

Similarly, to apply the interpolated temperatures, read in the file of **BF** commands (created by **BFINT**) using one of these methods (for example, **/INPUT**,,BFIN):
Command(s): /INPUT
GUI: Utility Menu> File> Read Input from

If real and imaginary data are involved, first read in the file(s) containing the real data, specify whether DOF constraint values and/or nodal body force loads are to be accumulated, and then read in the file containing the imaginary data.

Specify that DOF constraint values are to be accumulated:
Command(s): DCUM,ADD
GUI: Main Menu> Preprocessor> Loads> Define Loads> Settings> Replace vs Add> Constraints
Main Menu> Solution> Define Loads> Settings> Replace vs Add> Constraints

Specify that nodal body force loads are to be accumulated:
Command(s): BFCUM,,ADD
GUI: Main Menu> Preprocessor> Loads> Define Loads> Settings> Replace vs Add> Nodal Body Ld
Main Menu> Solution> Define Loads> Settings> Replace vs Add> Nodal Body Ld

Be sure to reset the **DCUM** and **BFCUM** commands to their default status before proceeding.

It is important that you duplicate on the submodel any other loads and boundary conditions that existed on the coarse model. Examples are symmetry boundary conditions, surface loads, inertia forces (such as gravity), concentrated force loads, etc. (see Figure 7.5: "Loads on the Submodel").

Then specify load step options (such as output controls) and initiate solution calculations using one of these methods:

Command(s): SOLVE
GUI: Main Menu> Solution> Solve> Current LS
Main Menu> Solution> Run FLOTRAN

After the solution is obtained, leave SOLUTION [**FINISH**].

The overall data flow for submodeling (without temperature interpolation) is shown in Figure 7.6: “Data Flow Diagram for Submodeling (Without Temperature Interpolation)”.

Figure 7.5 Loads on the Submodel

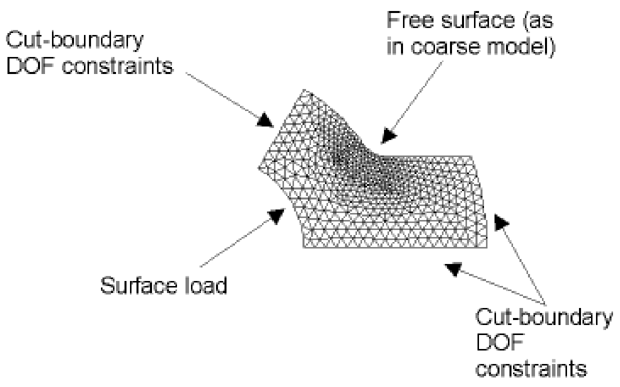
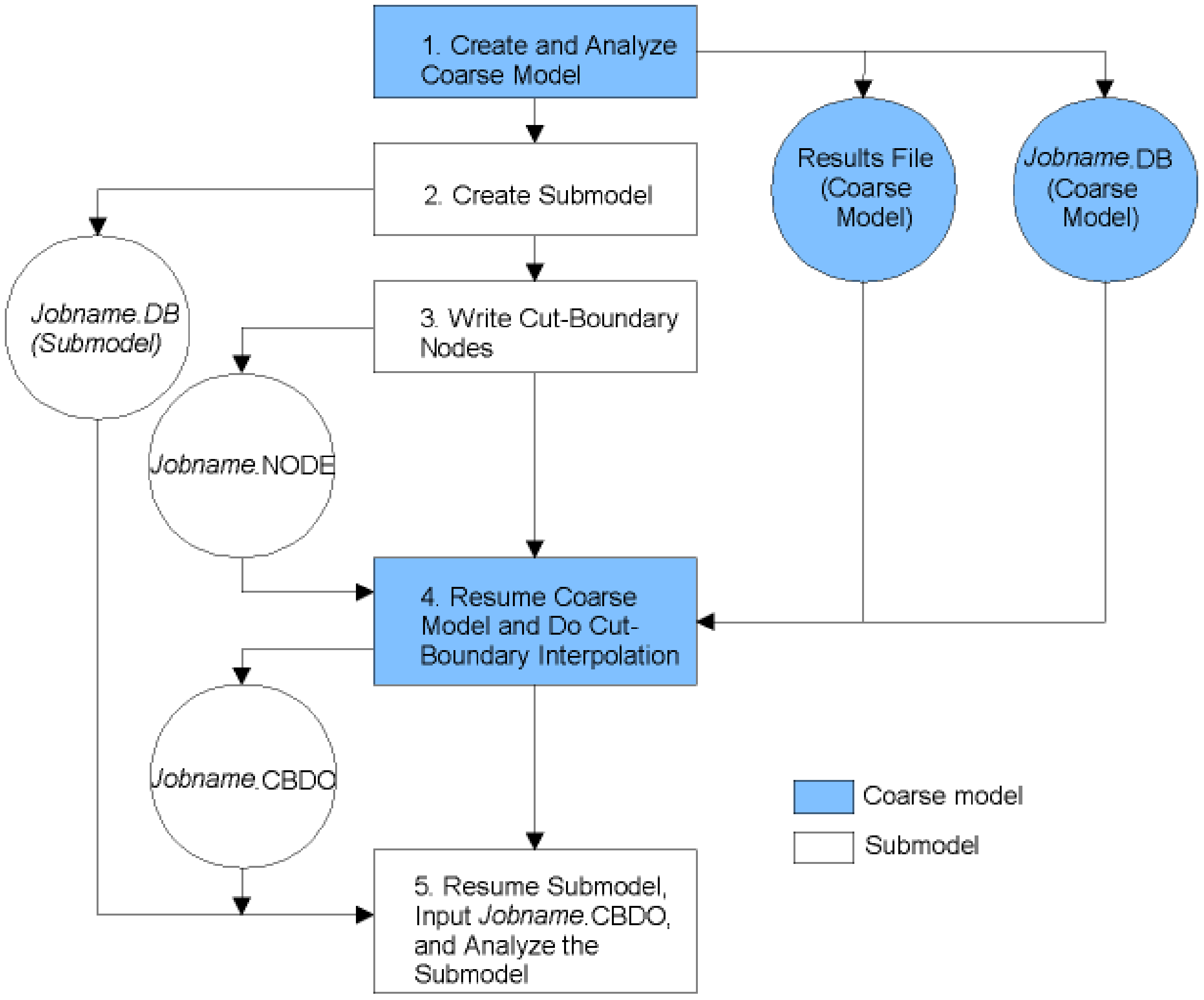

Cut-boundary DOF constraints
Free surface (as in coarse model)
Surface load
Cut-boundary DOF constraints

Figure 7.6 Data Flow Diagram for Submodeling (Without Temperature Interpolation)


1. Create and Analyze Coarse Model
Results File (Coarse Model)
Jobname.DB (Coarse Model)
2. Create Submodel
Jobname.DB (Submodel)
3. Write Cut-Boundary Nodes
Jobname.NODE
4. Resume Coarse Model and Do Cut-Boundary Interpolation
Jobname.CBDO
Coarse model
Submodel
5. Resume Submodel, Input *Jobname*.CBDO, and Analyze the Submodel

7.1.5. Verify the Distance Between the Cut Boundaries and the Stress Concentration

The final step is to verify that the cut boundaries of the submodel are far enough away from the concentration. You can do this by comparing results (stresses, magnetic flux density, etc.) along the cut boundaries with those along the corresponding locations of the coarse model. If the results are in good agreement, it indicates that proper cut boundaries have been chosen. Otherwise, you will need to recreate and reanalyze the submodel with different cut boundaries further away from the region of interest.

An effective way to compare results is to obtain contour displays and path plots, as shown in Figure 7.7: “Contour Plots to Compare Results” and Figure 7.8: “Path Plots to Compare Results”.

Figure 7.7 Contour Plots to Compare Results

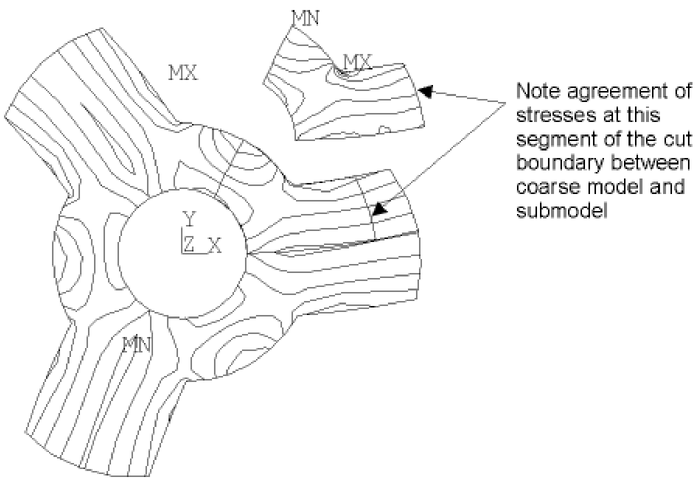
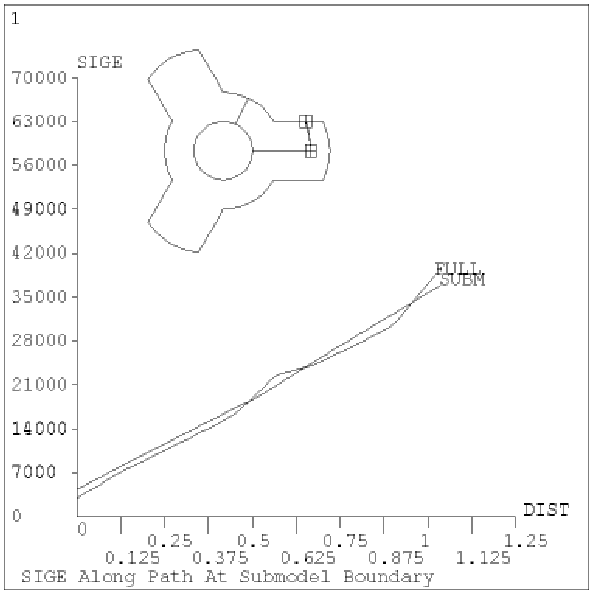

MN
MX
MX
Note agreement of stresses at this segment of the cut boundary between coarse model and submodel
Y
Z X
MN

Figure 7.8 Path Plots to Compare Results



7.2. Sample Analysis Input

A sample input listing for a submodeling analysis is shown below:

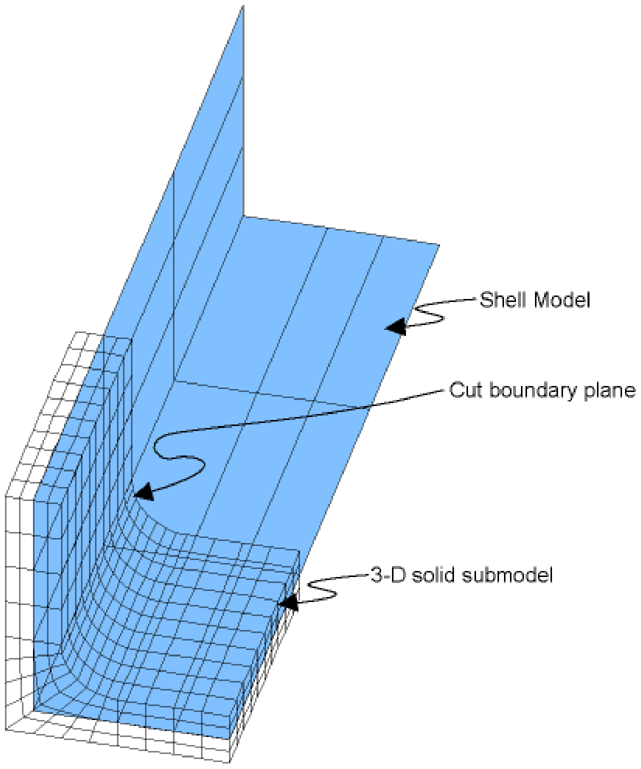
```
!  Start with coarse model analysis:
/FILNAME,coarse       ! Jobname = coarse
/PREP7                ! Enter PREP7
...
...                   ! Generate coarse model
...
FINISH

/SOLU                 ! Enter SOLUTION
ANTYPE,...            ! Analysis type and analysis options
...
D,...                 ! Loads and load step options
DSYMM,...
ACEL,...
...
SAVE                  ! Coarse model database file coarse.db
SOLVE                 ! Coarse model solution
```

```
                      ! Results are on coarse.rst (or rmg, etc.)
FINISH

!  Create submodel:
/CLEAR                ! Clear the database (or exit ANSYS and re-enter)
/FILNAME,submod       ! New jobname = submod
/PREP7                ! Re-enter PREP7
...
...                   ! Generate submodel
...

!  Perform cut boundary interpolation:
NSEL,...              ! Select nodes on cut boundaries
NWRITE                ! Write those nodes to submod.node
ALLSEL                ! Restore full sets of all entities
NWRITE,temps,node     ! Write all nodes to temps.node (for
                      !   temperature interpolation)
SAVE                  ! Submodel database file submod.db
FINISH

RESUME,coarse,db      ! Resume coarse model database (coarse.db)
/POST1                ! Enter POST1
FILE,coarse,rst       ! Use coarse model results file
SET,...               ! Read in desired results data
CBDOF                 ! Reads cut boundary nodes from submod.node and
                      !   writes D commands to submod.cbdo
BFINT,temps,node      ! Reads all submodel nodes from temps.node and
                      !   writes BF commands to submod.bfin (for
                      !   temperature interpolation)
FINISH                ! End of interpolation

RESUME                ! Resume submodel database (submod.db)
/SOLU                 ! Enter SOLUTION
ANTYPE,...            ! Analysis type and options
...
/INPUT,submod,cbdo    ! Cut boundary DOF specifications
/INPUT,submod,bfin    ! Interpolated temperature specifications
DSYMM,...             ! Other loads and load step options
ACEL,...
...
SOLVE                 ! Submodel solution
FINISH

/POST1                ! Enter POST1
...
...                   ! Verify submodel results
...
FINISH
```

7.3. Shell-to-Solid Submodels

In the shell-to-solid submodeling technique, the coarse model is a shell model, and the submodel is a 3-D solid model. A sample is shown in Figure 7.9: “3-D Solid Submodel Superimposed on Coarse Shell Model”.

Figure 7.9 3-D Solid Submodel Superimposed on Coarse Shell Model



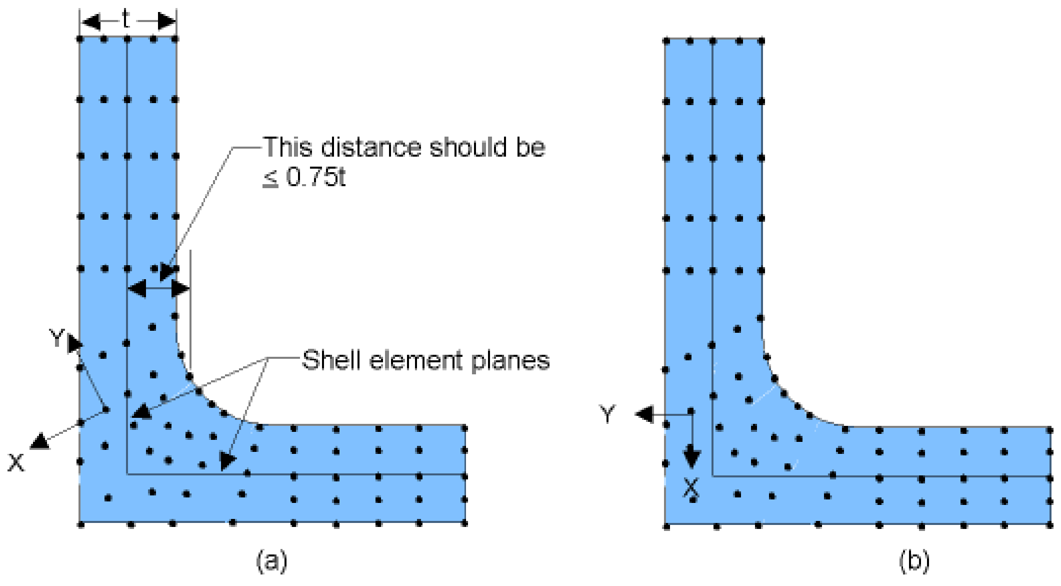
The procedure for shell-to-solid submodeling is essentially the same as that for solid-to-solid submodeling, with these exceptions:

- Shell-to-solid submodeling is activated by setting *KSHS* to 1 on the **CBDOF** command (**Main Menu> General Postproc> Submodeling> Interpolate DOF**) and the **BFINT** command (**Main Menu> General Postproc> Submodeling> Interp Body Forc**). This feature is not applicable to offsets used with SHELL91 (KEYOPT(11) $\neq$ 0), SHELL99 (KEYOPT(11) $\neq$ 0), or SHELL181 (SECOFFSET).
- Cut boundaries on the submodel are the end planes that are normal to the shell plane (see Figure 7.10: “Node Rotations”). Nodes on these cut boundaries are written to the node file [**NWRITE**] (**Main Menu> Preprocessor> Modeling> Create> Nodes> Write Node File**).
- To determine the DOF values at a cut boundary node [**CBDOF**], the program first projects the node onto the nearest element in the shell plane. The DOF values of this projected point are then calculated by interpolation and assigned to the corresponding node. Interpolated temperatures [**BFINT**] are calculated based on the average temperature at the midplane of the nearest shell element.

  Note — The nodes on the cut boundary must lie within a distance of 0.75 times the average thickness of the nearest shell element, as shown in Figure 7.10: “Node Rotations”. That is, the submodel should be approximately centered on the coarse model.

- In a structural analysis, only translational displacements are calculated for the cut boundary nodes, but their values are based on both the translations *and rotations* of the projected point. Also, the node is rotated such that the nodal UY direction is always perpendicular to the shell plane, as shown in Figure 7.10: “Node Rotations”. A UY constraint will be calculated only for nodes that are within 10% of the average shell element thickness from the shell plane. This prevents over-constraint of the submodel in the transverse direction.

Figure 7.10 Node Rotations



Node rotations: (a) before **CBDOF** command, (b) after **CBDOF** command

- The **.CBDO** file written by the **CBDOF** command will consist of two blocks:
  - a block of **NMODIF** commands (indicating node rotation angles) and **DDELE** commands (to delete UY constraints)
  - a block of **D** commands (to apply the interpolated DOF values).

The two blocks are separated by a **/EOF** command and a :CB*nn* label (where *nn* is the cumulative iteration number of the results set used).

- You must read in the **.CBDO** file in PREP7, because the **NMODIF** command is only valid in PREP7. To do so, enter the preprocessor, then use one of these methods:

  Command(s): /INPUT
  GUI: Utility Menu> File> Read Input from

Also, you will have to read in the **.CBDO** file twice, because the two blocks of commands are separated by a **/EOF** command. The second time you read in the file, use the *LINE* field on **/INPUT** ("Optional line number or label" in the GUI) to instruct the program to read the file starting with the :CB*nn* label, as shown below:

```
/PREP7! The .CBDO file must be read in PREP7
/INPUT,,cbdo        ! Reads Jobname.cbdo up to the /EOF command
/INPUT,,cbdo,,:cb1  ! Reads same file from the label :cb1
```

7.4. Where to Find Examples

The *ANSYS Verification Manual* consists of test case analyses demonstrating the analysis capabilities of the ANSYS program. While these test cases demonstrate solutions to realistic analysis problems, the *ANSYS Verification Manual* does not present them as step-by-step examples with lengthy data input instructions and printouts. However, most ANSYS users who have at least limited finite element experience should be able to fill in the missing details by reviewing each test case's finite element model and input data with accompanying comments.

The *ANSYS Verification Manual* contains the following submodeling test case:

VM142 - Stress Concentration at a Hole in a Plate

Chapter 8: Substructuring

Substructuring is a procedure that condenses a group of finite elements into one element represented as a matrix. The single-matrix element is called a *superelement*. You can use a superelement in an analysis as you would any other ANSYS element type. The only difference is that you first create the superelement by performing a substructure generation analysis. Substructuring is available in the ANSYS Multiphysics, the ANSYS Mechanical, and the ANSYS Structural products.

Substructuring reduces computer time and allows solution of very large problems with limited computer resources. Nonlinear analyses and analyses of structures containing repeated geometrical patterns are typical candidates for employing substructuring. In a nonlinear analysis, you can substructure the linear portion of the model so that the element matrices for that portion need not be recalculated at every equilibrium iteration. In a structure with repeated patterns (such as the four legs of a table), you can generate one superelement to represent the pattern and simply make copies of it at different locations, thereby saving a significant amount of computer time.

You can also use substructuring on models with large rotations. For these models, ANSYS assumes each substructure to rotate about its mass center. In 3-D cases, the substructures contain three rigid body rotations and three translational motions. With a large rotation model, you do not constrain the substructure until the use pass because each substructure is treated as a single finite element that should allow rigid body motions.

An example of reason (b) is an analysis that is too large for the computer in terms of wavefront size or disk space requirements. In such a situation, you can analyze the model in pieces, where each piece is a superelement small enough to fit on the computer.

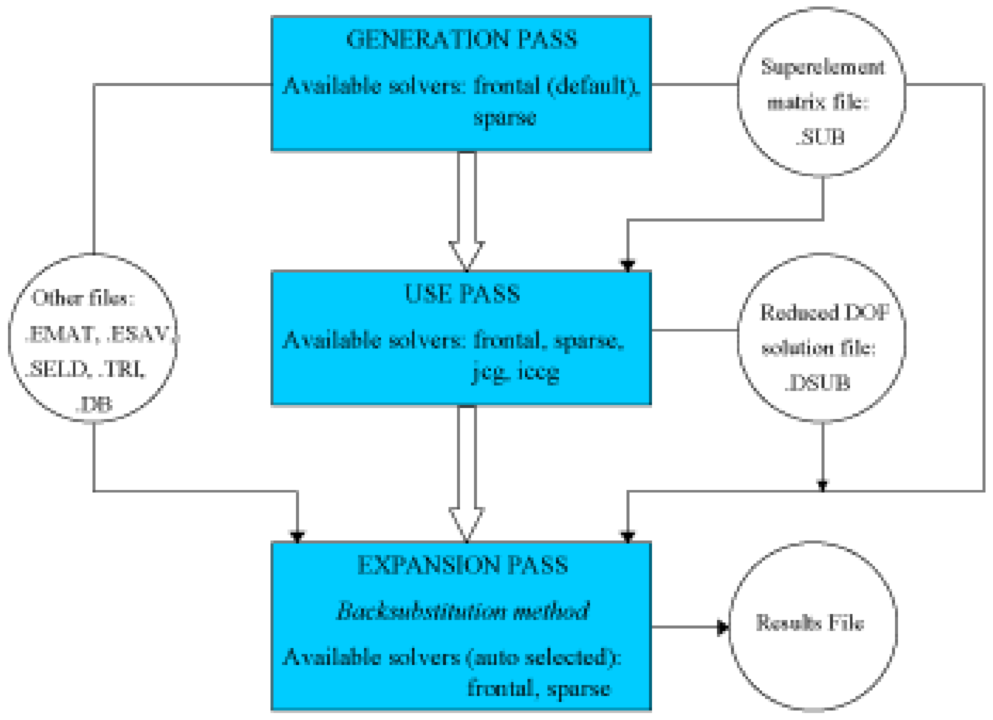
8.1. Employing Substructuring

A substructure analysis involves three distinct steps, called *passes*:

1. Generation pass
2. Use pass
3. Expansion pass.

Figure 8.1: "Applicable Solvers in a Typical Substructuring Analysis" shows the data flow for a complete substructure analysis and some of the files involved.

Figure 8.1 Applicable Solvers in a Typical Substructuring Analysis


GENERATION PASS
Available solvers: frontal (default), sparse
Superelement matrix file: .SUB
USE PASS
Available solvers: frontal, sparse, jcg, iccg
Other files: .EMAT, .ESAV, .SELD, .TRI, .DB
Reduced DOF solution file: .DSUB
EXPANSION PASS
Backsubstitution method
Available solvers (auto selected): frontal, sparse
Results File

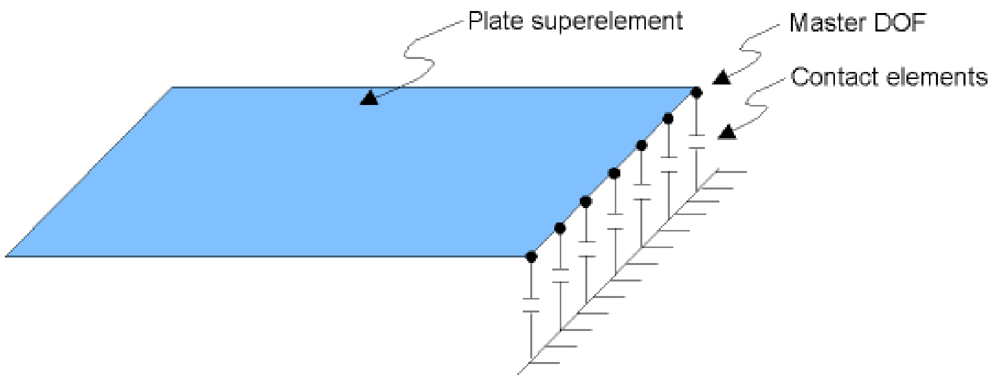
The three passes are explained in detail next.

> *Note* — Perform each step while using the same version of ANSYS. Do not go from one version of ANSYS to another while performing these steps.

8.1.1. Generation Pass: Creating the Superelement

The *generation pass* is where you condense a group of "regular" finite elements into a single superelement. The condensation is done by identifying a set of master degrees of freedom, used mainly to define the interface between the superelement and other elements and to capture dynamic characteristics for dynamic analyses. Figure 8.2: “Example of a Substructuring Application” shows a plate-like structure that is to be analyzed with contact (interface) elements. Since the contact elements require an iterative solution, substructuring the plate portion can result in a significant savings in computer time. The master DOF required in this case are the degrees of freedom that connect the plate to the contact elements.

Figure 8.2 Example of a Substructuring Application



The procedure to generate a superelement consists of two main steps:

1. Build the model.
2. Apply loads and create the superelement matrices.

8.1.1.1. Building the Model

In this step, you specify the jobname and analysis title and then use PREP7 to define the element types, element real constants, material properties, and the model geometry. These tasks are common to most analyses and are described in the *ANSYS Basic Analysis Guide*. For the generation pass, you should keep in mind the following additional points:

Jobname-This takes on *special significance* in a substructure analysis. By using jobnames effectively, you can eliminate much of the file handling inherent in a three-pass analysis.

Use one of these methods to specify the jobname:
Command(s): /FILNAME
GUI: Utility Menu> File> Change Jobname

For example,

```
/FILNAME,GEN
```

gives the jobname **GEN** to all files produced by the generation pass. The default jobname is **FILE** (or **file**) or whatever name was specified while entering the ANSYS program.

Element Types - Most ANSYS element types can be used to generate a substructure. In general, the only restriction is that elements within the superelement are assumed to be linear and cannot use Lagrange multipliers. If you include bilinear elements, they will be treated as linear elements (in their initial state).

Caution: Coupled-field elements used in a direct method coupled-field analysis with load vector coupling are not valid in a substructure analysis. Other elements in the same shape family should be used instead. See the *ANSYS Coupled-Field Analysis Guide* for details. In addition, elements with Lagrange multipliers cannot be used in substructuring. These type of elements include MPC184, CONTA171, CONTA172, CONTA173, CONTA174, CONTA175, and CONTA178 with appropriate KEYOPT(2) setting, and elements PLANE182, PLANE 183, SOLID185, SOLID186 and SOLID187 when using KEYOPT(6) > 0.

Material Properties - Define all necessary material properties. For example, if the mass matrix is to be generated, density (DENS) (or mass in some form) must be defined; if the specific heat matrix is to be generated, the specific heat (C) must be defined; and so on. Again, because a superelement is linear, any nonlinear material properties will be ignored.

Model Generation - In the generation pass, you are mainly concerned with creating the superelement portion of the model. The nonsuperelement portion, if any, is defined later in the use pass. However, you should plan the modeling approach for both portions before you start building the model. In particular, decide on how you want to connect the superelement to the other elements. To ensure the connection, use the same node numbers at the interface. (Other methods requiring less effort on your part are discussed in the use pass section later in this chapter.)

One approach might be to develop the *entire* model, save it on a named database file, and select only the portion to be substructured for the generation pass. In the use pass then, you can **RESUME** (**Utility Menu> File> Resume from**) from the named database file, unselect the portion that was substructured and replace it with the superelement matrix.

8.1.1.2. Applying Loads and Creating the Superelement Matrices

The "solution" from a substructure generation pass consists of the superelement matrix (or matrices). As with any other analysis, you define the analysis type and options, apply loads, specify load step options, and initiate the solution. Details of how to do these tasks are explained below.

Enter SOLUTION using either of these methods

> **Command(s): /SOLU**
> **GUI: Main Menu> Solution**

Define the analysis type and analysis options The applicable options are explained below.

Analysis Type - Choose a substructure generation using one of these methods:

> **Command(s): ANTYPE**
> **GUI: Main Menu> Solution> Analysis Type> New Analysis**

New analysis or restart - If you are starting a new analysis, choosing the analysis type (as described above) is all you need to do. However, if the run is a restart, you must also indicate this by setting *STATUS* = REST on the **ANTYPE** command (**Main Menu> Solution> Analysis Type> Restart**). A restart is applicable if you need to generate additional load vectors. (The files **Jobname.EMAT**, **Jobname.ESAV**, and **Jobname.DB** from the initial run must be available for the restart.)

> *Note* — Restarting a substructuring analysis is valid only if the backsubstitution method is chosen. You cannot restart a run if the full resolve option is selected using the **SEOPT** command.

Name of the superelement matrix file - Specify the name (`Sename`) to be assigned to the superelement matrix file. The program will automatically assign the extension SUB, so the complete file name will be ***Sename*.SUB**. The default is to use the jobname [**/FILNAME**]. To specify the name of the superelement matrix file:

> **Command(s): SEOPT**
> **GUI: Main Menu> Solution> Analysis Type> Analysis Options**

Equation Solver to Use - Specify which equation solver to use (FRONTAL (default) or SPARSE (recommended)) during the generation pass of the substructure analysis. To specify an equation solver:

> **Command(s): EQSLV**
> **GUI: Main Menu> Solution> Analysis Type> Analysis Options**

Matrices to be generated - You can request generation of just the stiffness matrix (or conductivity matrix, magnetic coefficient matrix, etc.); stiffness and mass matrices (or specific heat, etc.); or stiffness, mass, and damping matrices. The mass matrix is required if the use pass is a structural dynamic analysis or if you need to apply inertia loads in the use pass. For the thermal case, the specific heat matrix is required only if the use pass is a transient thermal analysis. Similar considerations apply to other disciplines and to the damping matrix. To make your request, use the **SEOPT** command as described above.

> *Note* — The sparse solver cannot be used to generate the damping matrix.

Matrices to be printed - This option allows you to print out the superelement matrices. You can request listing of both matrices and load vectors, or just the load vectors. The default is not to print any matrices. To print out the matrices, use the **SEOPT** command:

Expansion Pass Method - Allows you to select the expansion pass method you plan to use during subsequent expansion passes with this superelement. The backsubstitution method (default) saves the triangularized matrix files needed to perform a backsubstitution of the master DOF solution during the expansion pass. The full resolve method does not save any triangularized matrix files. Triangularized matrix files are **Sename.tri** for the frontal solver and **Sename.LNxx** for the sparse solver.

Note — Triangularized matrix files can become very large as the problem size increases, but are not needed if the full resolve method is chosen during the expansion pass.

During the expansion pass, the full resolve method reforms the elements used to create the superelement, reassembles the global stiffness matrix, and applies the master DOF solution as displacement boundary conditions internally.

Note — You cannot restart a substructure analysis with the full resolve expansion pass method chosen.

Mass matrix formulation - Applicable only if you want the mass matrix to be generated. You can choose between the default formulation (which depends on the element type) and a lumped mass approximation. We recommend the default formulation for most applications. However, for dynamic analyses involving "skinny" structures, such as slender beams or very thin shells, the lumped mass approximation has been shown to give better results. To specify a lumped mass approximation, use one of these methods:

Command(s): LUMPM
GUI: Main Menu> Solution> Analysis Type> Analysis Options

Define master degrees of freedom using one of these methods
Command(s): M
GUI: Main Menu> Solution> Master DOFs> User Selected> Define

In a substructure, master DOFs serve four purposes:

- They serve as the interface between the superelement and other elements. Be sure to define master DOFs at all nodes that connect to nonsuperelements, as shown in Figure 8.2: “Example of a Substructuring Application”. *All* degrees of freedom at these nodes should be defined as master DOFs (*Lab* = ALL on the **M** command). Master DOFs must be defined even if you plan to have no elements in the model other than a superelement.
- If the superelement is to be used in a dynamic analysis, master DOFs characterize the dynamic behavior of the structure. See Chapter 3, “Modal Analysis” in the *ANSYS Structural Analysis Guide* for guidelines.
- If constraints [**D**] or force loads [**F**] are to be applied in the use pass, master DOFs must be defined at those locations with the **M** command.
- Master DOFs are required in the use pass for large deflections [**NLGEOM**,ON] (**Main Menu> Solution> Analysis Type> Analysis Options**) or when used with the **SETRAN** command (**Main Menu> Preprocessor> Modeling> Create> Elements> Superelements> By CS Transfer**). For these cases, all nodes that have master DOFs must have all six (UX, UY, UZ, ROTX, ROTY, ROTZ) defined.

Apply loads on the model You can apply all types of loads in a substructure generation pass (see Table 8.1: “Loads Applicable in a Substructure Analysis”), but keep in mind the following points:

- The program will generate a load vector that includes the effect of all applied loads. One load vector per load step is written to the superelement matrix file. This load vector is the equivalent load for the combined loads in the load step. A maximum of 31 load vectors are allowed.
- Nonzero DOF constraints can be used in the generation pass and will become part of the load vector. (In the expansion pass, if the load step being expanded contains nonzero DOF constraints, the database must have matching DOF values. If it does not, the DOF constraints must be respecified [**D**] in the expansion pass.)
- Application of constraints [**D**] or force loads [**F**] can be postponed until the use pass, but master DOF must be defined at those locations with the **M** command or corresponding GUI path.

- Similarly, application of linear and angular accelerations can be postponed until the use pass, but only if a mass matrix is generated. This is recommended if you plan to rotate the superelement in the use pass, because load vector directions are "frozen" and rotated with the superelement.
- The Maxwell force flag (MXWF label on the **SF** family of commands) is normally used in a magnetic analysis to flag element surfaces on which the magnetic force distribution is to be calculated. This flag has no effect (and therefore should not be used) for a superelement in a magnetic analysis.

Note — If a mass matrix is generated, it is recommended that degree of freedom constraints be applied in the use pass at the master DOF (defined in the generation pass). This ensures that all mass is accounted for in the substructure.

For large rotation analyses - Do not apply constraints to the model in this pass. You will apply constraints for large rotation analyses in the use pass.

Table 8.1 Loads Applicable in a Substructure Analysis

Load Name	Load Category	Commands[1]	
		Solid Model Loads	**Finite Element Loads**
Displacement Temperature Mag. Potential ...	Constraints	**DK**, **DKLIST**, **DKDELE**, **DL**, **DLLIST**, **DLDELE**, **DA**, **DALIST**, **DADELE**, **DTRAN**	**D**, **DSYM**, **DLIST**, **DDELE**, **DSCALE**, **DCUM**
Force Heat Flow Rate Mag. Flux ...	Forces	**FK**, **FKLIST**, **FKDELE**, **FTRAN**	**F**, **FLIST**, **FDELE**, **FSCALE**, **FCUM**
Pressure Convection Maxwell Surface ...	Surface Loads	**SFL**, **SFLLIST**, **SFLDELE**, **SFA**, **SFALIST**, **SFADELE**, **SFGRAD**, **SFTRAN**	**SF**, **SFLIST**, **SFDELE**, **SFE**, **SFELIST**, **SFEDELE**, **SFBEAM**, **SFGRAD**, **SFFUN**, **SFSCALE**, **SFCUM**
Temperature Heat Generation Rate Current Density ...	Body Loads	**BFK**, **BFKLIST**, **BFKDELE**, **BFL**, **BFLLIST**, **BFLDELE**, **BFA**, **BFALIST**, **BFADELE**, **BFV**, **BFVLIST**, **BFVDELE**, **BFTRAN**	**BF**, **BFLIST**, **BFDELE**, **BFE**, **BFELIST**, **BFEDELE**, **BFSCALE**, **BFCUM**
Gravity, Linear and Angular Acceleration	Inertia Loads		**ACEL**, **DOMEGA**

1. The menu path used to access each command in the GUI will vary depending on the engineering discipline of the analysis (structural, magnetic, etc.). For a list of menu paths, see the description of individual commands in the *ANSYS Commands Reference*.

Specify load step options The only options valid for the substructure generation pass are dynamics options (damping).

Damping (Dynamics Options) - Applicable only if the damping matrix is to be generated.

To specify damping in the form of alpha (mass) damping:

Command(s): ALPHAD
GUI: Main Menu> Solution> Load Step Opts> Time/Frequenc> Damping

To specify damping in the form of beta (stiffness) damping:

Command(s): BETAD
GUI: Main Menu> Solution> Load Step Opts> Time/Frequenc> Damping

To specify damping in the form of material-dependent beta damping:

Command(s): MP,DAMP
GUI: Main Menu> Preprocessor> Material Props> Material Models> Structural> Damping

Save a backup copy of the database on a named file Doing this is required because you need to work with the same database in the expansion pass. To save a backup copy, use one of these methods:
Command(s): SAVE
GUI: Utility Menu> File> Save as Jobname.db

Start solution calculations using one of these methods:
Command(s): SOLVE
GUI: Main Menu> Solution> Solve> Current LS Output from the solution consists of the superelement matrix file, ***Sename*.SUB**, where *Sename* is the name assigned as an analysis option [**SEOPT**] or the jobname [**/FILNAME**]. The matrix file includes a load vector calculated based on the applied loads. (The load vector will be zero if no loads are defined.)

Repeat for additional load steps (that is, to generate additional load vectors) The load vectors are numbered sequentially (starting from 1) and appended to the same superelement matrix file. See Chapter 2, "Loading" in the *ANSYS Basic Analysis Guide* for other methods for multiple load steps.

Leave SOLUTION using one of these methods
Command(s): FINISH
GUI: Main Menu> Finish

8.1.2. Use Pass: Using the Superelement

The *use pass* is where you *use* the superelement in an analysis by making it part of the model. The entire model may be a superelement or, as in the plate example, the superelement may be connected to other nonsuperelements (see Figure 8.2: "Example of a Substructuring Application"). The solution from the use pass consists only of the *reduced* solution for the superelement (that is, the degree of freedom solution only at the master DOF) and complete solution for nonsuperelements.

The use pass can involve any ANSYS analysis type (except a FLOTRAN or explicit dynamics analysis). The only difference is that one or more of the elements in the model is a superelement that has been previously generated. The individual analysis guides contain detailed descriptions about performing various analyses. In this section, we will concentrate on the steps you need to make the superelement a part of your model.

8.1.2.1. Clear the Database and Specify a New Jobname

The use pass consists of a new model and new loads. Therefore, the first step is to clear the existing database. This has the same effect as leaving and re-entering the ANSYS program. To clear the database, use one of these methods:
Command(s): /CLEAR
GUI: Utility Menu> File> Clear & Start New

By default, clearing the database causes the **START.ANS** file to be reread. (You can change this setting if you so desire.)

Caution: If you are using the command input method to clear the database, additional commands may not be stacked on the same line (using the $ delimiter) as the **/CLEAR** command.

Be sure to define a jobname that is different from the one used for the generation pass. This way, you can ensure that no generation pass files will be overwritten. To define a jobname, use one of these methods:
Command(s): /FILNAME

GUI: Utility Menu >File> Change Jobname

8.1.2.2. Build the Model

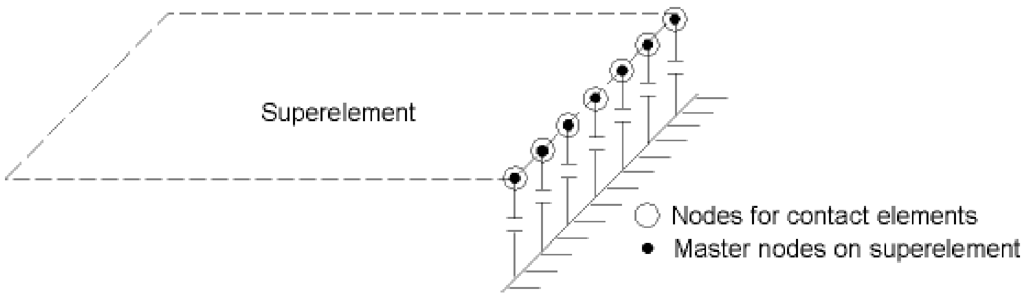
This step is performed in PREP7 and consists of the following tasks:

1. Define MATRIX50 (the superelement) as one of the element types. Use one of these methods:
   Command(s): ET
   GUI: Main Menu> Preprocessor> Element Type> Add/Edit/Delete
2. Define other element types for any nonsuperelements. Nonlinear behavior may or may not be allowed, depending on the type of analysis to be performed.
3. Define element real constants and material properties for the nonsuperelements. Nonlinear properties may or may not be allowed, depending on the type of analysis to be performed.
4. Define the geometry of the nonsuperelements. Take special care in defining the interfaces where the nonsuperelements connect to the superelements. *The interface node locations must exactly match the locations of the corresponding master nodes on the superelements* (see Figure 8.3: "Node Locations").

   There are three ways to ensure connectivity at these interfaces:

   - Use the same node *numbers* as the ones in the generation pass.
   - Use the same node number *increment* (or *offset*) between interface nodes in the generation pass and interface nodes in the use pass. (Use **SETRAN**, as described below in step 5b.)
   - *Couple* the two sets of nodes in all degrees of freedom using the **CP** family of commands [**CP**, **CPINTF**, etc.]. This method is helpful if you cannot use one of the above two methods. For example, to define a set of coupled degrees of freedom use one of the following:
     Command(s): CP
     GUI: Main Menu> Preprocessor> Coupling/Ceqn> Couple DOFs
     If the superelement is *not* connected to any other elements, you do not need to define any nodes in the use pass.

   Figure 8.3 Node Locations

   Interface nodes between superelement and nonsuperelement must exactly match the master node locations.
5. Define the superelement by pointing to the proper element type reference number and reading in the superelement matrix. To point to the element type:
   Command(s): TYPE
   GUI: Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes

   Now read in the superelement matrix using one of these methods (you may first need to use other commands to modify the matrix, as explained below):

Command(s): SE
GUI: Main Menu> Preprocessor> Modeling> Create> Elements> Superelements> From .SUB File

a. If there are no nonsuperelements in the model, or if there are nonsuperelements and the interface nodes have the exact same node numbers as the master nodes on the superelement, then simply read in the superelement using the **SE** command:

```
TYPE,...! Element type reference number
SE,GEN! Reads in superelement from file GEN.SUB
```

The `Sename` field on the **SE** command shown above identifies the name of the superelement matrix file. The extension **.SUB** is assumed, so the complete file name is ***Sename*.SUB** (**GEN.SUB** in the above example). The superelement is given the next available element number.

b. If there are nonsuperelements in the model and the interface nodes have a constant node number offset from the master nodes, you must first create a new superelement matrix with new node numbers and then read in the new matrix.

To create a new superelement matrix, use one of these methods:
Command(s): SETRAN
GUI: Main Menu> Preprocessor> Modeling> Create> Elements> Superelements> By CS Transfer
Main Menu> Preprocessor> Modeling> Create> Elements> Superelements> By Geom Offset

To read in the new matrix, use one of these methods:
Command(s): SE
GUI: Main Menu> Preprocessor> Modeling> Create> Elements> Superelements> From .SUB File

For example, given an existing superelement matrix file GEN.SUB and a node number offset of 2000, the commands would be:

```
SETRAN,GEN,,2000,GEN2,SUB   ! Creates new superelement GEN2.SUB with
                            !   node offset = 2000
TYPE,...                    ! Element type reference number
SE,GEN2                     ! Reads in new superelement from file GEN2.SUB
```

c. If there are nonsuperelements in the model and the interface nodes have no relationship with the master nodes (as would be the case with automatically meshed models), first observe the following caution.

Caution: It is quite likely that the node numbers of the master nodes from the generation pass overlap with node numbers in the use pass model. In such cases, reading in the superelement [**SE**] will cause existing use pass nodes to be overwritten by the superelement's master nodes. To avoid overwriting existing nodes, use a node number offset [**SETRAN**] before reading in the superelement. In any case, save the database [**SAVE**] before issuing the **SE** command.

Thus you should first save the database [**SAVE**], use the **SETRAN** command to create a new superelement matrix with a node number offset, and then use the **SE** command to read in the new matrix. The **CPINTF** command (**Main Menu> Preprocessor> Coupling/Ceqn> Coincident Nodes**) can then be used to connect the pairs of nodes at the interface. For example, given a superelement matrix file called GEN.SUB:

```
*GET,MAXNOD,NODE,,NUM,MAX    ! MAXNOD = maximum node number
SETRAN,GEN,,MAXNOD,GEN2,SUB  ! Creates new superelement with
                             !   node offset = MAXNOD, name = GEN2.SUB
SE,GEN2                      ! Reads in new superelement
NSEL,...                     ! Select all nodes at the interface
CPINTF,ALL                   ! Couples each pair of interface nodes in
                             ! all DOF
NSEL,ALL
```

d. If the superelement is to be transformed - moved or copied to a different position, or symmetrically reflected - you must first use the **SETRAN** command or **SESYMM** command (**Main Menu> Preprocessor> Modeling> Create> Elements> Superelements> By Reflection**), with the appropriate node number offsets, to create new superelement matrix files and then use **SE** to read in the new matrices. Connecting the superelements to the nonsuperelements is done the same way as above - by using common node numbers, a constant node number offset, or the **CPINTF** command.

Note — If you use **SETRAN** to transfer the superelement to a different coordinate system, the superelement's master nodes are rotated with it by default. This is typically useful if the original superelement's master nodes are rotated, into a cylindrical system for example. (In this case, the transfer does not effect the superelement stiffness matrix.) If the original superelement has no rotated nodes, it is likely that the transferred superelement will not need rotated nodes either. You can prevent node rotation in such cases by setting the NOROT field on **SETRAN** to 1. (The superelement stiffness matrix and load vector are modified by the program for this type of transfer.)

6. Verify the location of the superelement using graphics displays and listings. Superelements are represented by an edge outline display, the data for which are written to the matrix file in the generation pass. To produce a graphics display:
   Command(s): EPLOT
   GUI: Utility Menu> Plot> Elements

   To produce a listing:
   Command(s): SELIST
   GUI: Utility Menu> List> Other> Superelem Data

7. Save the complete model database:
   Command(s): SAVE
   GUI: Utility Menu> File> Save as Jobname.db

   Leave PREP7 using one of these methods:
   Command(s): FINISH
   GUI: Main Menu> Finish

8.1.2.3. Apply Loads and Obtain the Solution

This step is performed during the solution phase of the analysis. The procedure to obtain the use-pass solution depends on the analysis type. As mentioned earlier, you can subject a superelement to any type of analysis. You should, of course, have the appropriate matrices generated during the generation pass. For example, if you intend to do a structural dynamic analysis, the mass matrix must be available. The procedure is as follows:

1. Enter SOLUTION using one of these methods:
   Command(s): /SOLU
   GUI: Main Menu> Solution

2. Define the analysis type and analysis options.

For large rotation analyses - turn large deformation effects on [**NLGEOM**,ON], and define the proper number of substeps for the nonlinear analysis.

3. Apply loads on the nonsuperelements. These may consist of DOF constraints and symmetry conditions [**D** family of commands], force loads [**F** family], surface loads [**SF** family], body loads [**BF** family], and inertia loads [**ACEL**, etc.]. Remember that inertia loads will affect the superelement only if its mass matrix was generated in the generation pass.

   Note — For large rotation analyses, be sure to apply the proper constraints in this step.

4. Apply superelement load vectors (if any) using one of these methods:

   Command(s): SFE
   GUI: Main Menu> Solution> Define Loads> Apply> Load Vector> For Superelement

   One load vector per load step (created during the generation pass) is available on the superelement matrix file, and is identified by its reference number:

   ```
   SFE,63,1,SELV,,0.75
   ```

   applies, on element number 63, load vector number 1, with a scale factor of 0.75. Thus the `ELEM` field represents the element number of the superelement, `LKEY` represents the load vector number (default = 1), `Lab` is SELV, and `VAL1` represents the scale factor (default = 0.0). (See the **SFE** command description for more information.)

   Note — The load vector orientation is fixed (frozen) to the superelement, so if the superelement is used in a rotated position, the load vector rotates with it. The same applies to the degree of freedom directions (UX, UY, ROTY, etc.). They too are fixed to the superelement and will rotate with the superelement if it is rotated (unless `NOROT` = 1 on the **SETRAN** command, in which case the nodal coordinate systems will not be rotated).

5. Specifiy load step options that are appropriate for the analysis type. Use the **EQSLV** command to select an appropriate equation solver based on the chosen analysis type and the physics of the problem.

   Note — Some solvers, such as the PCG solver, are not available for the use pass.

6. Initiate the solution:

   Command(s): SOLVE
   GUI: Main Menu> Solution> Solve> Current LS

   Results from the solution consist of the complete solution for nonsuperelements and the reduced solution - DOF solution at masters - for the superelements. The complete solution for nonsuperelements is written to the results file (**Jobname.RST**, **RTH**, or **RMG**), which you can postprocess using normal postprocessing procedures.

   The reduced solution is written to the file **Jobname.DSUB**. You can review this file using one of these methods:

   Command(s): SEDLIST
   GUI: Main Menu> General Postproc> List Results> Superelem DOF
   Utility Menu> List> Results> Superelem DOF Solu
   To expand the reduced solution to all elements within the superelement, you will need to perform the expansion pass, explained next.

7. Leave SOLUTION:

   Command(s): FINISH

GUI: Main Menu> Finish

8.1.3. Expansion Pass: Expanding Results Within the Superelement

The *expansion pass* is where you start with the reduced solution and calculate the results at all degrees of freedom in the superelement. If multiple superelements are used in the use pass, a separate expansion pass will be required for each superelement.

The procedure for the expansion pass assumes that the **.EMAT**, **.ESAV**, **.SUB**, **.TRI**, **.DB**, and **.SELD** files from the generation pass and the **.DSUB** file from the use pass are available. The expansion pass logic automatically detects which, if any, triangularized matrix files are available and chooses the appropriate expansion pass method and solver accordingly. For example, if the **.TRI** exists, the frontal solver and backsubstitution method will be selected. If an offset of node numbers was used in the use pass [**SETRAN** or **SESYMM**], it will automatically be taken into account in the expansion pass.

The backsubstitution method uses the reduced solution from the use pass and substitutes it back into the available triangularized matrix file to calculate the complete solution. The full resolve solution reforms the element stiffness matrices originally used to create the superelement. The global stiffness matrix for these elements is then assembled. The reduced solution is applied to the model as displacement boundary conditions, and the complete solution within the superelement is solved.

> *Note* — The displacement boundary conditions are automatically applied internally at the master degrees of freedom during the expansion pass solution and are not deleted when the solution completes. If any subsequent analyses are to be performed, users must be aware these boundary conditions exist in the model, and delete them, if necessary.

The expansion pass logic for substructuring analyses first searches for the superelement **.TRI** file and, if found, chooses the frontal solver to perform a backsubstitution (the **EQSLV** command is ignored). If the **.TRI** file is not found, it searches for the superelement **.LN22** file and, if found, chooses the sparse solver to perform a backsubstitution (the **EQSLV** command is ignored).

If neither the **.TRI** nor the **.LN22** files are detected for the specified superelement, the full resolve method is chosen. The PCG solver is chosen by default for the full resolve method. You can select the frontal or sparse solver using the **EQSLV** command to override the default. Other equation solvers cannot be used with the full resolve method.

1. Clear the database:
   Command(s): /CLEAR
   GUI: Utility Menu> File> Clear & Start New

   This has the same effect as leaving and re-entering the ANSYS program.
2. Change the jobname to what it was during the generation pass. This way, the program can easily identify the files required for the expansion pass:
   Command(s): /FILNAME
   GUI: Utility Menu> File> Change Jobname
3. Restore the generation pass database:
   Command(s): RESUME
   GUI: Utility Menu> File> Resume Jobname.db
4. Enter SOLUTION using one of these methods:
   Command(s): /SOLU
   GUI: Main Menu> Solution

5. Activate the expansion pass and its options:

   Command(s): EXPASS
   GUI: Main Menu> Solution> Load Step Opts> ExpansionPass

   Expansion pass on or off - Choose "on."

   Name of superelement to be expanded - Specify the name (`Sename`):

   Command(s): SEEXP
   GUI: Main Menu> Solution> Load Step Opts> ExpansionPass> Expand Superelem

   (The complete name of the file is assumed to be ***`Sename`*.SUB**.)

   Name of the reduced solution file from use pass - Specify the name (`Usefil`) using the **SEEXP** command (or the menu path shown above). The complete name of the file is assumed to be ***`Usefil`*.DSUB**.

   Real or imaginary component of displacement - Applicable only if the use pass was a harmonic response analysis. Use the **SEEXP** command (or the menu path shown above).

   Solution to be expanded - Identify the use pass solution to be expanded. You can use either the load step and substep numbers or the time (or frequency) to identify the solution:

   Command(s): EXPSOL
   GUI: Main Menu> Solution> Load Step Opts> ExpansionPass> Single Expand> By Load Step
   Main Menu> Solution> Load Step Opts> ExpansionPass> Single Expand> By Time/Freq

   Note — If the load step being expanded contains nonzero DOF constraints, the database must have matching DOF values. If it does not, the DOF constraints must be respecified [**D**] in the expansion pass.

6. Specify load step options. The only options valid for a substructure expansion pass are output controls:

   Output Controls - These options control printed output, database and results file output, and extrapolation of results.

   If you want to include any results data on the printed output file (**Jobname.OUT**):

   Command(s): OUTPR
   GUI: Main Menu> Solution> Load Step Opts> Output Ctrls> Solu Printout

   If you want to control the data on the results file (**Jobname.RST**):

   Command(s): OUTRES
   GUI: Main Menu> Solution> Load Step Opts> Output Ctrls> DB/Results File

   If you want to review element integration point results by *copying* them to the nodes instead of *extrapolating* them (default):

   Command(s): ERESX
   GUI: Main Menu> Solution> Load Step Opts> Output Ctrls> Integration Pt

7. Start expansion pass calculations:

   Command(s): SOLVE
   GUI: Main Menu> Solution> Solve> Current LS

8. Repeat steps 5 to 7 for additional use pass solutions to be expanded. If you need to expand the solution for a *different superelement*, you will need to leave and re-enter SOLUTION.

9. Finally, leave SOLUTION:

Command(s): FINISH
GUI: Main Menu> Finish

10. Postprocess results in the superelement using standard techniques.

Note — An expansion pass is not valid if the use pass was a PSD analysis.

8.2. Sample Analysis Input

A sample command input listing for a substructuring analysis is shown below. This example assumes a single superelement which is possibly combined with nonsuperelements.

```
!       GENERATION PASS
! Build the model (superelement portion)
/FILNAME,GEN         ! Jobname = GEN (for example)
/TITLE,...
/PREP7               ! Enter PREP7
---
---                  ! Generate superelement portion of model
FINISH
! Apply loads and create the superelement matrices
/SOLU                ! Enter SOLUTION
ANTYPE,SUBST         ! Substructure analysis
SEOPT,GEN,...        ! Superelement name and other substructure analysis options
M,...                ! Master DOF
D,...                ! Loads.  A load vector will be generated and
---                  !   written to the superelement matrix file
---                  ! Load step options
SAVE                 ! Save the database for later expansion pass
SOLVE                ! Initiate solution -- creates GEN.SUB file
                     !   containing superelement matrix and load vector
---                  ! Loads for second load vector (D and M may not changed)
SOLVE                ! Add load vector 2
---                  ! Repeat loading and SOLVE sequence for additional load vectors
---                  !    (Up to 31 total)
FINISH
!       USE PASS
! Build the model
/CLEAR               ! Clear the database (or exit and re-enter ANSYS)
/FILNAME,USE         ! Jobname = USE (for example)
/PREP7               ! Enter PREP7
ET,1,MATRIX50        ! MATRIX50 is the superelement type
ET,2,...             ! Element type for nonsuperelements
---                  ! Generate nonsuperelement model
---
TYPE,1               ! Point to superelement type reference number
SETRAN,...           ! May be required for node number offset
SE,...               ! Read in the superelement created by SETRAN
EPLOT                ! Verify location of superelement
NSEL,...             ! Select nodes at interface
CPINTF,ALL           ! Couple node pairs at interface (required if
                     !   node numbers at interface don't match)
NSEL,ALL
FINISH

! Apply loads and obtain the solution
/SOLU                ! Enter SOLUTION
ANTYPE,...           ! Analysis type and analysis options
---
---
D,...                ! Loads on nonsuperelements
---
---
SFE,...              ! Apply superelement load vector
---                  ! Load step options
---
SAVE                 ! Save database before solution
SOLVE                ! Initiate solution -- calculates complete solution
                     !   for nonsuperelements (USE.RST, RTH or RMG) and
```

```
                    !   reduced solution for superelements (USE.DSUB)
FINISH

! ... Review results in nonsuperelements

!       EXPANSION PASS
/CLEAR              ! Clear the database
/FILNAME,GEN        ! Change jobname back to generation pass jobname
RESUME              ! Restore generation pass database
/SOLU               ! Enter SOLUTION
EXPASS,ON           ! Activate expansion pass
SEEXP,GEN,USE...    ! Superelement name to be expanded (GEN, unless SETRAN used)
---                 ! Load step options (mainly output controls)
---
SOLVE               ! Initiate expansion pass solution.  Full
                    !   superelement solution written to GEN.RST (or
                    !   RTH or RMG).
FINISH

! ... Review results in superelements
```

For more information, see the **ANTYPE**, **SEOPT**, **M**, **ET**, **SETRAN**, **SE**, **CPINTF**, **EXPASS**, and **SEEXP** command descriptions.

8.3. Top-Down Substructuring

The substructuring procedure described in the previous section is called *bottom-up* substructuring, meaning that each superelement is separately generated in an individual generation pass, and all superelements are assembled together in the use pass. This method is suitable for very large models which are divided into smaller superelements so that they can "fit" on the computer.

For substructuring of smaller models or of systems with global project geometry controls, and for isolated component analysis, you can use a slightly different technique known as *top-down* substructuring. This method is suitable, for instance, for substructuring of the linear portion of nonlinear models that are small enough to fit on the computer. An advantage of this method is that the results for multiple superelements can be assembled in postprocessing. The procedure for top-down substructuring is briefly explained below, and is followed by a sample input.

1. First build the entire model, including both the superelement and nonsuperelement portions. Save this model on a named database file (for example, **FULL.DB**). The full model database is later required for the expansion pass. It will also be required for the use pass if the model consists of nonsuperelements.
2. Perform the generation pass on a selected subset of the entire model. Because the full model has already been built, all you need to do is to select the elements for the superelement portion, apply the desired loads (for the load vector), and create the superelement with the **SOLVE** command (**Main Menu> Solution> Solve> Current LS**).

   The use of components may be helpful for this. To group items into a component, use the **CM** command (**Utility Menu> Select> Comp/Assembly> Create Component**).

   If multiple superelements are to be generated, you will need to exit and re-enter SOLUTION each time and repeat the select-load-solve steps. Be sure to use a different jobname for each superelement.
3. Perform the use pass. Enter PREP7 and start by restoring the full model database and then selecting only the nonsuperelement portion of the model. Next, define the superelement type [**ET**, **TYPE**] and read in the appropriate superelement matrices. In most cases, you don't need to worry about the connecting nodes between the superelements, because they were all generated from a single model.

   Enter SOLUTION and define the analysis type and analysis options. Apply loads on the nonsuperelements, read in load vectors (if any), specify load step options, and initiate the use pass solution.

4. Perform the expansion pass. Start by restoring the full model database, with all elements and nodes active. Then expand each superelement separately, using the appropriate jobnames and exiting and re-entering SOLUTION each time. You can then review the results in each superelement using normal postprocessing procedures. Use of the full database, **FULL.DB**, allows the reading in of multiple superelement results:

```
RESUME,FULL,DB
/POST1
FILE,GEN1
SET,...
FILE,GEN2
SET,...!Will not clear previous superelement results
```

A sample input for top-down substructuring follows. This example assumes a model with one superelement and other nonsuperelements.

```
!             Sample input for top-down substructuring
!
!     BUILD THE FULL MODEL
!
/FILNAME,FULL        ! Jobname = FULL (for example)
/TITLE,...
/PREP7               ! Enter PREP7
---
---                  ! Generate entire model, including both the
---                  !   superelement and nonsuperelement portions
---
SAVE                 ! Save the full model database.  It is required for
                     !   the (use pass and) expansion pass.
FINISH

!     GENERATION PASS

!

/FILNAME,GEN         ! Jobname = GEN (for example)
/SOLU                ! Enter SOLUTION
ANTYPE,SUBSTR        ! Substructure analysis type
SEOPT,GEN,...        ! Analysis options
ESEL,...             ! Select elements and
NSLE                 !   nodes in the superelement portion
M,...                ! Master DOF
D,...                ! Loads.  A load vector will be generated and written to the
---                  !   superelement matrix file
---                  ! Load step options
---
SOLVE                ! Initiate solution -- creates superelement
                     !   matrix file GEN.SUB.
---                  ! Loads for second load vector (D and M may not changed)
SOLVE                ! Add load vector 2
---                  ! Repeat loading and SOLVE sequence for additional load vectors
---                  !    (Up to 31 total)
FINISH

!     USE PASS
!
/CLEAR               ! Clear database for use pass
/FILNAME,USE         ! Jobname = USE (for example)
RESUME,FULL,DB       ! Restore full model database (for nonsuperelements)
ESEL,...             ! Select elements and
NSLE                 !   nodes in the nonsuperelement portion

/PREP7
ET,...,MATRIX50      ! Superelement type (type number not used by nonsuperelements)
TYPE,...             ! Point to superelement type reference number
SE,GEN               ! Read in superelement matrix (GEN.SUB created above)
EPLOT
FINISH
```

```
/SOLU
ANTYPE,...           ! Analysis type and analysis options
---
D,...                ! Loads on nonsuperelements
---
---
SFE,...              ! Superelement load vector
---
---                  ! Load step options
---
SOLVE                ! Initiates solution -- calculates complete
                     !   solution for nonsuperelements (USE.RST, etc.)
                     !   and reduced solution for superelement (USE.DSUB)
FINISH

!     EXPANSION PASS
!
/CLEAR               ! Clear database for expansion pass
/FILNAME,GEN         ! Change jobname back to generation pass jobname
RESUME,FULL,DB       ! Restore full model database

/SOLU                ! Enter SOLUTION
ANTYPE,SUBSTR
EXPASS,ON            ! Activate expansion pass
EXPSOL,...           ! Specifies the solution to be expanded
SEEXP,GEN,USE,...    ! Superelement name to be expanded
---                  ! Load step options (mainly output controls)
---
SOLVE                ! Initiate expansion pass solution.  Full
                     !   superelement solution written to GEN.RST (or
                     !   RTH or RMG).
FINISH

! ... Review results in superelement
```

Please see the **ANTYPE**, **SEOPT**, **M**, **ET**, **SE**, **EXPASS**, and **SEEXP** command descriptions for more information.

8.4. Automatically Generating Superelements

When creating multiple superelements, the two methods described in the previous sections (bottom-up substructuring and top-down substructuring) both require repeating a set of **/SOLU** commands for each superelement you want to create. These methods also require a master DOF to be defined for each superelement. If any superelements connect to each other, then the master DOF must be chosen carefully on the interface(s) between each connecting superelement.

When creating multiple superelements, use the following automatic superelement generation process to quickly create superelements (**.SUB** files), as well as the master DOF necessary on the interfaces between each superelement. This simplifies the creation of the superelements and it efficiently breaks a larger model into smaller models, for example, to be used in a nonlinear analysis.

To automatically generate superelements:

1. If using the bottom-up substructuring method first create the part of the model that will become superelements. If using the top-down substructuring method, first create whole model, then select the part of the model that will become superelements.
2. Perform the generation pass using **SEOPT** and any other **/SOLU** commands to define any necessary options for the substructuring analysis.
3. Use **SEGEN** to define the options for the automatic superelement generation process. If `stopStage` = PREVIEW is selected, then the model is only broken into domains (superelements). No reduced matrices are created and the superelements (**.SUB** files) are not actually created. You can then graphically (visually) preview each domain by using **/PNUM**,DOMAIN. By default, master DOFs are automatically defined at each of the following locations: all DOFs on the interfaces between each superelement, all DOFs associated

with contact elements (TARGE169 to CONTA175), and at all DOFs associated with nodes having a point load defined. The option to manually define the master DOF only makes sense AFTER a 'preview pass' has been made, as the exact number of superelements and the superelement boundaries for each superelement cannot be known until the process is completed at least once.

> *Note* — Due to the heuristics in the automatic domain decomposer, which is used to create the domains that will become superelements, the number of defined superelements may exceed the number of requested superelements.

After completing a preview pass, you can then add master DOFs or remove master DOFs that were automatically defined during the preview pass. At least one master DOF must be defined for each superelement. Then set `stopStage` = GEN, and if any master DOFs were added or removed, set `mDof` = YES, and solve the model.

4. Use **SOLVE** to either preview or generate the automatically created superelements. Note that multiple load steps are not supported with automatic superelement generation.

8.5. Nested Superelements

A powerful substructuring feature available in the ANSYS program is the ability to use *nested* superelements (one superelement containing another). When you generate a superelement, one of the elements in the generation pass may be a previously generated superelement.

For example, suppose that you have a superelement named PISTON. You can generate another superelement named CYLINDER which contains the superelement PISTON. Now, for a complete analysis of the cylinder and its piston, you will need to perform *one* use pass and *two* expansion passes. The use pass calculates the reduced solution for the master DOF in the superelement CYLINDER. The first expansion pass calculates the complete solution for CYLINDER and the reduced solution for PISTON. The second expansion pass then gives you the complete solution for PISTON.

8.6. Prestressed Substructures

In modeling a system's behavior properly, it may be important to consider its stress state. That stress state will influence the values of the stiffness matrix terms. The stress state from a previous structural solution may be included when the stiffness matrix is formed in a superelement generation pass. Stress stiffening can provide strength in a structure which would normally not have any resistance to certain loadings. For example, a stretched cable can resist a normal loading while a slack cable cannot. Stress stiffening can also change the response frequencies of a system which impacts both modal and transient dynamic problems.

Two different approaches can be used to generate prestressed substructures. These approaches are a static analysis prestress and a substructuring analysis prestress.

8.6.1. Static Analysis Prestress

1. Build the model, define a static analysis (**ANTYPE** command or menu path **Main Menu> Solution> Analysis Type> New Analysis**) and apply the stiffening load.
2. Specify that prestress effects be calculated (**PSTRES** command or menu path **Main Menu> Solution> Analysis Type> Analysis Options** or **Main Menu> Solution> Unabridged Menu> Analysis Type> Analysis Options**).
3. Specify that the File.EMAT be written (**EMATWRITE** command; not available via the GUI)
4. Initiate the static analysis (**SOLVE** command or menu path **Main Menu> Solution> Solve> Current LS**).

5. Perform the generation pass. Include the prestress effects from the static analysis by issuing the **PSTRES** command. (It is important to have the prestress effects active during the static analysis and the generation pass.)
6. Perform the use and expansion passes.
7. Review the results.

8.6.2. Substructuring Analysis Prestress

Note — This method does not require a static analysis on the full DOF model.

1. Build the model and perform the generation pass. Make sure to reserve space for the stress stiffening matrix by setting *SESST* = 1 on the **SEOPT** command (**Main Menu> Solution> Analysis Type> Analysis Options**).
2. Apply loads and perform the static use pass.
3. Perform the expansion pass using the **PSTRES** command (**Main Menu> Solution> Analysis Type> Analysis Options**) to calculate prestress effects.
4. Repeat the generation pass for a new substructure generation, while continuing the prestress effects using **PSTRES**.
5. Solve the second generation pass and perform the use pass.
6. Perform the expansion pass and review the results.

8.7. Where to Find Examples

The *ANSYS Verification Manual* presents test-case analyses demonstrating the analysis capabilities of the ANSYS program. While these test cases demonstrate solutions to realistic analysis problems, the *ANSYS Verification Manual* does not present them as step-by-step examples with lengthy data-input instructions and printouts; however, most ANSYS users with at least limited finite-element experience should be able to fill in the missing details by reviewing each test case's finite element model and input data with accompanying comments.

The *ANSYS Verification Manual* contains the following substructuring cases:

VM125 - Radiation Heat Transfer Between Concentric Cylinders
VM141 - Diametrical Compression of a Disk
VM147 - Gray-Body Radiation within a Frustum of a Cone
VM153 - Nonaxisymmetric Vibration of a Stretched Circular Membrane (Using Cyclic Symmetry)

Chapter 9: Component Mode Synthesis

Component Mode Synthesis (CMS) is a form of substructure coupling analysis frequently employed in structural dynamics.

CMS allows you to derive the behavior of the entire assembly from its constituent components. First, the dynamic behavior of each of the components is formulated. Then, by enforcing equilibrium and compatibility along component interfaces, ANSYS forms the dynamic characteristics of the full system model.

CMS is available in the ANSYS Mechanical and ANSYS Structural products.

This chapter covers the following topics:

9.1. Understanding Component Mode Synthesis

Although breaking up a single large problem into several reduced-order problems via substructuring saves time and processing resources, component mode synthesis (CMS) offers the following additional advantages:

- More accurate than a Guyan reduction for modal, harmonic and transient analyses. CMS includes truncated sets of normal mode generalized coordinates defined for components of the structural model.
- The ability to include experimental results, as the substructure model need not be purely mathematical.

A typical use of CMS involves a modal analysis of a large, complicated structure (such as an aircraft or nuclear reactor) where various teams each design an individual component of the structure. With CMS, design changes to a single component affect only that component; therefore, additional computations are necessary only for the modified substructure.

Finally, CMS supports these substructuring features:

- Top-down substructuring
- Nested superelements
- Prestressed substructures.

9.1.1. CMS Methods Supported

ANSYS supports two component mode synthesis methods:

- Fixed interface (**CMSOPT**,FIX)
- Free interface (**CMSOPT**,FREE)

For most analyses, the fixed-interface CMS method is preferable. The free-interface method is useful when your analysis requires more accurate eigenvalues computed at the mid- to high-end of the spectrum. The following table describes the primary differences between the fixed- and free-interface methods:

CMS Methods: Fixed-Interface vs. Free-Interface

Fixed	Free

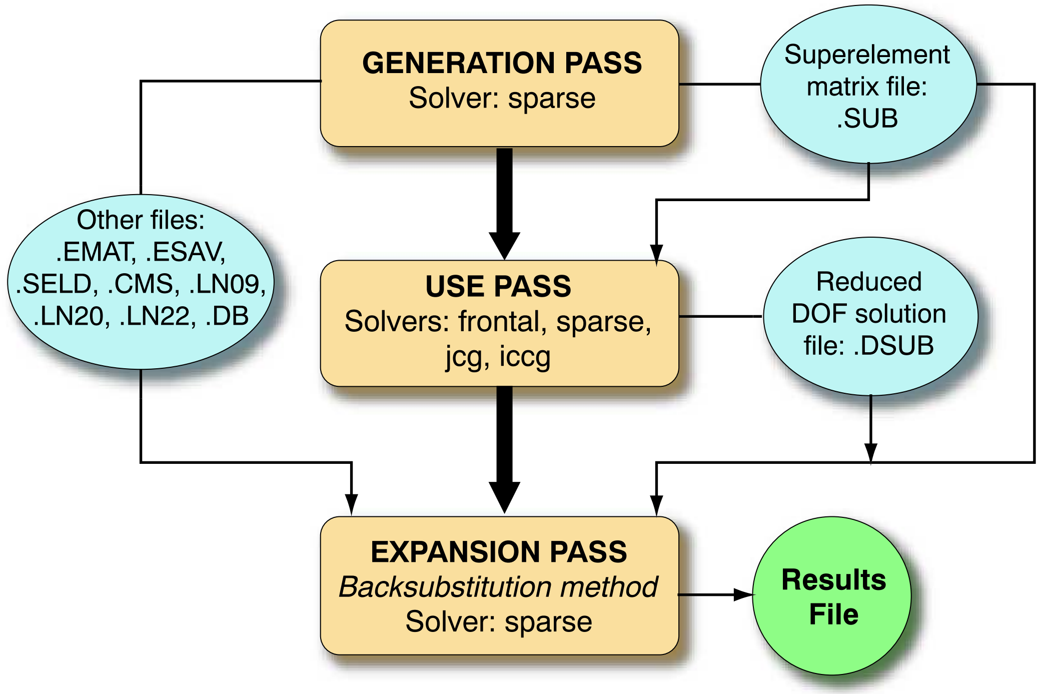
Interface nodes are constrained during the CMS superelement generation pass.	Interface nodes remain free during the CMS superelement generation pass.
No requirement to specify rigid body modes.	You must specify the number of rigid body modes (via the **CMSOPT** command).
Generally recommended when accuracy on only the lower modes of the assembled structure (use pass) is necessary.	Generally recommended when accuracy on both lower and higher modes of the assembled structure (use pass) is required.

For more information, see the discussion of component mode synthesis theory and methods in the *ANSYS, Inc. Theory Reference*.

9.1.2. Solvers Used in Component Mode Synthesis

Following are the solvers and files used in a typical component mode synthesis analysis:

Figure 9.1 Applicable CMS Solvers and Files


GENERATION PASS
Solver: sparse
Superelement matrix file: .SUB
Other files: .EMAT, .ESAV, .SELD, .CMS, .LN09, .LN20, .LN22, .DB
USE PASS
Solvers: frontal, sparse, jcg, iccg
Reduced DOF solution file: .DSUB
EXPANSION PASS
Backsubstitution method
Solver: sparse
Results File

9.2. Employing Component Mode Synthesis

As in substructuring, a component mode synthesis (CMS) analysis involves three distinct steps, called passes:

1. Generation pass (via **SEOPT** and **CMSOPT** commands)
2. Use pass (via **SETRAN**, **SESYMM** and **CPINTF** commands)
3. Expansion pass (via **EXPASS**, **SEEXP**, **EXPSOL** and **NUMEXP** commands)

The CMS generation pass condenses a group of "regular" finite elements into a single CMS superelement, which includes a set of master degrees of freedom (DOFs) and truncated sets of normal mode generalized coordinates. The master DOFs serve to define the interface between the superelements or other elements.

This section covers the following topics:

9.2.1. The CMS Generation Pass: Creating the Superelement

The process for generating a CMS superelement consists of two primary tasks:

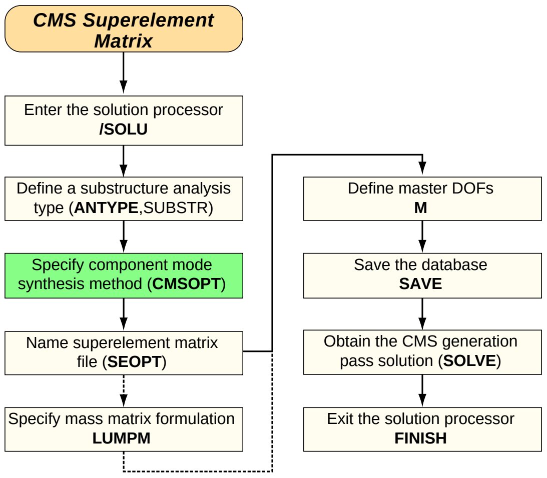
1. **Building the model**

   This step is identical to building the model for a substructure analysis. Define density (DENS)--or mass in some form--because CMS must generate both stiffness and mass matrices.

2. **Creating the superelement matrices**

   The "solution" from a CMS generation pass consists of the superelement matrices (generalized stiffness and mass matrix). This flowchart illustrates the process necessary for creating the superelement matrix file:

Figure 9.2 Process Flow for Creating a CMS Superelement Matrix



CMS Superelement Matrix
Enter the solution processor /SOLU
Define a substructure analysis type (ANTYPE,SUBSTR)
Specify component mode synthesis method (CMSOPT)
Name superelement matrix file (SEOPT)
Specify mass matrix formulation LUMPM
Define master DOFs M
Save the database SAVE
Obtain the CMS generation pass solution (SOLVE)
Exit the solution processor FINISH

Specifying the CMS method When specifying the CMS method, also specify the number of modes and, optionally, the frequency range used to generate the superelement. ANSYS supports both the fixed-interface (**CMSOPT**,FIX) and free-interface (**CMSOPT**,FREE) CMS methods. If employing the free-interface method, also specify the rigid body modes (**CMSOPT**,,,,FBDDEF).

Naming the superelement matrix file ANSYS assigns the extension SUB to the superelement matrix file name that you specify (**SEOPT**,`Sename`); therefore, the complete file name is *Sename*.SUB. The default file name is the **Jobname** (**/FILNAME**).

Specifying the lumped mass matrix formulation Specify the lumped mass matrix formulation (**LUMPM**) if necessary. For most applications, ANSYS recommends the default formulation (depending upon the element type); however, for dynamic analyses involving "skinny" structures such as slender beams or very thin shells, the lumped mass approximation typically yields better results.

Defining master DOFs In a substructure, master degrees of freedom (DOFs) serve as the interface between the superelements or other elements. Define master DOFs (**M**) at all nodes that connect to non-superelements (`Lab1` = ALL), as shown in Example of a Substructuring Application. *You must define master DOFs even if you intend to have no elements in the model other than a superelement.*

Saving a copy of the database Saving a copy of the database (**SAVE**) is necessary because you must work with the same data in the expansion pass.

Obtaining the CMS generation pass solution Output from the solution (**SOLVE**) consists of the superelement matrix file (`Sename`.SUB), where `Sename` is the file name you assigned (via the **SEOPT** command).

After obtaining the CMS superelement matrices, proceed to the use pass and then the expansion pass, as you would in a substructuring analysis.

For a detailed example of how to employ CMS, see Section 9.3: Sample Component Mode Synthesis Analysis.

9.2.2. The CMS Use and Expansion Passes

Although the CMS use pass and expansion pass are identical to those in a substructure analysis, for CMS the use pass supports modal analyses only.

As in substructuring, the generation and expansion passes occur *for each part* (CMS superelement) of the entire structure, and the use pass occurs only once because it uses all superelements together to build the full model. The use pass extracts the eigenvalues of the full model (but not the eigenvectors, because the expansion pass recovers them).

In a free-interface CMS analysis, the use pass may not always extract all of the modes requested via the **MODOPT** command. In such cases, increase or decrease the number of modes to extract and run the use pass eigensolution again.

9.2.3. Plotting or Printing Mode Shapes

Plotting or printing the mode shapes of the assembled structure occurs during postprocessing. The postprocessor (**/POST1**) uses the results files generated by the CMS superelements to display shape results.

Issue the **CMSFILE** command to import the CMS superelement results files into the postprocessor where you can view the assembled structure. (You can issue the command as often as needed to include all or some of the component results files.) Set the desired mode shape of the assembled structure via the **SET** command, then plot (**PLNSOL**) or print (**PRNSOL**) the mode shapes.

9.3. Sample Component Mode Synthesis Analysis

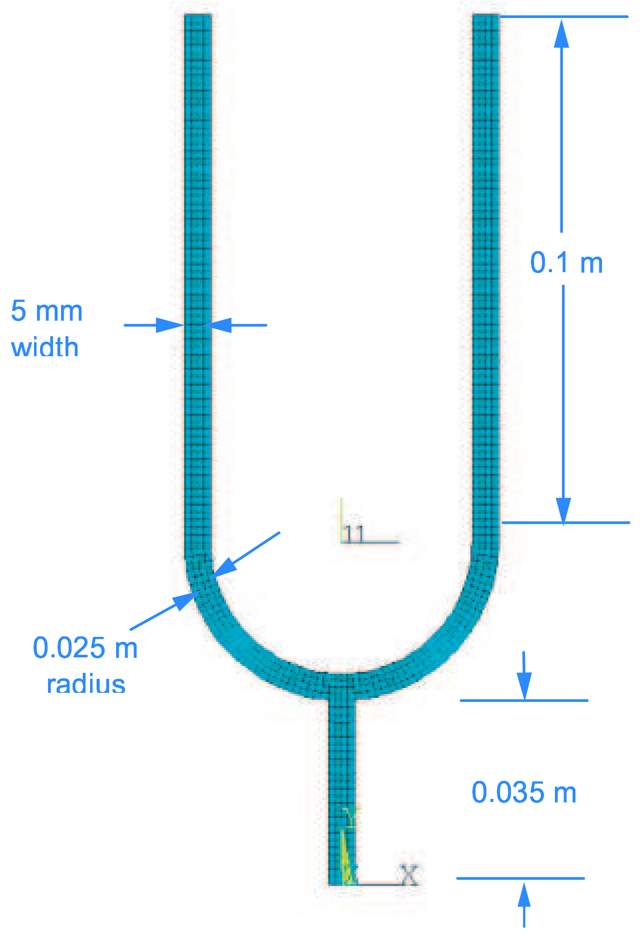
This section introduces you to the ANSYS product's component mode synthesis (CMS) analysis capabilities by way of example. The sample component mode synthesis presents a modal analysis of a 2D tuning fork.

9.3.1. Problem Description

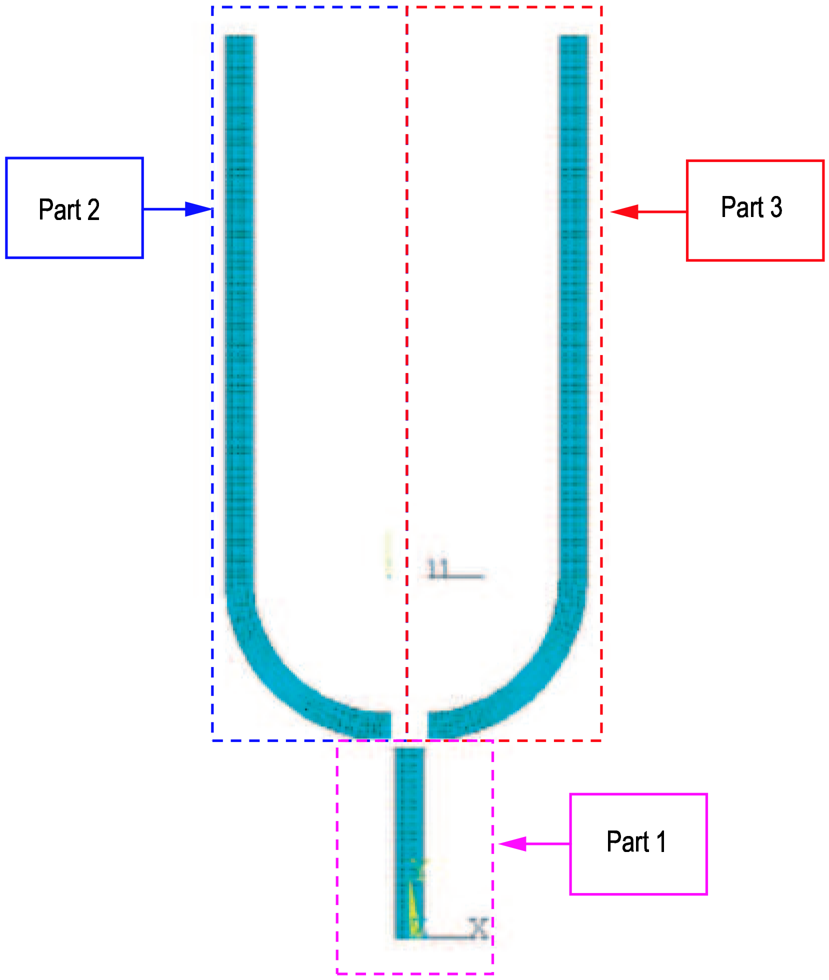
The model is an unconstrained stainless steel tuning fork. After dividing the fork into three CMS superelements, you must determine the vibration characteristics (natural frequencies and mode shapes) of the entire model.

9.3.2. Problem Specifications

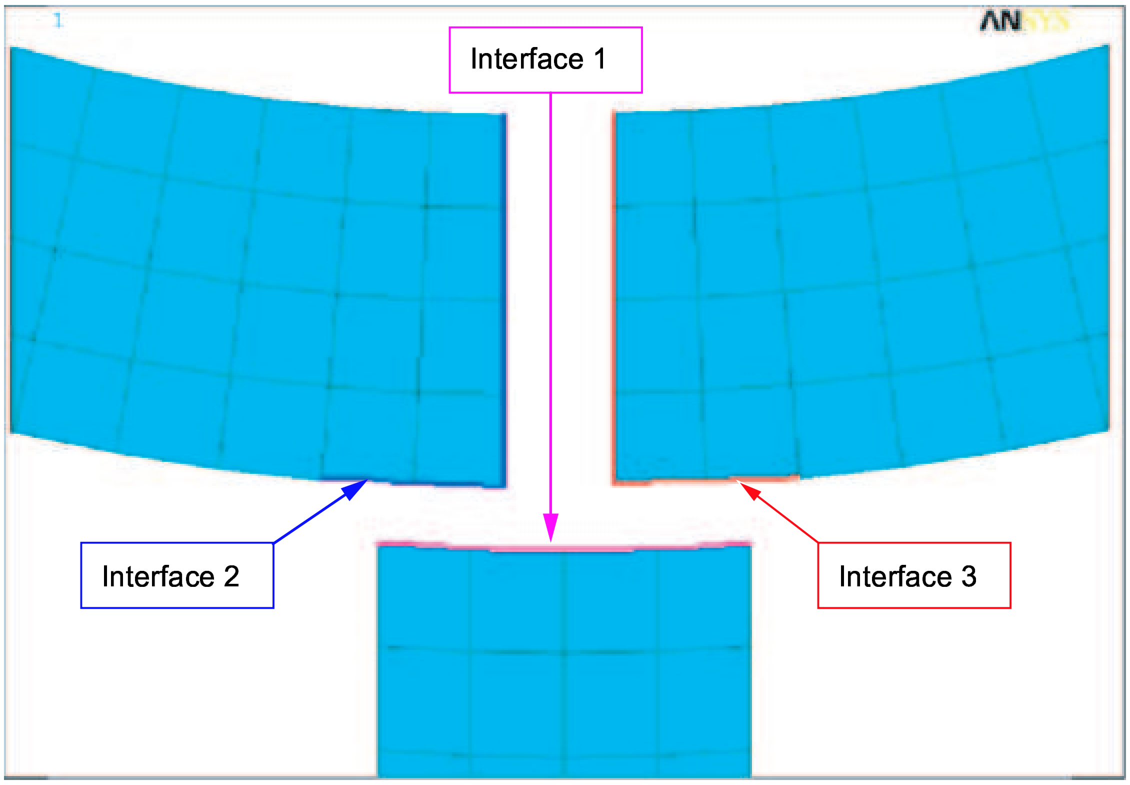
The geometric properties for this analysis follow.



The fork is divided into three CMS superelements:



The three interfaces are as follows:



The material properties for this analysis are as follows:

Young's modulus (E) = 190e9
Poisson's ratio (υ) = 0.3
Density = 7.7e3

The first 10 eigenfrequencies are extracted, and the fourth mode shape (the first non-rigid body mode) is expanded.

9.3.3. Input for the Analysis: Fixed-Interface Method

Use this input file (named **cms_sample.inp**) to perform the example CMS analysis via the fixed-interface method. The file contains the complete geometry, material properties, and components (nodes and elements).

```
/batch,list
/title, 2D Tuning Fork
! Component Mode Synthesis on a 2D example
! The Structure is divided into 3 CMS Superelements

! STEP #1
! Start an ANSYS interactive session

! STEP #2
! Read in this input file: cms_sample.inp

finish
/clear

/filnam,full
/units,si
blen=0.035
radi=0.025
tlen=0.1
tthk=0.005

/plopts,minm,0
/plopts,date,0
/pnum,real,1
/number,1

/prep7
k,1,-tthk/2
k,2,tthk/2
k,3,-tthk/2,blen
k,4,tthk/2,blen
local,11,1,,blen+tthk+radi
k,5,radi+tthk,-180
k,6,radi,-180
kgen,2,3,4,1,-tthk
k,9,radi
k,10,radi+tthk
a,5,6,7,3
a,3,7,8,4
a,4,8,9,10
csys,0
a,1,2,4,3
k,11,-radi-tthk,blen+tthk+radi+tlen
k,12,-radi,blen+tthk+radi+tlen
k,13,radi,blen+tthk+radi+tlen
k,14,radi+tthk,blen+tthk+radi+tlen
a,5,6,12,11
a,9,10,14,13
eshape,2
esize,tthk/3.5
et,1,plane42,,,3
r,1,tthk
amesh,all
mp,ex,1,190e9
mp,dens,1,7.7e3
mp,nuxy,1,0.3
```

```
nsel,s,,,38
nsel,a,,,174,176
nsel,a,,,170
cm,interface1,node
nsel,s,,,175
nsel,a,,,168
nsel,a,,,180,182
nsel,a,,,38,176,138
cm,interface2,node
nsel,s,,,175
nsel,a,,,168
nsel,a,,,180,182
nsel,a,,,170,174,4
cm,interface3,node
esel,s,,,273,372
cm,part1,elem
esel,s,,,373,652
esel,a,,,1,129
esel,a,,,130
esel,a,,,133,134
esel,a,,,137,138
esel,a,,,141,142
cm,part2,elem
cmsel,s,part1
cmsel,a,part2
esel,inve
cm,part3,elem
allsel,all
save
finish

/eof

! STEP #3 (a. through j.)
! Generation pass 1

! a.
! Change the active jobname which will become the superelement name

/filnam,part1

! b.
! Specify the analysis type as substructuring

/solu
antype,substr

! c.
! Specifies the name to be assigned to superelement matrix file
! Strongly suggested to be the same as the active jobname

seopt,part1,2

! d.
! Specifies CMS options

cmsopt,fix,10

! e.
! Selects element component named "part1"

cmsel,s,part1

! f.
! Selects node component named "interface1"

cmsel,s,interface1

! g.
! All the active DOFs (that is, on the nodes which belong to "interface1")
! are set as masters
```

```
m,all,all

! h.
! Selects all the nodes attached to the selected elements
! (that is, elements which belong to "part1")

nsle

! i.
! solve the first CMS generation pass

solve
finish

! j.
! Save the generation pass 1 database

save

! Repeat the generation pass for "part2"

! Generation pass 2

/filnam,part2
/solu
antype,substr
seopt,part2,2
cmsopt,fix,10
cmsel,s,part2
cmsel,s,interface2
m,all,all
nsle
solve
finish
save


! Repeat the generation pass for "part3"

! Generation pass 3

/filnam,part3
/solu
antype,substr
seopt,part3,2
cmsopt,fix,10
cmsel,s,part3
cmsel,s,interface3
m,all,all
nsle
solve
finish
save


! STEP #4 (a. through i.)
! Use pass

! a.
! Clears the database

/clear,nostart

! b.
! Change the active jobname which will become the use pass name

/filnam,use

! c.
! A superelement element type is created

/prep7
```

```
et,1,matrix50

! d.
! Element type attribute pointer set to 1

type,1

! e.
! Brings in the three superelements created above

se,part1
se,part2
se,part3
finish

! f.
! A modal analysis is performed

/solu
antype,modal

! g.
! Specifies modal analysis options

modopt,lanb,10

! h.
! Expands 10 modes

mxpand,10

! i.
! Solve the modal analysis

solve
finish

! STEP #5 (a. through g.)
! Expansion pass 1

! a.
! Clears the database

/clear,nostart

! b.
! Changes the jobname to superelement 1 name

/filnam,part1

! c.
! resume the database

resume

! d.
! Specifies the expansion pass

/solu
expass,on

! e.
! Specifies superelement name and use pass name

seexp,part1,use

! f.
! Specifies the loadstep and substep to be expanded

expsol,1,4

! g.
! Solve the first expansion pass

solve
finish

! Repeat the expansion pass for "part2"

! Expansion pass 2

/clear,nostart
/filnam,part2
resume
/solu
expass,on
seexp,part2,use
expsol,1,4
solve
finish

! Repeat the expansion pass for "part3"

! Expansion pass 3

/clear,nostart
/filnam,part3
resume
/solu
expass,on
seexp,part3,use
expsol,1,4
solve
finish


! STEP #6 (a. through c.)
! Reads results for "load step 1 - substep 4"

! a.
! Specifies the data file where results are to be found

/post1
cmsfile,add,part1,rst
cmsfile,add,part2,rst
cmsfile,add,part3,rst

! b.
! Reads the first data set

set,first

! c.
! Plots the displacement contour in the x direction

plnsol,u,x
finish
```

9.3.4. Analysis Steps: Fixed-Interface Method

The following table describes the input listing and the steps involved in the example fixed-interface CMS analysis in more detail.

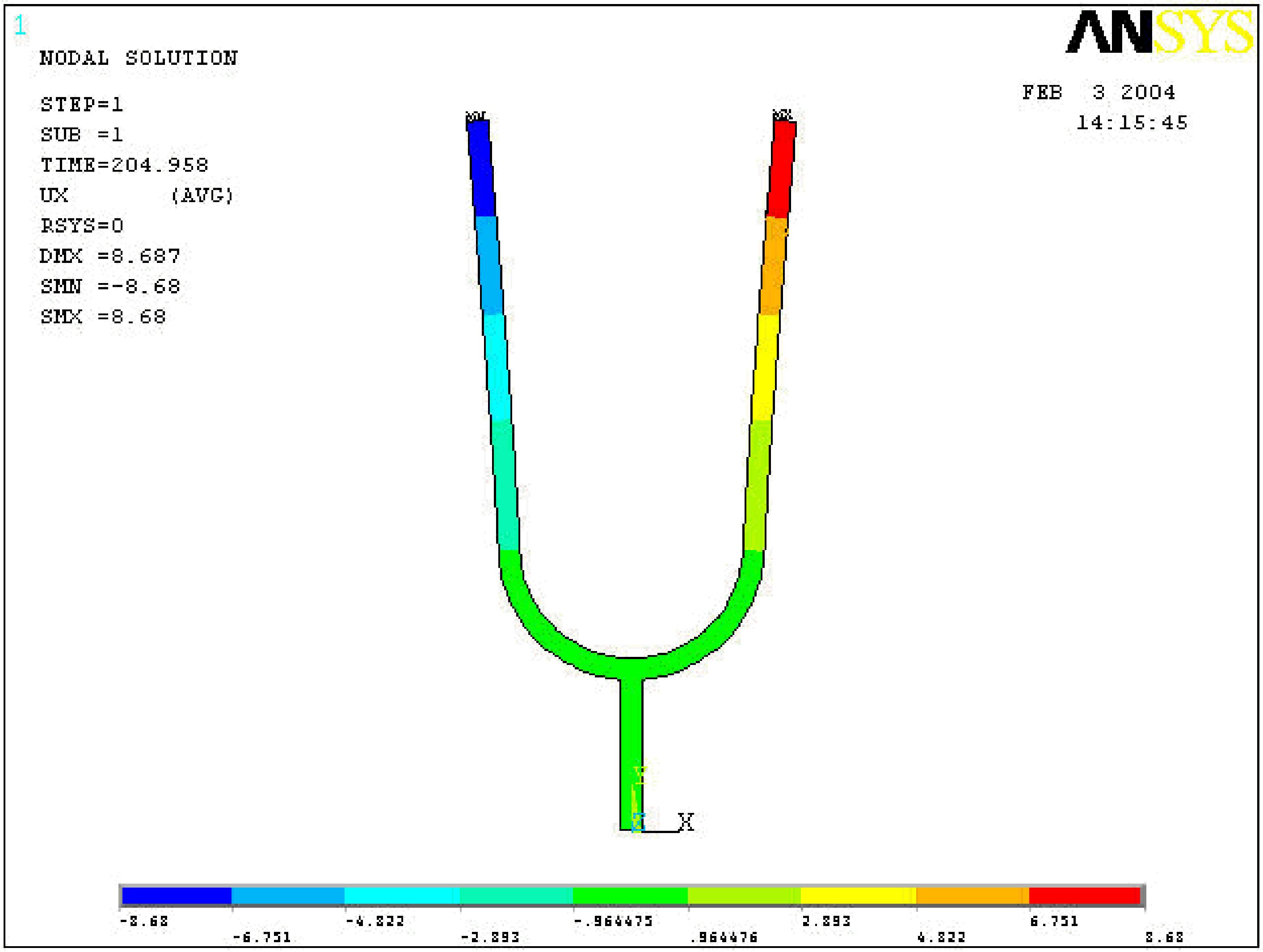
Step	Description	ANSYS Command(s)
1.	Start an ANSYS interactive session.	
2.	Read the input file: **cms_sample.inp**	**/INPUT**,CMS_SAMPLE.INP
3.	Perform the **generation pass**.	
	a. Change the **Jobname** to PART1.	**/FILNAME**,PART1

	b. Specify the analysis type as substructuring.	**/SOLU** **ANTYPE**,SUBSTR
	c. Assign a name to the superelement matrix file.	**SEOPT**,PART1,2
	d. Specify CMS options.	**CMSOPT**,FIX,10
	e. Select element component PART1.	**CMSEL**
	f. Select node component INTERFACE1.	**CMSEL**,S,INTERFACE1
	g. Set all active DOFs as masters.	**M**,ALL,ALL
	h. Select all nodes attached to the selected elements.	**NSLE**
	i. Solve the current analysis.	**SOLVE**
	j. Save the database. ---- *As coded in the input file, generation passes for the remaining parts PART2 and PART3 occur here. Steps a through j are repeated for PART2 and again for PART3. (The **Jobname** and superelement matrix file name change accordingly.) Also, for passes 2 and 3, the node component is INTERFACE2 and INTERFACE3, respectively.*	**SAVE**
Step	**Description**	**ANSYS Command(s)**
4.	Perform the **use pass**.	
	a. Clear the database.	**/CLEAR**,NOSTART
	b. Change the **Jobname** to USE.	**/FILNAME**,USE
	c. Define the element type.	**/PREP7 ET**,1,MATRIX50
	d. Define the element type attribute pointer.	**TYPE**,1
	e. Bring in the three superelements created in the generation passes (PART1, PART2 and PART3)..	**SE**,PART1 **SE**,PART2 **SE**,PART3 **FINISH**
	f. Specify the analysis type as modal.	**/SOLU** **ANTYPE**,MODAL
	g. Specify modal analysis options.	**MODOPT**,LANB,10
	h. Expand 10 modes.	**MXPAND**,10
	i. Solve the current analysis.	**SOLVE** **FINISH**
5.	Perform the **expansion pass**.	
	a. Clear the database.	**/CLEAR**,NOSTART
	b. Change the **Jobname** to PART1.	**/FILNAME**,PART1
	c. Resume the database.	**RESUME**
	d. Perform the expansion.	**/SOLU** **EXPASS**,ON
	e. Name the superelement and use pass.	**SEEXP**,PART1,USE

	f. Specify the loadstep and substep to expand.	**EXPSOL**,1,4
	g. Solve the current analysis. ---- *As coded in the input file, expansion passes for the remaining parts PART2 and PART3 occur here. Steps a through g are repeated for PART2 and again for PART3. (The **Jobname** and superlement name change accordingly.)*	**SOLVE** **FINISH**
Step	**Description**	**ANSYS Command(s)**
6.	**Read the results**.	
	a. Specify the superelement matrix file containing the results.	**/POST1** **CMSFILE**,ADD,PART1,RST **CMSFILE**,ADD,PART2,RST **CMSFILE**,ADD,PART3,RST
	b. Read the first data set.	**SET**,FIRST
	c. Plot the displacement contour in the X direction. ---- *This step completes the sample fixed-interface CMS analysis. Your results should match those shown in Figure 9.3: "Sample CMS Analysis Results: Fixed-Interface Method".*	**PLNSOL**,U,X

The results of your fixed-interface CMS analysis should match those shown here:

Figure 9.3 Sample CMS Analysis Results: Fixed-Interface Method



9.3.5. Input for the Analysis: Free-Interface Method

This input file fragment shows how to set up the generation pass to perform the example CMS analysis via the free-interface method. (All other input remains the same, as shown in Section 9.3.3: Input for the Analysis: Fixed-Interface Method.)

```
.
.
.
! STEP #3 (a. through j.)
! Generation pass 1

! a.
! Change the active jobname which will become the superelement name

/filnam,part1

! b.
! Specify the analysis type as substructuring

/solu
antype,substr

! c.
! Specifies the name to be assigned to superelement matrix file
! Strongly suggested to be the same as the active jobname

seopt,part1,2

! d.
! Specifies CMS options

cmsopt,FREE,10,,,FNUM,3
! If not otherwise specified, the CMSOPT command default behavior
! is to automatically determine rigid body modes in the calculation

! e.
! Selects element component named "part1"

cmsel,s,part1

! f.
! Selects node component named "interface1"

cmsel,s,interface1

! g.
! All the active DOFs (that is, on the nodes which belong to "interface1")
! are set as masters

m,all,all

! h.
! Selects all the nodes attached to the selected elements
! (that is, elements which belong to "part1")

nsle

! i.
! solve the first CMS generation pass

solve
finish

! j.
! Save the generation pass 1 database

save

! Repeat the generation pass for "part2"

! Generation pass 2

/filnam,part2
/solu
antype,substr
seopt,part2,2
cmsopt,free,10,,,FNUM,3
cmsel,s,part2
cmsel,s,interface2
m,all,all
nsle
solve
finish
save

! Repeat the generation pass for "part3"

! Generation pass 3

/filnam,part3
/solu
antype,substr
seopt,part3,2
cmsopt,free,10,,,FNUM,3
cmsel,s,part3
```

```
cmsel,s,interface3
m,all,all
nsle
solve
finish
save

.
.
.
```

9.3.6. Analysis Steps: Free-Interface Method

The following table describes the input code fragment for the generation pass used in a free-interface CMS analysis. (All other analysis steps remain the same, as shown in Section 9.3.4: Analysis Steps: Fixed-Interface Method.)

Step	Description	ANSYS Command(s)
1.	...	
2.	...	
3.	Perform the **generation pass**.	
	a. Change the **Jobname** to PART1.	**/FILNAME**,PART1
	b. Specify the analysis type as substructuring.	**/SOLU** **ANTYPE**,SUBSTR
	c. Assign a name to the superelement matrix file.	**SEOPT**,PART1,2
	d. Specify CMS options.	**CMSOPT**,FREE,10,,,FNUM,3
	e. Select element component PART1.	**CMSEL**
	f. Select node component INTERFACE1.	**CMSEL**,S,INTERFACE1
	g. Set all active DOFs as masters.	**M**,ALL,ALL
	h. Select all nodes attached to the selected elements.	**NSLE**
	i. Solve the current analysis.	**SOLVE**
	j. Save the database. ---- *As coded in the input file, generation passes for the remaining parts PART2 and PART3 occur here. Steps a through j are repeated for PART2 and again for PART3. (The **Jobname** and superelement matrix file name change accordingly.) Also, for passes 2 and 3, the node component is INTERFACE2 and INTERFACE3, respectively.*	**SAVE**
4.	...	
5.	...	
6.	...	

Chapter 10: Rigid Body Dynamics and the ANSYS-ADAMS Interface

The ADAMS software marketed by MSC Software is one of several special-purpose programs used to simulate the dynamics of multibody systems. One drawback of the ADAMS program is that all components are assumed to be rigid. In the ADAMS program, tools to model component flexibility exist only for geometrically simple structures. To account for the flexibility of a geometrically complex component, ADAMS relies on data transferred from finite-element programs such as ANSYS.

The ANSYS-ADAMS Interface is a tool provided by ANSYS, Inc. to transfer data from the ANSYS program to the ADAMS program. Use the ANSYS-ADAMS Interface whenever you want to include flexibility of a body in an ADAMS simulation. Flexibility can be an important aspect in a multibody system, for example, to recognize resonances or to accurately simulate forces and movements of the components. Often, the flexibility of a system is not negligible. A typical example is the model of a piston moving in an engine. The movement of the piston significantly depends on the flexibility of the crankshaft and/or the connecting rod. Because the geometry of a connecting rod can be complex, the ANSYS-ADAMS Interface can be used to account for the connecting rod flexibility.

To use the ANSYS-ADAMS Interface, you first model a flexible component using standard ANSYS commands. While building the model, you must give special attention to modeling interface points where joints will be defined in ADAMS. The next step is to use the ANSYS-ADAMS Interface to write a *modal neutral file* (**Jobname.MNF**) that contains the flexibility information for the component. This file is written in the format required by ADAMS/Flex, an add-on module available for ADAMS. See Section 10.3: Exporting to ADAMS for details on how to use the ANSYS-ADAMS Interface to create the **.MNF** file. For a complete description of the method used to create the modal neutral file and the information it contains, see Section 10.6.1: The Modal Neutral File.

After performing the dynamic simulation in ADAMS, you can use the export capabilities of ADAMS to create an ANSYS input file containing accelerations and rotational velocities of the rigid part and forces acting in the joints of the component. You can then import this file into ANSYS to perform a stress analysis. See Section 10.5: Transferring Loads from ADAMS to ANSYS for details on how to import the loads and perform a subsequent static structural analysis.

The process for transferring flexible components to ADAMS and forces back to ANSYS consists of these general steps:

- Building the model.
- Modeling interface points.
- Exporting to ADAMS (and creating the modal neutral file).
- Running ADAMS simulation using the modal neutral file.
- Transferring resulting loads from ADAMS to ANSYS and performing a static analysis.

Detailed descriptions for each step are presented in the following sections.

10.1. Building the Model

In order to use the ANSYS-ADAMS Interface, you must first create a complete finite element model in ANSYS. To build the model, you specify the jobname and analysis title, and use the **/PREP7** preprocessor to define the element types, element real constants, material properties, and the model geometry. These tasks are common to most analyses and are described in Section 1.2: Building a Model in the *ANSYS Basic Analysis Guide*. For further details on how to create the geometry and mesh, see the *ANSYS Modeling and Meshing Guide*.

When building your model, remember these points:

- The interface is designed to support most element types that have displacement degrees of freedom. Exceptions are axisymmetric elements (for example, PLANE25) and explicit dynamic elements (for example, SOLID164).
- Only linear behavior is allowed in the model. If you specify nonlinear elements, they are treated as linear. For example, if you include nonlinear springs (like COMBIN39), their stiffnesses are calculated based on their initial status and never change.
- Material properties can be linear, isotropic or orthotropic, constant or temperature-dependent. You must define both Young's modulus (EX, or stiffness in some form) and density (DENS, or mass in some form) for the analysis. Nonlinear properties are ignored.
- Damping is ignored when the interface computes the modal neutral file (**Jobname.MNF**). Damping of the flexible component can be added later in the ADAMS program.
- The ADAMS program requires a lumped mass approach (**LUMPM**,ON). This requirement results in the following special considerations.
  - For most structures that have a reasonably fine mesh, this approximation is acceptable. If a model has a coarse mesh, the inertia properties may have errors. To determine what the effect will be, start a modal analysis with and without **LUMPM**,ON and compare the frequencies.
  - When using SHELL63, set KEYOPT(3) = 2 to activate a more realistic in-plane rotational stiffness. If the elements are warped, use SHELL181 with KEYOPT(3) = 2 instead.
  - When using two dimensional elements, the corresponding ADAMS model must lie in the X-Y-plane. Remember that ADAMS models are always three dimensional. The 2-D flexible component transferred will not have any component in the Z-direction.
  - Nodes of a plane element only have two degrees of freedom: translations in the X- and Y-direction. Thus, no moment loads (forces, joints) can be applied in the ADAMS analysis. Likewise, nodes of a solid element only have translational degrees of freedom.
- You cannot apply constraints (**D** command) to the model. Also, make sure that no master degrees of freedom (**M** or **TOTAL** commands) were defined in an earlier analysis.

10.2. Modeling Interface Points

When building a model that will be used in an ADAMS simulation, an important consideration is how to represent interface points within the structure. An interface point is a node that will have an applied joint or force in the ADAMS program. Keep in mind that, in ADAMS, the forces can only be applied to interface points.

The number of interface points used will determine the number of constraint modes for the model. Constraint modes are the static shapes assumed by the component when one degree of freedom of an interface point is given a unit deflection while holding all other interface degrees of freedom fixed. The number of constraint modes is equal to the number of degrees of freedom of all interface points. (For 3-D models, the interface points have 6 DOF; therefore, each interface point has 6 constraint modes.)

You must pay special attention to modeling interface points for these reasons:

- An interface point must have six degrees of freedom (except for 2-D elements).

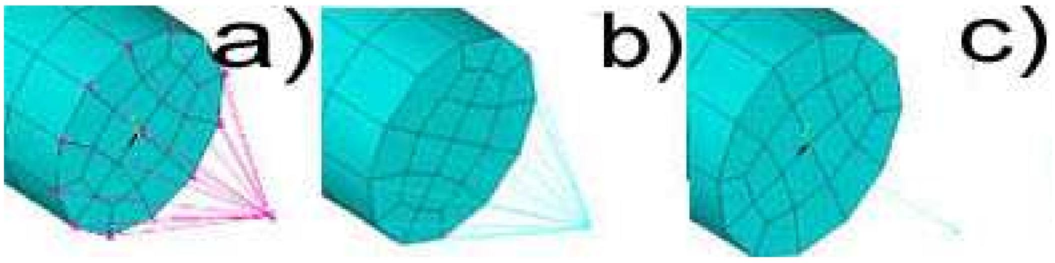
- Force (applied directly or via a joint) should be applied to the structure by distributing it over an area rather than applying it at a single node.
- If there is no node in the structure where you can apply the force or joint in ADAMS (for example, a pin center), you need to create a geometric location for that point.

Use the following guidelines to determine the best way to model the interface points for your structure:

- To ensure that all your loads will be projected on the deformation modes in the ADAMS simulation, you must define all nodes where you are going to apply a joint or a force as interface points.
- Interface points in ANSYS must always have 6 degrees of freedom, except for 2-D elements. If your model consists of solid elements, use constraint equations or a spider web of beam elements (as shown in Figure 10.1: "Connecting a Structure to an Interface Point") to ensure that the interface node has 6 degrees of freedom.
- A good practice for modeling interface points is to reinforce the area using beam elements or constraint equations. Using one of these techniques will distribute the force over an area rather than applying it to a single node, which would be unrealistic.
- If you use a spider web of beam elements, use a high stiffness and a small mass for the beams. Otherwise, you will alter the stiffness and mass of your model, which could result in eigenmodes and frequencies that do not represent the original model.
- You may use constraint equation commands such as **CE** and **CERIG** to attach the interface node (for example, **CERIG**,*MASTE*,*SLAVE*,UXYZ, where *MASTE* is the interface node). Avoid the **RBE3** command since problems can occur with the master degrees of freedom. If you use constraint equations, mesh the interface point with a MASS21 element (use KEYOPT(3) = 0) that has small (negligible) inertias.
- Do not define interface points that lie next to each other and are connected by constraint equations or short beams. This type of connection would require too many eigenmodes and result in a model that is not well conditioned.

Figure 10.1: "Connecting a Structure to an Interface Point" shows three different ways that you may attempt to attach an interface point to a structure. The first two examples (a and b) demonstrate valid methods of attachment. The third example (c) demonstrates a poor method of attachment that should not be used.

Figure 10.1 Connecting a Structure to an Interface Point



Each method depicted in the figure is described below.

- a) Constraint equations are connecting the interface point to the structure. This method is recommended because:
  - Force is distributed over an area.
  - A MASS21 element is used to define the six degrees of freedom of the interface point.
  - Moment loads are transmitted.

- b) A spider web of beams is connecting the interface point to the structure. This method is recommended (and preferred) because:
  - Force is distributed over an area.
  - No MASS21 element is necessary (because the beams supply the six degrees of freedom).
  - Moment loads are transmitted.
- c) One beam is used to connect the interface point to the structure. This is *not recommended* because:
  - The force is applied to the structure at a single node.
  - Solid elements do not have rotational degrees of freedom. Therefore, moments will not be properly transmitted from the interface point to the structure (a spider web scheme should be used).

10.3. Exporting to ADAMS

After building the model in ANSYS (including all interface points), the next step is to invoke the ANSYS-ADAMS Interface to create the modal neutral file, **Jobname.MNF**. Creation of this file is driven by an ANSYS command macro called **ADAMS.MAC**.

To start the interface, select the following GUI path.

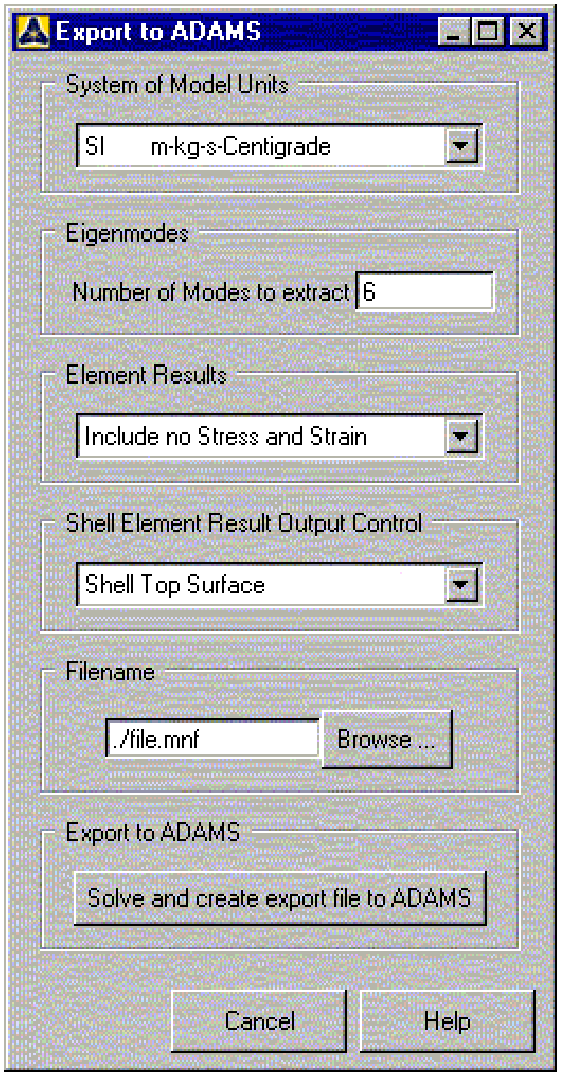
Main Menu> Solution> ADAMS Connection> Export to ADAMS

The Select Interface Points dialog box appears first. From this dialog box, you must select two or more interface points.

> *Note* — Do not choose too many interface points since one point gives rise to 6 degrees of freedom in ADAMS. Too many interface points may lead to huge files and models.

After you confirm your selection by picking OK, the Export to ADAMS dialog box appears.

Figure 10.2 Export to ADAMS Dialog Box



Export to ADAMS

System of Model Units: SI m-kg-s-Centigrade

Eigenmodes

Number of Modes to extract: 6

Element Results: Include no Stress and Strain

Shell Element Result Output Control: Shell Top Surface

Filename: ./file.mnf

Complete the following steps using this dialog box.

1. System of Model Units: The units used for the model is important to the ADAMS program, whereas ANSYS only requires that you use a consistent set of units. The units chosen will be written to the **.MNF** file and can be recalled with the ADAMS/Flex module. If no units are specified, ADAMS assumes that the same units were used in ANSYS as the ones chosen in the ADAMS model. See the **/UNITS** command for details. If you specify user defined units, a Define User Units dialog box will appear for you to input the conversion factors (for length, mass, force, and time) between SI units and your chosen units. Below is an example of user defined units in which the component has been modeled using millimeter, tonne (metric ton), newton, and second.

Length Factor =	1 meter/millimeter	= 1000
Mass Factor =	1 kilogram/tonne	= 0.001
Force Factor =	1 newton/newton	= 1
Time Factor =	1 second/second	= 1

2. Number of Modes to Extract: Input the number of normal modes to compute. Normal modes are the eigenmodes of the component with all degrees of freedom of all interface points fixed. The number of

normal modes depends on the frequency range of the excitation you will apply in your ADAMS model. You must choose a sufficient number of modes to represent your structure in that frequency range. In ADAMS, if you have chosen too many normal modes, you are able to deactivate eigenmodes based on the frequency or an energy criterion.

3. Element Results: Specify whether or not the program should write stress and/or strain results. This option has no effect on the output for beam elements. If you want to output stress and strain for only a subset of nodes, you should create a node component named "STRESS" before running the **ADAMS** command macro.
4. Shell Element Result Output Control: Specify the shell element output location (top, middle, bottom). This option has no effect on the output for solid elements and beam elements.
5. Filename: Specify a filename for the modal neutral file. The default name is **Jobname.MNF**. If a file with the chosen name exists, it will be moved to a file named **filename.MNFBAK**.
6. Export to ADAMS: Choose “Solve and create export file to ADAMS” to initiate the solution sequence. Static and normal modes are computed and all information required by ADAMS is written to the **.MNF** file specified above. Only the selected elements are considered. The current model is written to the database file **Jobname.DBMNF**.

Note — Note that the algorithm used to compute the **.MNF** file adds constraints to the interface points. If you create the **.MNF** file a second time using the same model in the same run, be sure to delete all constraints on the interface points (or resume the database file **Jobname.DBMNF**) before you run it again.

10.3.1. Exporting to ADAMS via Batch Mode

If you prefer to work in batch mode, you may choose to run the **ADAMS** command macro by command input. After building the model and defining interface points, use the following commands to compute the **.MNF** file.

```
/UNITS,Label        ! Specify the units chosen for modeling
NSEL,...            ! Select two or more interface points
SAVE                ! Save the model for a possible resume from
                    ! this point
ADAMS,NMODES,...    ! Activate ADAMS.MAC to compute the .MNF file
```

See the **ADAMS** command description for more information. When you use command input to compute the **.MNF** file, there is no option to change the file name. The default name of **Jobname.MNF** will be used.

10.3.2. Verifying the Results

It is a good practice to verify the correctness of the results after the **.MNF** file is created. Below are guidelines you can use to complete this task.

- Check the number of orthonormalized eigenmodes in the ANSYS output window. These eigenmodes are the result of an orthonormalization of the normal modes and the constraint modes. You should observe the following:
  - The number of modes equals the number of normal modes plus the number of constraint modes.
  - The first six modes are rigid body modes. These are marked with “(probable rigid body mode).” If there is a mode close to a rigid body mode but not marked, you may deactivate it later in the ADAMS program.
  - If a mode is marked with “Infinity. Possible mass singularity. Ignored,” check your model carefully. There might be a problem with the Interface points.

  - The first few modes are equal to the free-free eigenmodes of the component. You might want to verify this by doing a modal analysis: Set analysis option to **ANTYPE**,MODAL with **MODOPT**,LANB (Block Lanczos); activate the lumped mass approach with **LUMPM**,ON.

- Review the normal modes (load step 1) and the constraint modes (load step 2) in the General Postprocessor.
- Verify the transfer by doing a modal analysis of the component in ADAMS with all interface degrees of freedom fixed. Compare the results with the normal modes computed in ANSYS (load step 1).

10.4. Running the ADAMS Simulation

After you have verified that the **.MNF** file contains accurate information, you are ready to run an ADAMS simulation with a flexible component. Import the **.MNF** file into your ADAMS model and attach it to the rigid bodies using joints. To keep any numerical imbalance between inertia and external loads small, make sure you simulate your ADAMS model with high accuracy.

For general information about the ADAMS program and how to import flexible bodies, refer to the ADAMS manuals (provided by Mechanical Dynamics, Inc.), especially the documentation provided for the ADAMS/Flex product.

10.5. Transferring Loads from ADAMS to ANSYS

There are two ways to transfer loads and/or deformations from ADAMS to ANSYS:

- If the component is assumed to be rigid in ADAMS, you can transfer joint and external forces, accelerations, and rotational velocity acting on the component as described in Section 10.5.1: Transferring Loads on a Rigid Body.
- If the component is flexible, you can transfer the deformed shape of the component using the MSR toolkit from ADAMS. This type of transfer is not supported by the ANSYS-ADAMS Interface. See Section 10.5.2: Transferring the Loads of a Flexible Body for more information on this transfer method.

10.5.1. Transferring Loads on a Rigid Body

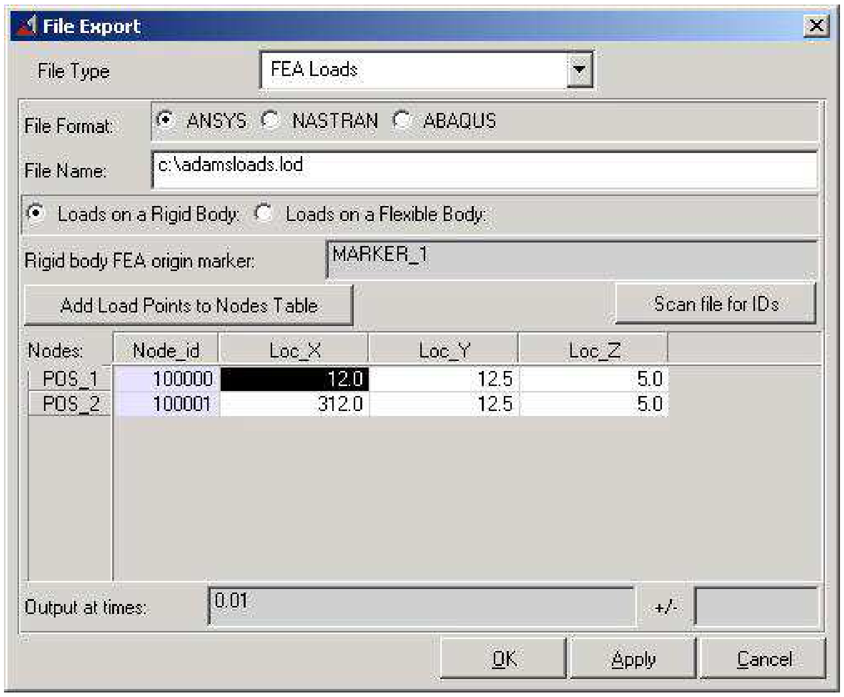
If you model your component as a rigid body in ADAMS, you can use the Export FEA Loads feature in ADAMS to export the loads to a file. This file can then be imported into ANSYS for a subsequent stress analysis.

If you model your component as a flexible body, ADAMS allows you to use the Export FEA Loads feature to transfer the loads, but the loads will be incomplete. Therefore, this load transfer procedure should generally not be used for flexible bodies. The transfer of loads may work, however, if the flexible bodies experience only small dynamic effects. If this is not the case, you may want to change the component temporarily to a rigid body (using the Modify utility in ADAMS) and run another simulation before you transfer the loads.

10.5.1.1. Exporting Loads in ADAMS

After performing an ADAMS simulation, you can export loads on a specific component at specific times. In ADAMS, choose **File> Export> FEA Loads** to access the ADAMS Export FEA Loads dialog box.

Figure 10.3 ADAMS Export FEA Loads Dialog Box



File Type: FEA Loads

File Format: [✓ ANSYS] NASTRAN ABAQUS

File Name: c:\adamsloads.lod

[✓ Loads on a Rigid Body] Loads on a Flexible Body

Rigid body FEA origin marker: MARKER_1

Nodes:	Node_id	Loc_X	Loc_Y	Loc_Z
POS_1	100000	12.0	12.5	5.0
POS_2	100001	312.0	12.5	5.0

Output at times: 0.01 +/-

Complete the following steps in this dialog box.

1. File Type: Choose FEA Loads
2. File Format: Choose ANSYS
3. File Name: Specify a name for the load file. The default extension is **.LOD**.
4. Specify whether you want to export loads on a rigid body or a flexible body.
   - Rigid body: You must define a marker on the body that has the same position and orientation relative to the body as the global origin does in the ANSYS model.
   - Flexible body: The marker is set automatically since this information is known from the **.MNF** file.
5. Click on “Add Load Points to Nodes Table.”
   - If you chose a rigid body, you can input the Node IDs of the nodes where the loads have to be applied in ANSYS.
   - If you chose a flexible body, ADAMS will automatically input the correct Node IDs.
6. Output at times: Specify at what time steps you want to export the loads.
7. Finally, ADAMS will ask you about the units. The units for export must be the same as those chosen for building the ANSYS model. If they are not the same, change them temporarily to the ANSYS units or scale the loads in the load file later.

Every time step in ADAMS is treated as a load step in ANSYS. In ADAMS versions up to 11.0.0, ADAMS writes the **LSWRITE** command before the load commands. Therefore, if you are using ADAMS version 11.0.0 or earlier, you must use a text editor to move the **LSWRITE** command to the end of each time step in the **.LOD** file.

The following loads will be included in the load file:

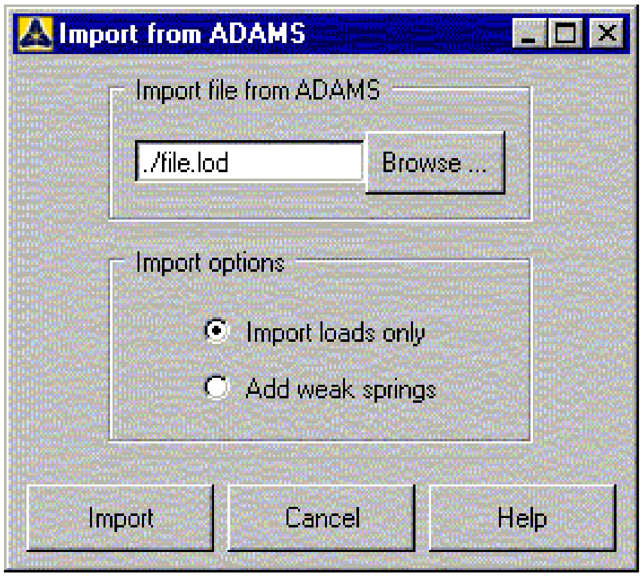
Joint forces (**F** command)
External forces (**F** command)
Accelerations and rotational velocities (**ACEL**, **OMEGA**, **DOMEGA** commands)

10.5.1.2. Importing Loads into ANSYS

After exporting the load file from ADAMS, you can use the ANSYS-ADAMS Interface to import the load file and initiate a static structural analysis. To access the Import from ADAMS dialog box, pick:

Main Menu> Solution> ADAMS Connection> Import fr ADAMS

Figure 10.4 Import from ADAMS Dialog Box



Complete the following steps in the dialog box:

1. Import file from ADAMS: Enter the name of the load file that was exported from ADAMS.
2. Import option: Theoretically, external forces and inertia forces are in equilibrium. Due to numerical errors or due to mass discrepancies between ADAMS and ANSYS, this is insufficient to prevent a rigid body motion of the component. Hence, you must constrain the component against rigid body motion in order to do a static structural analysis. The ANSYS-ADAMS Interface offers two import options to achieve this.

   - Import loads only. The program applies inertia loads and external forces to the structure according to the load file. For this option, you must manually add constraints to the ANSYS model that are compatible with the constraints used in the ADAMS model (if possible), or use common engineering sense to prevent rigid body motion.
   - Add weak springs: The program adds weak springs (COMBIN14 elements) to the corners of the bounding box of the component. (For more information, see the **WSPRINGS** command documentation). The weak springs prevent rigid body motion without influencing the stress results. (See Section 10.6.2: Adding Weak Springs for more information on how the program adds weak springs to the model.)

3. Import button: When you pick the Import button, one load step file is written per time step exported from ADAMS; existing load step files are deleted. If you chose the “Import loads only” option, you will have to start the static solution manually by issuing the **SOLVE** command for each load step. If you chose

the “Add weak springs” option, inertia relief is activated (**IRLF**,1) to compute accurate acceleration loads, and the static analysis is started automatically.

Note — If you use the import procedure a second time with the “Add weak springs” option, additional weak springs will be added to the model. This will have only a small influence on the results.

10.5.1.3. Importing Loads via Commands

If you prefer to work in batch mode, you may choose to import the load file and initiate the solution by command input. After exporting the load file from ADAMS, use the following commands to read the load file and initiate the static solution.

```
/INPUT,...             ! Read the load file from ADAMS
WSPRINGS (or D)        ! Apply weak springs (or use D commands to
                       ! apply rigid body constraints)
*DO,par,ival,fval,inc  ! Specify load steps to solve
  IRLF,1               ! Activate inertial relief to achieve higher accuracy
                       ! (this step is optional)
  LSREAD,par           ! Read load step
  SOLVE                ! Solve model
*ENDDO
```

Every subsequent call of the **WSPRINGS** command will apply weak springs. Therefore, this command may be omitted when importing new loads.

10.5.1.4. Reviewing the Results

When the structural analysis is complete, you review the results as you would for any linear structural analysis.

When using the weak springs option with inertia relief check that:

- The accelerations ANSYS computed for inertia relief are small compared to the applied acceleration loads from ADAMS (**ACEL**, **OMEGA**, **DOMEGA**). Issue the command **IRLIST** to view the inertia relief accelerations (translational and rotational).
- The forces in the springs are small compared to the external forces. The forces in the springs can be viewed by listing the reaction forces. Use the **PRRSOL** command to list reaction forces.

The external forces have to be balanced by the applied inertia forces only. If one of the above is not true, there is an imbalance in your model that must be removed. Check your ANSYS and ADAMS models, respectively.

10.5.2. Transferring the Loads of a Flexible Body

If you want to model a flexible component in the ADAMS program and perform a subsequent stress analysis, you may want to use the Modal Stress Recovery (MSR) toolkit provided by Mechanical Dynamics, Inc. Using the features of this toolkit, it is possible to transfer the loads of a flexible component from ADAMS to ANSYS for stress analysis. This toolkit provides several strategies for interfacing with ANSYS:

- Export Time-Domain Displacements: If you have only a few time steps to analyze, this is a fast option. The displacement of every node of the component is written directly into an ANSYS input file. This file can then be imported using the **/INPUT** command. A simple static analysis can be started in ANSYS after the import of this file.
- Export Mode Shapes: The toolkit writes an ANSYS input file that can be used to compute the orthonormalized or unorthonormalized eigenmodes of the component. By using the Export of Modal Coordinates option, these eigenmodes can be scaled in ANSYS, and the stresses in the component can be computed for every time step.

- Export Nodal Loads: Using this feature, you can write an ANSYS input file to perform stress recovery as a superposition of unit force load steps. This method ignores the inertia load contribution to the flexible body deformation, so it may be inaccurate when interpreting dynamic effects.

Note — The MSR toolkit features described here are not supported by the ANSYS-ADAMS Interface.

10.6. Methodology Behind the ANSYS-ADAMS Interface

Some tasks performed by the ANSYS-ADAMS Interface involve substantial “behind-the-scenes” work. Two tasks in particular fall in this category: the creation of the modal neutral file (**Jobname.MNF**) and the addition of weak springs via the **WSPRINGS** command. The following sections provide details on how ANSYS performs those tasks.

10.6.1. The Modal Neutral File

The algorithm used to create the modal neutral file (**.MNF**) is based on a formulation called component mode synthesis (also known as dynamic substructuring). ADAMS uses the approach of Craig Bampton with some slight modifications. According to this theory, the motion of a flexible component with interface points is spanned by the interface constraint modes and the interface normal modes. Constraint modes and interface normal modes together are referred to as component modes.

Because the algorithm relies on the component mode synthesis method, which is based on the modal analysis, only linear properties are considered during the formation of the modal neutral file. All geometric and physical nonlinearities are ignored. If significant geometric nonlinear effects are present in your component, you must subdivide the component into several smaller components and transfer each one separately. You can then assemble the subdivided components in ADAMS to form a flexible component with geometric nonlinearity.

The modal neutral file contains the following information:

- Header information: date, ANSYS version, title, **.MNF** version, units
- Body properties: mass, moments of inertia, center of mass
- Reduced stiffness and mass matrices in terms of the interface points
- Interface normal modes (the user requests the number of modes generated)
- Interface constraint modes

To supply the above information, ANSYS does a sequence of analyses through a macro file called **ADAMS.MAC** (see the **ADAMS** command) in order to generate the required interface constraint modes and interface normal modes.

10.6.2. Adding Weak Springs

During the import of loads from ADAMS to ANSYS, you can instruct ANSYS to add weak springs to the model via the **WSPRINGS** command. The weak springs are added to the corners of the bounding box of the component. The stiffnesses of the springs are many orders of magnitude less than the stiffness of the structure, and hence prevent rigid body motion without influencing the stress results. The program takes the following steps when adding weak springs:

- To define the bounding box, the algorithm finds the nodes with the maximum and minimum coordinates. Six nodes are created by this approach. These nodes define the bounding box of the component. Because a three dimensional model is required for this approach, simple beam models that only have an extension in one dimension cannot be handled by the weak springs options.

- COMBIN14 elements are used to link the six nodes of the bounding box to the ground in all three translational directions. The stiffness of the spring element is computed as $k = (E_{mean})(10^{-6})$, where E_{mean} is the mean value of all moduli of elasticity defined. This is a very rough approach, but one which has proven to be effective in practical applications. If the stress results are influenced by the springs, you can change the stiffness by changing the corresponding COMBIN14 real constant.

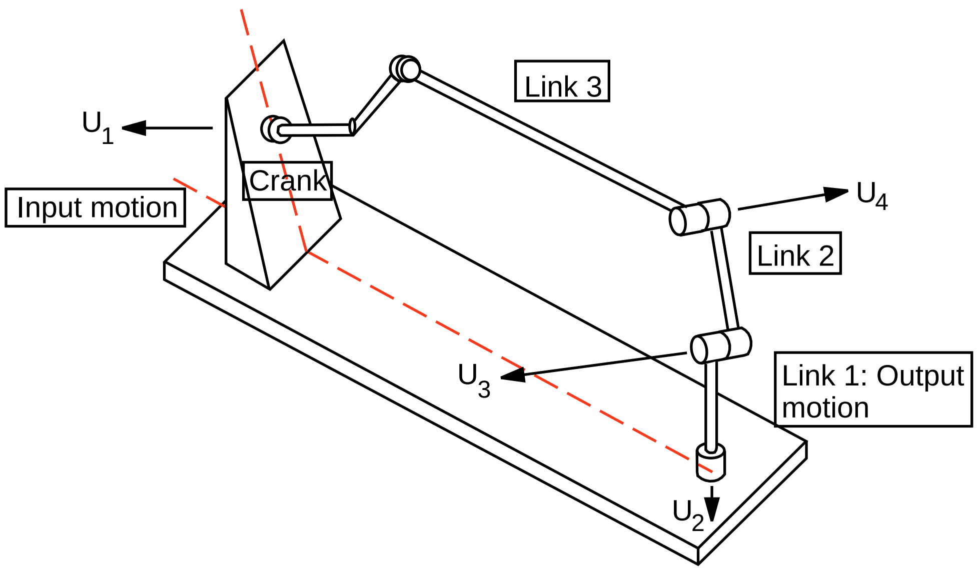
10.7. Sample Rigid Body Dynamic Analysis

This sample analysis demonstrates how to model a flexible component in ANSYS and export the flexible body information to a file for use in ADAMS. The example also provides brief instructions on how to perform the rigid body dynamic analysis in ADAMS, and details on how to transfer the loads from ADAMS to ANSYS in order to perform a stress analysis.

10.7.1. Problem Description

In the linkage assembly shown below, Link3 is a flexible component. Link3 is modeled as a rectangular rod in ANSYS using SOLID45 elements. The joints in ADAMS will be attached to interface points (nodes) at the middle of the holes at either end of Link3. These middle points are connected to the cylindrical joint surfaces by a spider web of BEAM4 elements.

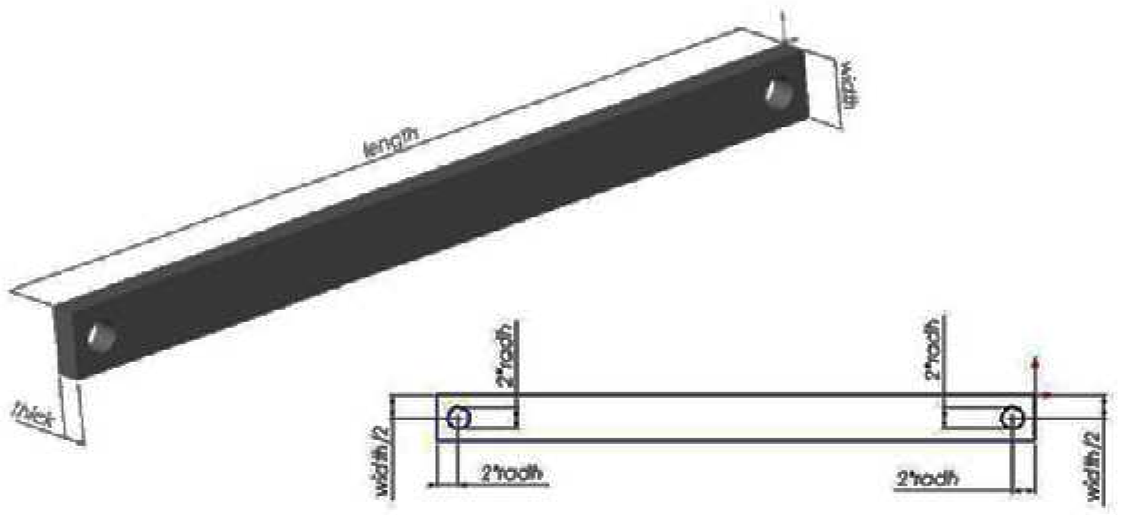
Figure 10.5 Linkage Assembly



10.7.2. Problem Specifications

The figure below shows the Link3 component as it is modeled in ANSYS.

Figure 10.6 Link3 Component



The following are dimensions and properties for the Link3 component.

Radius of holes (radh) = 6mm
Width of rectangular rod (width) = 25mm
Thickness of rectangular rod (thick) = 10mm
Length of rectangular rod (length) = 300mm + 4*Radius of holes = 324mm
Young's modulus for rod = 7.22 x 10^4 MPa
Poisson's ratio for rod = 0.34
Density of rod = 2.4 x 10^{-9} tons/mm^3
Young's modulus for beams = 2.1 x 10^5 MPa
Poisson's ratio for beams = 0.3
Density of beams = 0.1 x 10^{-9} tons/mm^3

10.7.3. Command Input

```
/BATCH,list
/FILNAME,adamsout        ! Define jobname
/TITLE,Export flexible component to ADAMS
!
/PREP7             ! Enter preprocessor
!
! Define Parameters of rectangular rod
radh=6             ! Radius of the holes in the rod
thick=10           ! Rod thickness
width=25           ! Rod width
length=300+4*radh  ! Rod length
! Build geometry
RECTNG,0,length,0,width
CYL4,2*radh,width/2,radh
CYL4,length-2*radh,width/2,radh
ASBA,1,2
ASBA,4,3
VEXT,1, , ,0,0,thick
!
ET,1,solid45       ! Define SOLID45 as element type 1
ET,2,beam4         ! Define BEAM4 as element type 2
!
MP,EX,1,7.22e4     ! Material of the rectangular rod
```

```
MP,PRXY,1,0.34
MP,DENS,1,2.4e-9
!
MP,EX,2,2.1e5      ! Material of the beams used for the spider web
MP,PRXY,2,0.3
MP,DENS,2,0.1e-9
!
R,1,78.528,490.67,490.67,10,10,981.34,    ! Real constant for BEAM4
RMORE,,,0.85716,0.85716,
!
TYPE,1             ! Set element type attribute pointer to 1
MAT1,1             ! Set material attribute pointer to 1
ESIZE,thick/3,0,   ! Define global element size
VSWEEP,1           ! Mesh rod
!
! Define interface points: numbers must be higher than highest
! node number already defined
N,100000,2*radh,width/2,thick/2          ! Define interface point 1
N,100001,length-2*radh,width/2,thick/2   ! Define interface point 2
!
NWPAVE,100000      ! Set working plane to interface point 1
WPSTYL,,,,,,1      ! Set working plane type to cylindrical
CSYS,4             ! Activate working plane
NSEL,S,LOC,X,radh  ! Select nodes on cylindrical hole
NSEL,A,,,100000    ! Also select interface node
!
! Generate spider web of beams
*GET,nmin,node,,num,min
*GET,nnum,node,,count
*SET,jj,0
TYPE,2
MAT,2
REAL,1
*DO,jj,1,nnum-2
 E,100000,nmin
 NSEL,u,,,nmin
 *GET,nmin,node,,num,min
*ENDDO
!
ALLS
!
NWPAVE,100001      ! Set working plane to interface point 2
WPSTYL,,,,,,1      ! Set working plane type to cylindrical
CSYS,4             ! Activate working plane
NSEL,S,LOC,X,radh  ! Select nodes on cylindrical hole
NSEL,A,,,100001    ! Also select interface node
!
! Generate spider web of beams
*GET,nmin,node,,num,min
*GET,nnum,node,,count
*SET,jj,0
TYPE,2
MAT,2
REAL,1
*DO,jj,1,nnum-2
 E,100001,nmin
 NSEL,u,,,nmin
 *GET,nmin,node,,num,min
*ENDDO
!
ALLS
!
/UNITS,MPA                        ! Define units used: millimeter
                                  ! megagram, second, newton
SAVE                              ! Save database
NSEL,s,,,100000,100001            ! Select interface points
ADAMS0,20,1                       ! Start ADAMS macro,
! adamsout.mnf is written
FINISH
/EXIT,nosave
```

At this point you may import the **adamsout.mnf** file into your ADAMS model and perform a rigid body dynamics simulation. The ADAMS model should consist of the components shown in Figure 10.5: “Linkage Assembly”. After the simulation is done, export the loads acting on the Link3 component at five arbitrary time steps. Name the load file **loads.lod**.

Once you have exported the load file, you can perform a stress analysis for Link3 in ANSYS using the command input shown below.

```
RESUME,adamsout,db      ! Resume model
/FILNAM,adamsin         ! Change jobname
/TITLE,Import loads from ADAMS     ! Change title
!
WSPRINGS                ! Create weak springs
!
! Enter Solution and solve all load steps
/SOLU
/INPUT,loads,lod        ! Read in 5 load steps written by ADAMS
*DO,i,1,5               ! Use a do loop to solve each load step
  LSREAD,i              ! Read in load step
  IRLF,1                 ! Activate inertia relief
  SOLVE                 ! Solve current load step
*ENDDO
!
/POST1                  ! Enter the general postprocesser
! Write deformation and equivalent stress to graphics file
/VIEW,1,1,1,1
/AUTO,1
EPLOT
/TYPE,1,4
/SHOW,
EPLOT
*DO,i,1,5
  SET,i
  PLNSOL,u,sum
  PLNSOL,s,eqv
*ENDDO
/SHOW,term
FINISH
/EXIT,nosave
```

Chapter 11: Element Birth and Death

If material is added to or removed from a system, certain elements in your model may become "existent" or "nonexistent.” In such cases, you can employ element *birth* and *death* options to deactivate or reactivate selected elements, respectively. Applicable element types are listed in Table 11.1: “Elements Supporting Birth and Death”.

The element birth and death feature is useful for analyzing excavation (as in mining and tunneling), staged construction (as in shored bridge erection), sequential assembly (as in fabrication of layered computer chips), and many other applications in which you can easily identify activated or deactivated elements by their known locations.

The birth and death feature is available in the ANSYS Multiphysics, ANSYS Mechanical, and ANSYS Structural products.

Table 11.1 Elements Supporting Birth and Death

LINK1	MASS21	SOLID45	SOLID70	SOLID98	CONTA171
PLANE2	BEAM23	PLANE53	MASS71	PLANE121	CONTA172
BEAM3	BEAM24	BEAM54	PLANE75	SOLID122	CONTA173
BEAM4	PLANE25	PLANE55	PLANE77	SOLID123	CONTA174
SOLID5	MATRIX27	SHELL57	PLANE78	SHELL131	CONTA175
LINK8	LINK31	PIPE59	PLANE82	SHELL132	LINK180
LINK10	LINK32	PIPE60	PLANE83	SHELL143	SHELL181
LINK11	LINK33	SOLID62	SOLID87	SURF151	PLANE182
PLANE13	LINK34	SHELL63	SOLID90	SURF152	PLANE183
COMBIN14	PLANE35	SOLID64	SOLID92	SURF153	SOLID185
PIPE16	SHELL41	SOLID65	SHELL93	SURF154	SOLID186
PIPE17	PLANE42	PLANE67	SOLID95	SHELL157	SOLID187
PIPE18	SHELL43	LINK68	SOLID96	TARGE169	BEAM188
PIPE20	BEAM44	SOLID69	SOLID97	TARGE170	BEAM189

User elements may also be given the birth and death capability. See the *Guide to ANSYS User Programmable Features* for more information on user elements

In some circumstances, an element's birth and death status may be dependent upon an ANSYS-calculated quantity, such as temperature, stress, strain, etc. You can issue commands such as **ETABLE** and **ESEL** to determine the value of such quantities in selected elements, and to change the status (melted, solidified, ruptured, etc.) of those elements accordingly. This capability is useful for modeling effects due to phase changes (as in welding processes, when structurally inactive molten material solidifies and becomes structurally active), failure-surface propagation, and other analysis-dependent element changes.

11.1. How Does Element Birth and Death Work?

To achieve the "element death" effect, the ANSYS program does not actually *remove* "killed" elements. Instead, it *deactivates* them by multiplying their stiffness (or conductivity, or other analogous quantity) by a severe reduction factor (**ESTIF**). This factor is set to 1.0E-6 by default, but can be given other values. (For more information, see Section 11.3.2: Apply Loads and Obtain the Solution.)

Element loads associated with deactivated elements are zeroed out of the load vector, however, they still appear in element-load lists. Similarly, mass, damping, specific heat, and other such effects are set to zero for deactivated elements. The mass and energy of deactivated elements are not included in the summations over the model. An element's strain is also set to zero as soon as that element is killed.

In like manner, when elements are "born," they are not actually *added* to the model; they are simply *reactivated*. You must create all elements, including those to be born in later stages of your analysis, while in PREP7. You cannot create new elements in SOLUTION. To "add" an element, you first deactivate it, then reactivate it at the proper load step.

When an element is reactivated, its stiffness, mass, element loads, etc. return to their full original values. Elements are reactivated with no record of strain history (or heat storage, etc.); however, initial strain defined as a real constant (for elements such as LINK1) will not be affected by birth and death operations.

Unless large-deformation effects are activated (**NLGEOM**,ON), some element types will be reactivated in their originally specified geometric configuration. (Large-deformation effects should be included to obtain meaningful results.)

Thermal strains are computed for newly-activated elements based on the current load step temperature and the reference temperature. Thus, newborn elements with thermal loads may not be stress-free as intended. The material property REFT can be used instead of the global TREF to specify material-dependent reference temperatures, allowing you to specify the activation temperature as a stress-free temperature.

11.2. Element Birth and Death Usage Hints

The following guidelines apply to analyses employing the element birth and death capability of ANSYS:

- Constraint equations (**CE**, **CEINTF**, etc.) cannot be applied to inactive DOFs. Inactive DOFs occur when a node has no active ("alive") elements attached to it.
- You can model stress-relieving operations (such as annealing) by deactivating and then reactivating elements.
- In nonlinear analyses, be careful not to deactivate or reactivate elements in such a way as to create singularities (such as sharp re-entrant corners in a structural analysis) or sudden large changes in stiffness. Such situations are likely to cause convergence difficulties.
- If the model is entirely linear--that is, if the model has no contact or other nonlinear element present and the material is linear--except for birth and death elements, ANSYS treats the analysis as linear and will therefore not activate optimized defaults (**SOLCONTROL**,ON) applicable to nonlinear solutions.
- The full Newton-Raphson option with adaptive descent activated (**NROPT**,FULL,,ON) often yields good results in analyses employing element birth and death.
- You can retrieve a parameter whose value will indicate the status (active or inactive) of an element (***GET**,*Par*,ELEM, n, ATTR, LIVE) This parameter could be used in APDL logical branching (***IF**, etc.) or in other applications for which you need to monitor the birth-and-death status of an element.
- The load-step file method (**LSWRITE**) for solving multiple load steps cannot be used with the birth-death option, because it will not write the status of deactivated or reactivated elements to the load step file. Birth and death analyses having multiple load steps must therefore be performed using a series of explicit **SOLVE** commands.

11.2.1. Changing Material Properties

You might be tempted to deactivate or reactivate elements by changing their material properties via the **MPCHG** command.

You must proceed cautiously if you attempt such a procedure. The safeguards and restrictions applying to "killed" elements do not apply to elements that have had their material properties changed in the solution phase of the analysis. (Element forces will not be automatically zeroed out; nor will strains, mass, specific heat, etc.) Many problems can result from careless use of the **MPCHG** command. For example, if you reduce an element's stiffness to almost zero, but retain its mass, it could result in a singularity if subjected to acceleration or inertial effects.

One application of the **MPCHG** command would be in modeling construction sequences in which the strain history of a "born" element is maintained. Using **MPCHG** in such cases will enable you to capture the initial strain experienced by elements as they are fitted into the displaced nodal configuration.

11.3. Employing Birth and Death

You can apply element birth and death behavior to most static and nonlinear transient analyses using the same basic procedures described in the various analysis guides.

Modify your basic analysis procedure as follows to incorporate the element birth and death feature:

11.3.1. Build the Model

While in **/PREP7**, create *all* elements - even those that will not be activated until later load steps. You cannot create new elements outside of **/PREP7**.

11.3.2. Apply Loads and Obtain the Solution

For all analyses employing element birth and death, perform the following actions in the solution (**SOLU**) phase:

11.3.2.1. Define the First Load Step

In the first load step, you must choose the analysis type and all appropriate analysis options via the **ANTYPE** command.

For a structural analysis, activate large-deflection effects via the **NLGEOM**,ON command.

For all birth and death applications, set the Newton-Raphson option to full explicitly in the first load step via the **NROPT** command. (The ANSYS program cannot predict the presence of an **EKILL** command in a subsequent load step.) Deactivate (**EKILL**) all of the initially inactive elements that you intend to add (reactivate) in later load steps.

Elements are deactivated (or activated) in the *first* substep of the load step, and maintain that status through the rest of the load step. The default reduction factor used as a stiffness multiplier might not suffice for some problems; sometimes, you may need to use a more severe reduction factor. To provide a new value for the reduction factor, issue the **ESTIF** command.

Nodes not connected to any active elements may "float," or pick up stray degree-of-freedom (DOF) responses. You may want to constrain inactive DOFs (**D**, **CP**, etc.) in some cases to reduce the number of equations to be solved and to avoid ill-conditioning. Constraining inactive DOFs can become more important for cases in which you want to reactivate elements with a specific shape (or temperature, etc.). If so, remove the artificial constraints when you reactivate elements, and remove nodal loads from inactive DOFs (that is, at nodes not connected to any active elements). Similarly, you must specifically add nodal loads (if any) to reactivated DOFs.

11.3.2.1.1. Sample Input for First Load Step

Part of your input listing could look like this for your first load step:

```
! First load step
TIME,...            ! Sets TIME value (optional for static analyses)
NLGEOM,ON           ! Turns large-deflection effects on
NROPT,FULL          ! You must explicitly set the Newton-Raphson option
ESTIF,...           ! Sets non-default reduction factor (optional)
ESEL,...            ! Selects elements to be deactivated in this load step
EKILL,...           ! Deactivates selected elements
ESEL,S,LIVE         ! Selects all active elements
NSLE,S              ! Selects all active nodes
NSEL,INVE           ! Selects all inactive nodes (those not attached to any
                    ! active elements)
D,ALL,ALL,0         ! Constrains all inactive DOFs (optional)
NSEL,ALL            ! Selects ALL nodes
ESEL,ALL            ! Selects ALL elements
D,...               ! Adds constraints as appropriate
F,...               ! Adds nodal loads to active DOFs as appropriate
SF,...              ! Adds element loads as appropriate
BF,...              ! Adds body loads as appropriate
SAVE
SOLVE
```

11.3.2.2. Define Subsequent Load Steps

In the remaining load steps, you can deactivate and reactivate elements as desired. As before, be sure to apply and delete constraints and nodal loads as appropriate.

To deactivate and reactivate elements, issue the **EKILL** and **EALIVE** commands, respectively.

11.3.2.2.1. Sample Input for Subsequent Load Steps

The following simplified input listing demonstrates how you might deactivate and reactivate elements:

```
! Second (or subsequent) load step:
TIME,...
ESEL,...
EKILL,...                   ! Deactivates selected elements
ESEL,...
EALIVE,...                  ! Reactivates selected elements
...
FDELE,...                   ! Deletes nodal loads at inactive DOFs
D,...                       ! Constrains inactive DOFs
...
F,...                       ! Adds nodal loads as appropriate to active DOFs
DDELE,...                   ! Deletes constraints from reactivated DOFs
SAVE
SOLVE
```

11.3.3. Review the Results

Typically, you will follow standard procedures when postprocessing an analysis containing deactivated or reactivated elements.

Be aware that "killed" elements are still present in your model, even though they make an insignificant contribution to the stiffness (conductivity, etc.) matrix; therefore, they *are included* in element displays, output listings, etc. For example, deactivated elements are included in nodal results averaging (via the **PLNSOL** command) and will "smear" the results. Ignore the entire element printout for deactivated elements because many items computed make little physical sense.

To remove deactivated elements for element displays and other postprocessing operations, issue the **ESEL** command.

11.3.4. Use ANSYS Results to Control Birth and Death

At times, you will not explicitly know the location of elements that you need to deactivate or reactivate. For example, if you want to "kill" melted elements in a thermal analysis (that is, to model the removal of melted material), you will not know the location of those elements beforehand; you will need to identify them on the basis of their ANSYS-calculated temperatures. When the decision to deactivate or reactivate an element depends on the value of an ANSYS result item (such as temperature, stress, strain, etc.), you can use commands to identify and select the critical elements.

To identify the critical elements, issue the **ETABLE** command. To select the critical elements, issue the **ESEL** command.

You could then deactivate or reactivate the selected elements. To deactivate the selected elements, issue the **EKILL**,ALL command. To reactivate the selected elements, issue the **EALIVE**,ALL command.

> *Note* — You could also use the ANSYS Parametric Design Language to write a macro to perform such an operation. See the *ANSYS APDL Programmer's Guide* for more information.

11.3.4.1. Sample Input for Deactivating Elements

The following simplified input listing demonstrates how you might deactivate elements that rupture when their total strain has exceeded some critical value:

```
/SOLU                   ! Enter SOLUTION
...                     ! Typical solution procedure
SOLVE
FINISH
!
/POST1                  ! Enter POST1
SET,...
ETABLE,STRAIN,EPTO,EQV  ! Store total equivalent strain in ETABLE
ESEL,S,ETAB,STRAIN,0.20 ! Select all elements with total equivalent strain
                        !  greater than or equal to 0.20
FINISH
!
/SOLU                   ! Re-enter SOLUTION
ANTYPE,,REST
EKILL,ALL               ! Deactivate selected (overstrained) elements
ESEL,ALL                ! Restore full element set
...                     ! Continue with solution
```

11.4. Where to Find Examples

The *ANSYS Verification Manual* consists of test case analyses demonstrating the analysis capabilities of the ANSYS program. While these test cases demonstrate solutions to realistic analysis problems, the *ANSYS Verification Manual* does not present them as step-by-step examples with lengthy data input instructions and printouts; however, if you have even limited finite-element experience, you should have no trouble understanding the problems by reviewing each test case's finite-element model, input data and accompanying comments.

The *ANSYS Verification Manual* contains the following test case featuring element birth and death:

VM194 - Element Birth/Death in a Fixed Bar with Thermal Loading

Chapter 12: User-Programmable Features and Nonstandard Uses

The ANSYS program's open architecture allows you to link it to your own FORTRAN routines and subroutines. In fact, some standard ANSYS features began as user-programmed features.

Typically, you can obtain good results with the ANSYS program when you exercise documented features using standard, recommended procedures. In some cases, however, you may need to employ nonstandard procedures that ANSYS, Inc. Quality Assurance has not or cannot fully test.

Following is an introduction to both user-programmable features (UPFs) and nonstandard uses of the ANSYS program.

12.1. User-Programmable Features

User-programmable features (UPFs) are ANSYS capabilities for which you can write your own FORTRAN routines. UPFs allow you to customize the ANSYS program to your needs, which may be a user-defined material-behavior option, element, failure criterion (for composites), and so on. You can even write your own design-optimization algorithm that calls the entire ANSYS program as a subroutine. UPFs are available in the ANSYS Multiphysics, ANSYS Mechanical, ANSYS Structural, ANSYS PrepPost, and ANSYS University (Advanced and Research versions) products. For detailed information, see the *Guide to ANSYS User Programmable Features*.

Caution: By linking in your own FORTRAN routines, you are creating a custom, site-specific version of the ANSYS program. When you use UPFs, you are using ANSYS in a nonstandard way, one that ANSYS, Inc. verification testing does not cover. *You are responsible* for verifying that the results produced are accurate and that the routines you link to ANSYS do not adversely affect other, standard areas of the program.

Exercise care when using UPFs on parallel systems. Do not use the **/CONFIG** command or a **config81.ans** file to activate parallelism on a system with UPFs.

12.1.1. Employing User-Programmable Features

UPFs can range from a simple element output routine for custom output to a much more complex user element or user-optimization algorithm; therefore, it is difficult to present the process without describing specific programming details. This section presents a *general* sequence of steps to follow. The *Guide to ANSYS User Programmable Features* contains more detail on UPFs.

A typical UPF involves the following steps:

1. Design and program the desired user routine in FORTRAN 90. The source codes for all user routines are available on your ANSYS distribution medium. Most of them demonstrate at least simple functionality.
2. Compile and link your user routine into the ANSYS program. The *Guide to ANSYS User Programmable Features* describes how to do this on your system.
3. Verify that the changes you have made do not affect other, standard ANSYS features. (One way to do so is by running a set of *ANSYS Verification Manual* problems.)
4. Verify the user routine using whatever procedures you feel are adequate.

The ANSYS program activates some UPFs (such as user elements) automatically when you use them. For example, to activate a user element, all you need to do is specify it as one of the element types in the model (via the **ET** command), set the element type attribute pointer (via the **TYPE** command)), and define elements using the solid modeling (**AMESH**, **VMESH**, etc.) or direct generation (**ET**, etc.) method.

For other UPFs, you must issue the **USRCAL** command to activate them. If you fail to issue the command, standard ANSYS logic applies by default.

For example, when you apply a convection load, the default is to use standard ANSYS logic *even if you have linked a user convection routine*. You must activate the appropriate user routine with the **USRCAL** command if you want the user convection routine to be used. Refer to the **USRCAL** command description for a list of user routines affected by the command. Use the **NSVR** command to define the number of extra variables that need to be saved for such user-programmable element options as user plasticity. (The **NSVR** command has no equivalent GUI path.)

Another useful command is **/UCMD**, which allows you to create your own command from a user routine. Suppose you link in a user routine for a parabolic distribution of pressure. If you name the routine USER*nn* (where *nn* = 01 to 10), you can create your own command to call the routine:

```
/UCMD,PARAB,1
```

PARAB now becomes a valid ANSYS command that simply calls the user routine USER01. You can call up to ten such user routines as commands. By including **/UCMD** commands in your start-up file (**start81.ans**), you can make the user routines available in all of your ANSYS sessions.

12.1.2. Types of User-Programmable Features Available

This list provides a brief description of the types of UPFs available for use in the ANSYS program:

User element - A user-written element type that you can add to the ANSYS element library and use as you would any other element. You can create up to six independent element types (USER100 - USER105). Example copies of the routines for MASS21, the structural mass element, and LINK8, the 3-D spar, are included in ANSYS distribution media for demonstration purposes.

User element coordinate system orientation - Available for the following element types: SHELL43, SHELL63, SHELL91, SHELL93, SHELL99 , SHELL181, SOLID46, and SOLID64. For the layered elements (SOLID46, SHELL91, and SHELL99), you can also customize the orientation of a given layer.

User real constants - Elements COMBIN7 and COMBIN37 allow the input real constants to be modified based upon your own nonlinear function.

User thickness - Available for SHELL181.

User stresses - Available for PLANE2, PLANE42, SOLID45, PLANE82, SOLID92, SOLID95, LINK180, SHELL181, PLANE182, PLANE183, SOLID185, SOLID186, SOLID187, BEAM188, and BEAM189.

User plasticity law - Allows you to calculate plastic strains and form the tangent stress-strain matrix at an integration point based on your own plasticity law.

User creep equation - Allows you to specify your own creep equation.

User swelling law - If you need to account for swelling in an analysis (such as due to neutron bombardment), you must write the appropriate swelling law as a user routine. No built-in swelling laws are available in the ANSYS program.

User hygrothermal growth - Allows you to induce growth caused by moisture content, and is available for the SHELL91 element.

User hyperelasticity - Available for hyperelastic elements HYPER56, HYPER58, HYPER74, and HYPER158.

User failure criteria - Available for the layered elements SOLID46 and SHELL99. Up to six user-defined failure criteria can be supplied.

User viscosity - You can define viscosity as a function of pressure, temperature, position, time, velocity, and velocity gradients for FLUID141 and FLUID142.

User loads - Body loads such as temperatures, heat generations, and fluences (such as neutron flux), as well as surface loads such as pressures, convections, heat fluxes and charge density may be defined by way of user-written logic.

User load vector - This allows you to create a complex load vector for the frequency domain logic of the PIPE59 element. You can use it to represent hydrodynamic forces.

ANSYS as a subroutine - You can call the entire ANSYS program as a subroutine in your own program, such as a user-written design optimization algorithm.

User optimization - You can replace the ANSYS optimization logic with your own algorithm and termination logic.

User access at the beginning and end of each ANSYS run solution, load step, substep, and equilibrium iteration - Allows you to evaluate results and perform any desired calculations during solution.

12.2. Nonstandard Uses of the ANSYS Program

The ANSYS program endures a rigorous verification testing plan before its release. You can be reasonably assured of obtaining good results when you exercise documented features using standard, recommended procedures. In some situations, however, you may need to employ nonstandard procedures or techniques that have not been or cannot be fully tested by ANSYS, Inc. because of their very nature. (An example is employing user-programmable features.) Be aware that verifying the results in such cases is *your responsibility*.

12.2.1. What Are Nonstandard Uses?

The results of nonstandard uses of the ANSYS program cannot be predicted; therefore, ANSYS, Inc.'s testing cannot fully cover such uses. Although ANSYS, Inc. does not discourage nonstandard uses, you must exercise caution and use your engineering judgment when doing so. For example, if you program your own element and use it in an ANSYS analysis, the results depend primarily on how well you programmed the element. In such cases, you must verify the results and make sure that other, standard areas of the program are not adversely affected.

Following is a partial list of nonstandard ANSYS features and uses:

- User programmable features (UPFs) - writing your own user routines, linking them into the ANSYS executable, and using them in an analysis. UPFs are described earlier in this chapter.
- Reading into the ANSYS program an ANSYS file which was created or modified external to the ANSYS program, for example, a results file or a superelement file created by you or by another program.
- High-risk capabilities such as the following:
  - Changing element real constants during the solution phase in between load steps. Depending on the element type being used, the element may not properly use the updated real constant value.
  - Deactivating the cross-reference checking of the solid model (via the **MODMSH**,NOCHECK command).
  - Turning off element shape-checking (via the **SHPP**,OFF command).

- Using undocumented features, such as an element option not documented in the *ANSYS Elements Reference* or a command argument not mentioned in the *ANSYS Commands Reference*. Undocumented features are, by definition, unsupported and unverified; use them with caution.

If the ANSYS program can detect the use of a nonstandard feature, it will often issue a warning message to that effect.

12.2.2. Hints for Nonstandard Use of ANSYS

Follow these guidelines when you need to use the ANSYS program in a nonstandard manner:

- Use your engineering judgment and carefully review the analysis results.
- Do not assume that other, standard areas of the program are not affected. Run a few test problems to verify.
- If you need to contact ANSYS Technical Support concerning an analysis involving nonstandard use of the ANSYS program, be sure to mention the nature and extent of the nonstandard feature that you employed.

For detailed information on UPFs, see the *Guide to ANSYS User Programmable Features*.

Chapter 13: Improving ANSYS Performance and Parallel Performance for ANSYS

Solving a large model with millions of DOFs can require many hours. To decrease processing time, ANSYS offers several options for distributing model-solving power over multiple processors. These options fall under two primary configurations: shared memory architecture and distributed memory architecture.

Shared memory architecture runs a solution over multiple processors on a single machine. Typically these multiprocessor machines would be servers. The shared memory options in ANSYS include:

- ANSYS solvers such as the PCG or ICCG run over multiple processors but share the same memory address. No additional add-on licenses are required for these solvers. While these solvers feature greater ease-of-use, they have limited scalability.
- The Parallel Performance for ANSYS add-on using the AMG solver. Use this option to solve static or transient analyses over multiple processors on the same system to speed up processing time, or to solve ill-conditioned problems that have difficulty converging with the conventional PCG or ICCG solvers.

Distributed memory architecture runs a solution over multiple processors on a single or on multiple machines. It decomposes large problems into smaller domains, transfers the domains to each processor, solves each domain, and creates a complete solution to the model. Because the solutions are running in parallel, the whole model solution takes much less time to solve. The memory required is also distributed over multiple systems. The distributed memory options in ANSYS include:

- The Parallel Performance for ANSYS add-on using the distributed solvers (DDS, DPCG, DJCG). This option affords better scalability and better memory distribution than a shared memory architecture. See Section 13.2.2: About the Distributed Solvers (DDS, DPCG, DJCG) for specific recommendations for each of the distributed solvers.

The distributed solvers can run under both the shared or the distributed memory architecture, running on a shared memory machine by treating each processor in a distributed manner. However, the shared memory solvers do not run under the distributed memory architecture.

Note — If you are running ANSYS LS-DYNA, you can use the shared memory parallel processing (SMP) capabilities of LS-DYNA. For more information, see Section 5.3: Shared Memory Parallel Processing in the *ANSYS LS-DYNA User's Guide*.

13.1. Activating Parallel Processing

Regardless of which method you use, you must first specify the number of processors to use.

For the shared memory architecture methods, you have several options.

- Specify multiple processors via the NUM_PROC configuration parameter. Place this line (which *must* begin in column 1) in your **config81.ans** file:

```
NUM_PROC=N
```

- Issue the ANSYS **/CONFIG** command, as follows:

```
/CONFIG,NPROC,N
```

- Issue the **SETNPROC** macro in one of the following ways. **SETNPROC** sets the number of processors to the maximum allowed on the system. (The value may be less than the actual number of processors.)

  ```
  SETNPROC,N
  ```

  sets the number of processors to *N*.

  ```
  SETNPROC,-P
  ```

  sets the number of processors to the maximum allowed on the system *minus* the number specified by *P* (but never less than one processor).

In each case, *N* represents the number of processors to use. If *N* exceeds the number of available processors, ANSYS uses *all* available processors.

For large multiprocessor servers, ANSYS recommends setting *N* to a value no higher than the number of available processors *minus one*. For example, on an eight-processor system, set *N* to 7. However, on multiprocessor workstations, You may want to use all available processors to improve the total solution time.

For the distributed memory architecture method, issue the **DSPROC** command in SOLUTION to specify the total number of processors to be used. You will also need to issue the **DSOPT** command to specify the distributed mode, and optionally, the number of preconditioner domains. Specify the **DSOPT**,Local option to use the distributed solvers on a single, multi-processor machine, which is normally a shared memory machine.

For optimal performance when solving a large model, close down all other applications before launching ANSYS with a multiprocessor solver. ANSYS recommends running distributed solvers when the network is not busy, such as at night and on weekends.

For a complete and up-to-date list of systems on which ANSYS supports parallel processing, point your Web browser to the following URL:

```
http://www.ansys.com/services/hardware_support/parallel/index.htm
```

13.1.1. System-Specific Considerations

For the shared memory architecture methods, the number of processors that the ANSYS program uses is limited to the *lesser* of one of the following:

- The actual number of CPUs available
- The value specified via the **/CONFIG**,NPROC command *or* the NUM_PROC configuration parameter (if used).

  Note — Issuing the **/CONFIG**,NPROC command to change the number of processors *overrides* the number specified via the NUM_PROC configuration parameter.

- On SGI systems, the number of processors is limited to the lesser of the actual quantity of CPUs and the value of the environment variable **MP_SET_NUMTHREADS** (if used).

You can specify multiple settings for the number of CPUs to use during an ANSYS session. However, ANSYS recommends that you issue the **/CLEAR** command before resetting the number of processors for subsequent analyses.

On IBM AIX systems, verify that the **host.list** file is consistent with the number of processors specified. For more information, see the IBM Parallel Environment for AIX documentation, *Operation and Use, Volume 1, Using the Parallel Operating Environment*.

If you are using the distributed memory architecture (and thus the DDS, DPCG, or DJCG solvers), you can specify more processors (via **DSPROC**) than are physically available at the cost of slower total elapsed run time. This usage is not recommended; however, it is useful to test distributed parallel computing over a limited-resource network.

13.2. Using the Parallel Performance for ANSYS Add-On

The Parallel Performance for ANSYS add-on is a powerful tool for solving large (10 million or more DOFs) models over multiple processors. When ANSYS solves parts of the model in parallel, the total solution time is reduced.

> *Note* — Occasionally, memory limitations can prevent ANSYS from solving very large models. For greater control over running deferred jobs and recovering from systems failures during deferred jobs, ANSYS recommends using a batch-management product such as LSF/Batch from Platform Computing.

The Parallel Performance for ANSYS add-on consists of two groups of solvers which can run large models over multiple processors:

- **Algebraic Multigrid iterative equation solver (AMG)** -- Solves static or transient analyses over multiple processors on the same system to speed up processing time.
- **Distributed Solvers: Distributed Domain Solver (DDS)**, **Distributed Preconditioned Conjugate Gradient (DPCG)**, **Distributed Jacobi Conjugate Gradient (DJCG)** -- Solves static or transient analyses over multiple systems or multiple processors. Each solver's specific uses and limitations is described in more detail in Section 13.2.2: About the Distributed Solvers (DDS, DPCG, DJCG).

This section does not document the *configuration* of the Parallel Performance for ANSYS solvers. If you want to run Parallel Performance for ANSYS solvers, see the *ANSYS Installation and Configuration Guide* for your platform.

13.2.1. About the Algebraic Multigrid (AMG) Solver

The Algebraic Multigrid (AMG) solver, which is based on the multi-level method, is an iterative solver that you can use in both single-processor and multiprocessor environments on a single machine with shared memory architecture. To use this solver, you must have a license for the Parallel Performance for ANSYS add-on.

In a multiprocessor environment, the AMG solver provides better performance than the PCG and ICCG solvers on shared memory parallel machines. It also handles indefinite matrix problems for nonlinear analyses. However, the AMG solver typically uses 50 percent more memory than the PCG solver. The AMG solver is also intended for problems in which the PCG and ICCG solvers would have difficulty converging (for example, large, ill-conditioned problems where the ill-conditioning is due to large element aspect ratios within a mesh, or cases in which shell or beam elements are attached to solid elements). In terms of CPU time when used in a single-processor environment, the AMG solver performs better than the PCG and ICCG solvers for ill-conditioned problems, and it delivers about the same level of performance for ordinary problems.

The AMG solver is available only for static analyses and full transient analyses. (These analyses can be linear or nonlinear.) In addition, the efficiency of the AMG solver is limited to single-field structural analyses in which the solution DOFs are combinations of UX, UY, UZ, ROTX, ROTY, and ROTZ. For analyses such as single-field thermal analyses in which the solution DOF is TEMP, the AMG solver is less efficient than the PCG or ICCG.

13.2.1.1. Using the Algebraic Multigrid (AMG) Solver

The AMG solver calculates solutions over multiple processors on the same system. If you have a Parallel Performance for ANSYS license, you can use the AMG solver by following these steps:

1. When starting your ANSYS session, specify the Parallel Performance for ANSYS add-on from the operating system command line:

   ```
   ansys81 -pp
   ```

2. When defining the analysis, select the AMG solver [**EQSLV**,`AMG`].
3. (Optional.) To use the AMG solver in a multiprocessor environment, activate parallel processing. You can do this either by issuing the **/CONFIG**,NPROC,*VALUE* command, or by executing the **SETNPROC** macro. (*VALUE* indicates the number of processors to be used. The AMG solver is expected to scale linearly when used with up to 8 processors. When the AMG solver is used with more than 8 processors, any additional scalability is minimal; the larger the problem, the higher the potential for scalability.)

13.2.2. About the Distributed Solvers (DDS, DPCG, DJCG)

The distributed solvers (DDS, DPCG, DJCG) decompose large models into smaller domains, then send those domains to multiple processors. To use these solvers, you must have a license for the Parallel Performance for ANSYS add-on.

The distributed solvers work by dividing the meshed model into domains without user intervention. During the solution phase, the solution is launched on the host system and on all designated systems simultaneously. Each processor solves its own set of domains, communicating with the other processors. The host system then builds the total DOF solution, and the analysis continues on the host system only.

The domains noted here are domains which are used by a solver's preconditioners only. The larger domains built based on these “preconditioner” domains are called CPU domains. The number of CPU domains is equal to the number of processors specified.

Choose the appropriate distributed solver based on your particular analysis. In general, use the DDS solver for large static or full transient analyses. The DJCG solver is best suited for large 3-D scalar field analyses, such as a 3-D thermal or magnetic field analysis. The DPCG solver is best suited for structural analyses. It is valid for elements with symmetric, sparse, or indefinite matrices.

> *Note* — The probabilistic design system can also distribute solutions across processors, but uses a system independent of the distributed solver. See Chapter 3, “Probabilistic Design” in the *ANSYS Advanced Analysis Techniques Guide* for more information.

DDS Solver The DDS solver is scalable, intended for large static or full transient analyses, with symmetric matrices that do not involve inertia relief or problems using the probabilistic design system (PDS). The solver cannot be used with models containing p-elements, superelements, or with PRETS179 elements, and should not be used with SOLID97, INTER115, or SOLID117. ANSYS does not recommend using the DDS solver with high-aspect-ratio elements; doing so could result in convergence difficulties. The solver can be used in analyses involving coupling and constraint equations.

The DDS solver works with the deformable-to-deformable, surface-to-surface contact elements TARGE169 to CONTA174, and the node-to-node contact elements. It does not work with the rigid-to-deformable or node-to-surface contact elements in general. It also does not support elements with u-P formulation option or Lagrange multipliers.

In general, the DDS solver works well when simple coupling and/or constraint equations are used in the model. However, it may not work with complicated constraint equation cases, which often arise from the use of **CERIG**, **RBE3**, **CEINTF** or internal constraint equations generated by contact applications. In this last case, the program will issue an error message.

The solver treats contact nodes and DOFs in coupling and constraint equations as internal master DOFs, resulting in higher memory requirements.

> **Caution:** The solver may fail or its scalability may deteriorate for models with many (greater than 10% of total nodes) contact pairs.

DPCG Solver The Distributed Preconditioned Conjugate Gradient iterative equation solver is based on the PCG solver. The DPCG solver preserves all of the merits of the PCG solver and can be run on either shared memory or distributed memory machines with superior scalability to the PCG solver. Compared to the DDS solver, the DPCG solver is more robust and uses less memory, but has inferior scalability, especially when more than 16 processors are used. The total sum of memory used by the DPCG (summed total over the network or all processors) is about 30% more than the PCG solver. In addition to the limitations of the PCG solver, the DPCG solver does not support subspace eigensolver (Powerdynamics), the **PRECISION** command, or p-elements.

DJCG Solver The Distributed Jacobi Conjugate Gradient iterative equation solver is based on the JCG solver. Scalability of this solver is superior to the JCG solver with little extra memory required. DJCG solver is available only for static and full transient analyses where the stiffness is symmetric. This solver does not support the fast thermal option (**THOPT**). Similar to the DPCG solver, the DJCG solver decomposes large models into domains, but unlike the DPCG, the DJCG uses a less sophisticated diagonal preconditioner. Because of the simplicity of the diagonal preconditioner, the scalability of the DJCG is superior to the DPCG or the DDS solvers. However, as with the JCG solver, this solver is only effective for well-conditioned problems.

13.2.3. Using the Distributed Solvers (DDS, DJCG, DPCG)

Before you can run a distributed solver, you must first install Message Passing Interface (MPI) software on every system on which you intend to use the distributed solver. Throughout ANSYS documentation, the term “MPI software” is used to refer to the MPI software (either native MPI, MPI/Pro, or MPICH) that is right for your platform. See the information on configuring parallel performance solvers in the *ANSYS Installation and Configuration Guide for UNIX* or the *ANSYS Installation and Configuration Guide for Windows* for more information on acquiring MPI and on platform-specific MPI issues.

If you have a Parallel Performance for ANSYS license, you can use a distributed solver by following these steps:

1. Verify that you have access (valid logins, etc.) to all remote systems on which you will be running. On Windows and some UNIX systems, verify that each machine being used has a working directory with a path identical to that on the host machine. We recommend that you always use identical paths on all machines, regardless of platform.
2. Verify that you are running on a homogenous network. Distributed solving will work only across machines of the same platform. If running on a single system, verify that the system has more than one processor.
3. Verify that you have the correct MPI software installed. All machines must use the same version of MPI, installed in the same path. Both ANSYS 8.1 and MPI software must be installed on each system where you will run.
4. Configure the necessary **hosts81.ans** if using the GUI, and configure or check the appropriate **host.list** or **host.list.dds/.dpcg** files. Use the **Configure Cluster** option of the **ANS_ADMIN** utility to configure the **hosts81.ans** file.

   On UNIX and Linux systems, verify that you have an **.rhosts** file that includes all machines that can be connected on the system.
5. When starting your ANSYS session, specify the Parallel Performance for ANSYS add-on and the correct MPI type in the launcher, or from the operating system command line by issuing the following commands:

   For native MPI or MPI/Pro:

```
ansys81 -pp
```

or for MPICH:

```
ansys81 -pp -mpi mpich
```

6. Choose the solver type and options.

   Interactively, choose **Main Menu>Solution>Sol'n Options**. From there you can choose the type of solver (DDS, DPCG, or DJCG), as well as the number of domains (we recommend that you use “auto”), the distribution method (local, file, or script), and the number of processors. Use the **Host Selection** button (available only if you choose the script option and have already created a **hosts81.ans** file and placed it in your working directory) to choose the hosts and number of processors to run on each machine for this run from those specified in the **hosts81.ans** file.The **Host Selection** button will create the script file and the correct **host.list** file in the appropriate format for your platform if they do not already exist.

   When using the DPCG or DJCG solvers, you must specify the total number of processors on the network via the **DSPROC** command.

   All options are not available on all platforms; you will see only those combinations that are valid on your platform.

   In batch mode, you can specify this same information via the **EQSLV**, **DSOPT**, and **DSPROC** commands. If you use the commands directly, you can also specify the maximum number of iterations and the level of difficulty via **DSOPT**. Be aware, however, that if you start in batch mode or by using an existing input listing, and then use the GUI to modify or complete your parallel setup and run, the GUI settings will override any existing commands from your input listing.

7. Click **OK** or issue a SOLVE command, and run your analysis as you normally would.

13.2.3.1. Required Files

Host.list File

The **host.list** files (**host.list**, **host.list.dds**, or **host.list.dpcg**, depending on the platform and type of solver selected) always contains the list of machines in the distributed environment. In addition to listing the machines to be used, the **host.list** file can also list the number of processors on that machine, the path to the distributed executable (**ansdds.e81**, etc.), number of threads, or other information, depending on the platform. Platform specifics are discussed in a later section, and include examples of the **host.list** files for each platform.

The correct **host.list** file and script files are created automatically via the GUI when you choose **Host Selection**.

Note — This file is sometimes referred to as a host file or a machines file.

.rhosts File

On UNIX and Linux systems, you need to specify the **.rhosts** file, and it must reside directly under the home directory on each machine. You must be able to log in to each system listed in the **.rhosts** file. The **.rhosts** file must include all machines involved in distributed solving. The format for the **.rhosts** file is:

```
machinename1 username
machinename2 username
machinename1.ansys.com username
machinename2.ansys.com username
localhost
```

For example, on an SGI machine, the **.rhosts** file might look like this:

lmserv jqdoe
beast jqdoe
lmserv.ansys.com jqdoe
beast.ansys.com jqdoe
localhost

CONFIG and Script Files

If you are using the file option, you will also need to specify the correct **config81.dds** file. If you are using the script option, you will need a script to call the correct executable. These files are explained in Section 13.2.3.2: Distribution Method Options.

To run a distributed solver process, you must be able to log in to each system listed in the launch script. If you want to see more examples of launch scripts, look in the **/ansys_inc/v81/ansys/bin/** directory.

Hosts81.ans File

For parallel processing you need to specify the remote hosts you want to use. This information is placed in a file called **hosts81.ans**. You can create this file using a text editor or you can use the **ANS_ADMIN** utility (see the online help available with the **ANS_ADMIN** utility for more information). This file contains host information for all remote machines on which you may want to run. This file is global and does not contain job-specific information. If you have multiple users running parallel or distributed jobs at your site, you should have one **hosts81.ans** file for all users.

A sample **hosts81.ans** file looks like this:

```
# This file is used to specify those hosts that the ANSYS Nanny may
# run children on.
#
# Each host entry is to be on its own line.  The host entry consists of
# several fields which are space delimited.
#
#  Field 1 - host IP address or name
#  Field 2 - host machine type
#  Field 3 - execution key (used for Probabilistic Design only):
#        0-Use a remote shell to start the child process;
#          this requires a remote shell server to be
#          running on the host machine.
#    >1024-Use a running ANSYS thin server on the host
#          which is listening on this port number.
#  Field 4 - The default maximum number of jobs to run on this host
#  Field 5 - The time in minutes to check again if the host is available.
#            If this is zero then the host will not be checked again.
#  Field 6 - The local port number to start the communication with the
#            ANSYS Thin Server on. This is tied to authentication on the
#            ANSYS Thin Server.
#  Field 7 - The directory to create the children subdirectories in
#  Field 8 - The cluster type.  Only valid entry is MPI.
#  Field 9 - The speed factor (relative speed to other machines listed).
             Only valid entry is 1.
#  Field 10 - Number of OpenMP threads.  Only valid entry is 1.
# Example:
#
#  UNIX box that has five processors
#  zeus  sgi64    0     5  30    2000     /scratch/wjc
#  Microsoft box using the ANSYS Thin Server
#  wjcpc XP       2010  1  0     2000     C:\TEMP
alpha1   alpha    0      1     15    2000     /scratch/epc   MPI 1 1
athena   sgi64    0      1     15    2000     /scratch/epc   MPI 1 1
rs43p    rs6000   0      1     15    2000     /home/pdstest  MPI 1 1
rs260    rs64     0      1     15    2000     /home/pdstest  MPI 1 1
snoopy   hppa8000 0      1     15    2000     /home/pdstest  MPI 1 1
alpha24  alpha    0      1     15    2000     /home/pdstest  MPI 1 1
hp770    hppa8000 0      1     15    2000     /home/pdstest  MPI 1 1
```

```
    us60     usparc   0      1     15    2000     /home/pdstest  MPI 1 1
    ss60     sun64    0      1     15    2000     /home/pdstest  MPI 1 1
```

On Windows, ANSYS searches for this file first in the local directory, followed by the home directory, and finally the apdl directory. On UNIX, the file needs to reside in your working directory.

Although you can create the **hosts81.ans** manually using a text editor, we recommend that you use the **ANS_ADMIN** utility to automatically create the file in the correct format for your platform.

Use caution when setting field #4 (maximum number of jobs to run on this host). We recommend that you do not set this field to a number greater than the number of processors on the machine. Although distributed solving will still work in such a scenario, you will see a marked degredation in performance.

On UNIX or Linux platforms, you need to run ANSYS from within the working directory if you want the Host Selection option in the GUI to work properly.

13.2.3.2. Distribution Method Options

When you choose the distribution method in the GUI or invoke **DSOPT**, you can enter one of three configuration options:

- `local` -- Sends domains to processors on the local system only. Used for a single machine with multiple processors.
- `file` -- Calls **config81.dds** to set processor information for the distributed solver. Use of the **config81.dds** file is explained in more detail in later sections. See your MPI software documentation for more information about MPI commands and options.

  Note — The `file` option will not run multiple machines across a network on Sun and IBM platforms. Use the `script` option instead.

- `script` -- Calls the distributed solver executable (**ansdds81**, **ansddsmpich81**, etc.) to set processor information for the distributed solvers. Use this option only if you are experienced in writing MPI applications.

To handle more difficult analyses, specify a level of difficulty on the **DSOPT** command (DDS and DPCG solvers only). The higher the level of difficulty, the more difficult the analysis the solver can handle, but the more memory and CPU time the solver will take. Lev_Diff = 1 or 2 is recommended for less difficult problems (such as well-shaped 3-D solid elements). Lev_Diff = 3 or 4 is recommended for more ill-conditioned (difficult) problems (more than 30% of elements are beams or shells). For models using nonuniform materials, use a higher level of difficulty. Analyses with Lev_Diff = 3 or 4 will converge faster, but with high memory consumption and increased CPU time per iteration.

Using the File Option and the config81.dds File

The **file** option specifies that MPI should look in the **config81.dds** file for configuration options to define processor information.

The name of the configuration file differs, depending on solver and type of MPI:

- **config81.dds** for the DDS solver using MPI
- **config81.dpcg** for the DPCG and DJCG solvers using MPI
- **configmpich81.dds** for the DDS solver using MPICH
- **configmpich81.dpcg** for the DPCG and DJCG solvers using MPICH

On UNIX platforms, the format of the **config81.dds** file varies, depending on the platform. Refer to the MPI documentation for the specific format of each platform.

In general, the **config81.dds** file contains the following information on each line; specific examples for each platform are shown in platform specifics.

> *Note* — The `-np` immediately before the number of processors is an **mpirun** command argument indicating "number of local processors." Other **mpirun** command arguments can be used in this position in the file. For information about the **mpirun** command arguments, type:
>
> ```
> man mpirun
> ```
>
> from the UNIX command prompt.

`SystemName`
: The name of the system to send distributed solver processes to.

`NumProcessors`
: The number of processors on the named system to send distributed solver processes to.

`ExecutablePath`
: The full path name to the executable on the local system. Do *not* use relative path names.

  Executable names for DDS are:

  - ansdds.e81 - UNIX DDS
  - ansdds.exe - Windows DDS
  - ansddsmpich.e81 - MPICH Linux DDS
  - ansddsmpich.exe - MPICH Windows DDS

  Executable names for DPCG or DJCG are:

  - ansdpcg.e81 - UNIX DPCG, DJCG
  - ansdpcg.exe - Windows DPCG, DJCG
  - ansdpcgmpich.e81 - MPICH Linux DPCG, DJCG
  - ansdpcgmpich.exe - MPICH Windows DPCG, DJCG

Use the file option to specify exactly how many processes you want sent to certain machines. Machines are listed in the **host.list** file. The first name in the **host.list** file should be the local hostname.

If the **config81.dds** file contains entries for multiple machines, the master or host machine *must* be the first machine referenced in the file.

ANSYS searches your directories in the following order to look for the **config81.dds** file:

1. Your current directory
2. Your home directory

Using the Script Option and the Distributed Solver Launch Scripts

If you are familiar with writing MPI applications, you can create a launch script to further customize the application. When you choose the script distribution option in the GUI or invoke **DSOPT**,`script`, ANSYS searches your directories in the following order to look for these scripts:

1. Your current directory
2. Your home directory
3. The **/ansys_inc/v81/ansys/bin/** directory (On Windows, C:\Program Files\ANSYS Inc\V81\ANSYS\bin\platform)

Caution: If the script does not have execution permission, you will get a fatal error.

For best performance, we recommend that you place the scripts and config files in your working directory.

Note to Alpha Users

If you are running an SC cluster on an Alpha, you must rename the distributed solver executable **ansdds_sc.e81** to **ansdds.e81** to use the **ansdds81** script. You *must* use this script (**DSOPT**,`script`).

Launch Script Names

The launch script name depends on your platform and type of distributed solver:

UNIX	**Solver Type**
ansdds81	DDS
ansddsmpich81	MPICH Linux DDS
ansdpcg81	DPCG, DJCG
ansdpcgmpich81	MPICH Linux DPCG, DJCG

Windows	**Solver Type**
ansdds.bat	DDS
ansddsmpich.bat	MPICH Windows DDS
ansdpcg.bat	DPCG, DJCG
ansdpcgmpich.bat	MPICH Windows DPCG, DJCG

Examples for each platform are shown under platform specifics. Remember that with the DPCG and DJCG solvers, you must also specify the total number of (network) processors.

13.2.3.3. Platform Specific Information

The following tables show examples of the various files needed for each platform. The first column shows the appropriate file name. The second shows the use. Uses can be:

- S - script option
- F - file option
- MP - native MPI or MPIPro
- MC - MPICH (Windows or Linux only)
- A - required for all uses

The third column shows an example of the file format.

Table 13.1 HP Itanium2 64-bit

File	Use	Format and Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`rx5670 hpia64 0 8 10 2000 /disk1/people/jqdoe/dstest MPI 1 1`
		`hpux64 hpia64 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`

host.list	A	Format:
		`MPIFlag MachineName NumProcessors ExecutablePath NumThreads`
		Example:
		`-h rx5670 -np n /ansys_inc/v81/ansys/bin/hpia64/ansdds.e81 -n 1`
		`-h hpux64 -np n /ansys_inc/v81/ansys/bin/hpia64/ansdds.e81 -n 1`

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -f host.list`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your **ANSYS81_DIR** path is different.

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -f host.list`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your **ANSYS81_DIR** path is different.

config81.dds	MP, F	Format:
		`NumProcessors ExecutablePath`
		`MPIFlag MachineName NumProcessors ExecutablePath`
		Example:
		`-np n /ansys_inc/v81/ansys/bin/hpia64/ansdds.e81`
		`-h hpux64 -np n /ansys_inc/v81/ansys/bin/hpia64/ansdds.e81`

File	Use	Format and Example

config81.dpcg	MP, F	Format:
		`NumProcessors ExecutablePath`
		`MPIFlag MachineName NumProcessors ExecutablePath`
		Example:
		`-np n /ansys_inc/v81/ansys/bin/hpia64/ansdpcg.e81`
		`-h hpux64 -np n /ansys_inc/v81/ansys/bin/hpia64/ansdpcg.e81`

.rhosts	A	See .rhosts File.

Table 13.2 HP PA 8000 64-bit

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`snoopy hppa8000-64 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`
		`hpuxdemo.ansys.com hppa8000-64 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`

host.list.dds	A	Format:
		`MPIFlag MachineName NumProcessors ExecutablePath NumThreads`
		Example:
		`-h snoopy -np n /ansys_inc/v81/ansys/bin/hppa8000-64/ansdds.e81 -n 1`
		`-h hpuxdemo -np n /ansys_inc/v81/ansys/bin/hppa8000-64/ansdds.e81 -n 1`

host.list.dpcg	A	Format:
		`MPIFlag MachineName NumProcessors ExecutablePath NumThreads`
		Example:
		`-h snoopy -np n /ansys_inc/v81/ansys/bin/hppa8000-64/ansdpcg.e81 -n 1`
		`-h hpuxdemo -np n /ansys_inc/v81/ansys/bin/hppa8000-64/ansdpcg.e81 -n 1`

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -f host.list`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your **ANSYS81_DIR** path is different.

File	Use	Example

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -f host.list`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your **ANSYS81_DIR** path is different.

config81.dds	MP, F	Format:
		`NumProcessors ExecutablePath`
		
		`MPIFlag MachineName NumProcessors ExecutablePath`
		Example:
		`-np `*`n`*` /ansys_inc/v81/ansys/bin/hppa8000-64/ansdds.e81`
		`-h hpuxdemo -np `*`n`*` /ansys_inc/v81/ansys/bin/hppa8000-64/ansdds.e81`

config81.dpcg	MP, F	Format:
		`NumProcessors ExecutablePath`
		
		`MPIFlag MachineName NumProcessors ExecutablePath`
		Example:
		`-np `*`n`*` /ansys_inc/v81/ansys/bin/hppa8000-64/ansdpcg.e81`
		`-h hpuxdemo -np `*`n`*` /ansys_inc/v81/ansys/bin/hppa8000-64/ansdpcg.e81`

.rhosts	A	See .rhosts File.

Table 13.3 SGI 64-bit

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`lmserv sgi64 0 8 10 2000 /home/staff/jqdoe/dstest MPI 1 1`
		
		`beast sgi64 0 8 10 2000 /home/staff/jqdoe/dstest MPI 1 1`

host.list	A	Not required

File	Use	Example

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun lmserv -np n /ansys_inc/v81/ansys/bin/sgi64/ansdds.e81 -n 1`
		`: beast -np n /ansys_inc/v81/ansys/bin/sgi64/ansdds.e81 -n 1`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, except for the ANSYS path and machine names, and number of processors.

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun lmserv -np n /ansys_inc/v81/ansys/bin/sgi64/ansdpcg.e81 -n 1`
		`: beast -np n /ansys_inc/v81/ansys/bin/sgi64/ansdpcg.e81 -n 1`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, except for the ANSYS path and machine names.

config81.dds	MP, F	Format: *MachineName NumProcessors MachineName NumProcessors (...) ExecutablePath* Example: `lmserv -np n,beast -np n /ansys_inc/v81/ansys/bin/sgi64/ansdds.e81`

config81.dpcg	MP, F	Format:
		MachineName NumProcessors MachineName NumProcessors (...) ExecutablePath
		Example:
		`lmserv -np n,beast -np n /ansys_inc/v81/ansys/bin/sgi64`
		`/ansdpcg.e81`

.rhosts	A	See .rhosts File.

Table 13.4 Sun ULTRA III and UltraSPARC 64-bit

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`sunds3 usIII 0 8 10 2000 /ansys/users/jqdoe/dstest MPI 1 1`
		`sunds5 usIII 0 8 10 2000 /ansys/users/jqdoe/dstest MPI 1 1`

File	Use	Example
host.list	A	Format: *MachineName* *MachineName* *...* Example: sunds3 sunds3 sunds5 sunds5

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mprun -np 5 -m host.list /ansys_inc/v81/ansys/bin/usIII`
		`     /ansdds.e81 -n 1`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mprun -np n -m host.list /ansys_inc/v81/ansys/bin/usIII`
		`   /ansdpcg.e81 -n 1`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

config81.dds/.dpcg	MP, F	File option not supported by Sun
.rhosts	A	See .rhosts File.

Table 13.5 IBM 64-bit

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`hammer aix64 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`
		`ibm630 aix64 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`

host.list	A	Format: *`MachineName`* *`MachineName`* `...` Example: hammer ibm630 (One proc on each machine)

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`poe /ansys_inc/v81/ansys/bin/aix64/ansdds.e81 -procs 2`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`poe /ansys_inc/v81/ansys/bin/aix64/ansdpcg.e81 -procs 2`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

config81.dds/con-fig81.dpcg	MP, F	File option not supported on IBM
.rhosts	A	See .rhosts File.
MP_HOSTFILE	A	Environment variable - set to **host.list** location. Use absolute addressing, not relative.
NLSPATH	A	Environment variable - set to **/usr/lib/nls/msg/en_US/pepoe.cat**

Table 13.6 HP Alphaserver

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`deva1.ansys.com alpha 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`
		`alpha3.ansys.com alpha 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`

host.list.dds	A	Format:
		`MPIFlag MachineName ExecutablePath`
		Example:
		`1 deva1.ansys.com /ansys_inc/v81/ansys/bin/alpha/ansdds.e81`
		`1 alpha3.ansys.com /ansys_inc/v81/ansys/bin/alpha/ansdds.e81`

host.list.dpcg	A	Format:
		`MPIFlag MachineName ExecutablePath`
		Example:
		`1 deva1.ansys.com /ansys_inc/v81/ansys/bin/alpha/ansdpcg.e81`
		`1 alpha3.ansys.com /ansys_inc/v81/ansys/bin/alpha/ansdpcg.e81`

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`dmpirun -pf host.list`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your **ANSYS81_DIR** path is different.

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`dmpirun -pf host.list`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your **ANSYS81_DIR** path is different.

config81.dds	MP, F	Format:
		`MPIFlag MachineName ExecutablePath`
		Example:
		`1 deva1 /ansys_inc/v81/ansys/bin/alpha/ansdds.e81`
		`1 alpha3 /ansys_inc/v81/ansys/bin/alpha/ansdds.e81`

File	Use	Example

config81.dpcg	MP, F	Format:
		`MPIFlag MachineName ExecutablePath`
		Example:
		`1 deva1 /ansys_inc/v81/ansys/bin/alpha/ansdpcg.e81`
		`1 alpha3 /ansys_inc/v81/ansys/bin/alpha/ansdpcg.e81`

.rhosts	A	See .rhosts File.

Table 13.7 Linux 32-bit

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`linux32.ansys.com linuxia32 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`
		`linux3.ansys.com linuxia32 0 8 10 2000 /home/jqdoe/dstest MPI 1 1`

host.list	A	Format: *`MachineName`* *`MachineName`* `...` Example: linux32.ansys.com linux3.ansys.com

ansdds81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -np` *`n`* `-mach_file host.list`
		`"/ansys_inc/v81/ansys/bin/linuxia32/ansdds.e81"`
		`exit`
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

File	Use	Example

ansdpcg81	MP, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -np n -mach_file host.list`
		`"/ansys_inc/v81/ansys/bin/linuxia32/ansdpcg.e81"`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

ansddsmpich81	MC, S	`#!/bin/sh`
		`ANSYS81_DIR=/home/jqdoe/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun  -np n -machinefile host.list "/home/jqdoe`
		`     /ansys_inc/v81/ansys/bin/linuxia32/ansddsmpich.e81"`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

ansdpcgmpich81	MC, S	`#!/bin/sh`
		`ANSYS81_DIR=/home/jqdoe/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun  -np n -machinefile host.list "/home/jqdoe`
		`     /ansys_inc/v81/ansys/bin/linuxia32/ansddsmpich.e81"`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

config81.dds	MP, F	Format:
		`SystemName NumProcessors ExecutablePath`
		Example:
		`linux32 1 /ansys_inc/v81/ansys/bin/linuxia32/ansdds.e81`
		`linux3 1 /ansys_inc/v81/ansys/bin/linuxia32/ansdds.e81`

config81.dpcg	MP, F	Format:
		`SystemName NumProcessors ExecutablePath`
		Example:
		`linux32 n /ansys_inc/v81/ansys/bin/linuxia32/ansdpcg.e81`
		`linux3 n /ansys_inc/v81/ansys/bin/linuxia32/ansdpcg.e81`

File	Use	Example

configmpich81.dds	MC, F	Format:
		`MachineName Master/Slave ExecutablePath UserName`
		Example:
		`linux32 0`
		`/home/jqdoe/ansys_inc/v81/ansys/bin/linuxia32/ansddsmpich.e81`
		`     jqdoe`
		`linux3 1`
		`/home/jqdoe/ansys_inc/v81/ansys/bin/linuxia32/ansddsmpich.e81`
		`     jqdoe`

configmpich81.dpcg	MC, F	Format:
		`MachineName Master/Slave ExecutablePath UserName`
		Example:
		`linux32 0`
		`/home/jqdoe/ansys_inc/v81/ansys/bin/linuxia32/ansdpcgmpich.e81`
		`     jqdoe`
		`linux3 1`
		`/home/jqdoe/ansys_inc/v81/ansys/bin/linuxia32/ansdpcgmpich.e81`
		`     jqdoe`

.rhosts	A	See .rhosts File.

Table 13.8 Linux 64-bit

File	Use	Example

hosts81.ans	A	Format:
		See Hosts81.ans File.
		Example:
		`cloud9 linuxia64 0 8 10 2000 /disk1/home/jqdoe/dstest MPI 1 1`
		`linux2 linuxia64 0 8 10 2000 /disk1/home/jqdoe/dstest MPI 1 1`

host.list	A	Format: *`MachineName`* *`MachineName`* `...` Example: cloud9 linux2

File	Use	Example

ansddsmpich81	MC, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdds`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -np` *`n`* `-machinefile host.list`
		`/ansys_inc/v81/ansys/bin/linuxia64/ansddsmpich.e81`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

ansdpcgmpich81	MC, S	`#!/bin/sh`
		`ANSYS81_DIR=/ansys_inc/v81/ansys`
		`export ANSYS81_DIR`
		`ANSYS_DIR=$ANSYS81_DIR`
		`WHICHANSCRIPT=ansdpcg`
		`export WHICHANSCRIPT`
		`. $ANSYS_DIR/bin/anssh.ini`
		`stat=0`
		`mpirun -np` *`n`* `-machinefile host.list`
		`/ansys_inc/v81/ansys/bin/linuxia64/ansdpcgmpich.e81`
		`exit`
		
		Note — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.

configmpich81.dds	MC, F	Format:
		`MachineName Master/Slave ExecutablePath UserName`
		Example:
		`cloud9 0 /ansys_inc/v81/ansys/bin/linuxia64/ansddsmpich.e81`
		`jqdoe`
		`linux2 1 /ansys_inc/v81/ansys/bin/linuxia64/ansddsmpich.e81`
		`jqdoe`

configmpich81.dpcg	MC, F	Format:
		`MachineName Master/Slave ExecutablePath UserName`
		Example:
		`cloud9 0 /ansys_inc/v81/ansys/bin/linuxia64/ansdpcgmpich.e81`
		`jqdoe`
		`linux2 1 /ansys_inc/v81/ansys/bin/linuxia64/ansdpcgmpich.e81`
		`jqdoe`

.rhosts	A	See .rhosts File.

Table 13.9 Windows XP/2000/IA-64

File	Use	Example
hosts81.ans	A	Format: See Hosts81.ans File. Example: `w2testlab1 intel 0 2 10 2000 C:\dstest MPI 1 1` `w2testlab2 intel 0 2 10 2000 C:\dstest MPI 1 1`
host.list	A	Format: *`MachineName`* *`MachineName`* `...` Example: w2testlab1 w2testlab2
ansdds.bat	MP, S	`mpirun  -np n -mach_file host.list` `"C:\Program Files\ANSYS Inc\V81\ANSYS\bin\intel\ansdds.exe"` `exit` *Note* — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.
ansdpcg.bat	MP, S	`mpirun  -np 2 -mach_file host.list` `"C:\Program Files\ANSYS Inc\V81\ANSYS\bin\intel\ansdpcg.exe"` `exit` *Note* — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.
ansddsmpich.bat	MC, S	`mpirun -np n -machinefile host.list` `"C:\Program Files\ANSYS Inc\V81\ANSYS\bin\intel\ansddsmpich.exe"` `exit` *Note* — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.
ansdpcgmpich.bat	MC, S	`mpirun -np n -machinefile host.list` `"C:\Program Files\ANSYS Inc\V81\ANSYS\bin\intel\ansdpcgmpich.exe"` `exit` *Note* — If you create this file manually, it should look exactly as it appears here, unless your ANSYS path is different.
config81.dds	MP, F	Format: *`SystemName NumProcessors ExecutablePath`* Example: `w2testlab3 n "C:\Program Files\ANSYS Inc\V81\ANSYS\bin\` `intel\ansdds.exe"` `w2testlab2 n "C:\Program Files\ANSYS Inc\V81\ANSYS\bin\` `intel\ansdds.exe"`

File	Use	Example

config81.dpcg	MP, F	Format:
		`SystemName NumProcessors ExecutablePath`
		Example:
		`w2testlab3 n "C:\Program Files\ANSYS Inc\V81\ANSYS\bin\`
		`intel\ansdpcg.exe"`
		`w2testlab2 n "C:\Program Files\ANSYS Inc\V81\ANSYS\bin\`
		`intel\ansdpcg.exe"`

configmpich81.dds	MC, F	Format:
		`ExecutionCommand ExecutablePath hosts SystemName (...)`
		Example:
		`exe C:\Program Files\ANSYS Inc\V81\ANSYS\bin\intel\`
		`ansddsmpich.exe`
		`hosts`
		`w2testlab3 1`
		`w2testlab2 1`
		Note — Your file format must match this format line-by-line.

configmpich81.dpcg	MC, F	Format:
		`ExecutionCommand ExecutablePath hosts SystemName (...)`
		Example:
		`exe C:\Program Files\ANSYS Inc\V81\ANSYS\bin\intel\`
		`ansdpcgmpich.exe`
		`hosts`
		`w2testlab3 1`
		`w2testlab2 1`
		Note — Your file format must match this format line-by-line.

Usage Notes: You must add the following to your PATH environment variable:

```
C:\Program Files\MPICH\mpd\bin
```

You will also be required to enter your account information in the ANSYS output window for the first distributed solve after you login.

Refer to the MPICH documentation at http://www-unix.mcs.anl.gov/mpi/mpich/ for recommendations on the machine file format for a cluster.

To avoid confusion and minimize difficulties, install ANSYS on the same drive letter on all machines to be used in the parallel run.

13.2.4. Files Written When Using Distributed Solvers

The distributed solvers write out a number of files during the solution phase. Here is a list of those files and a brief description of their contents:

Jobname.DCS	Contains the status of the DDS run
Jobname0.PCS	Contains the status of the DPCG run.
Jobname.dds	Contains the nodal coordinates, element connectivity and displacement boundary conditions

Jobname.erot	Contains element matrices and load vectors
scratch.dds	Contains the name of the Jobname, **Jobname.dds**, and **Jobname.erot** files

Additional files will also be written, but will be removed at the end of the SOLVE.

13.3. Troubleshooting Parallel Performance

This section describes problems which you may encounter while using the Parallel Performance for ANSYS add-on, as well as methods for overcoming these problems. Some of these problems are specific to a particular system, as noted.

Restarting stopped jobs

You must restart a job from the beginning; it is impossible to restart a job from the point where it ended.

Recovering from a computer or network crash

Be sure to kill any lingering processes (UNIX: type **kill -9** from command level, Windows: invoke the Task Manager) on all processors and start the job again.

No permission to system

This error can occur if you do not have login access to a remote system where the distributed solver is supposed to run. If you use UNIX, you can experience this problem if you have not set permissions on the **.rhosts** file properly. Before starting a distributed solver run, be sure you have access to all the remote systems you have specified and that the control node is included in its own **.rhosts** file. If you run on UNIX, be sure you have run the following command:

```
chmod 600 .rhosts
```

MPI configuration file config81.dds not found.

Be certain that the **config81.dds** file exists and is in the correct directory.

Distributed solver failure, no U file detected

Several conditions can cause this error:

If you get this error with a Windows error of **The instruction at “nnnnnnnnnn” referenced memory at “mmmmmmmmmm”. The memory could not be “written”. Click on OK to terminate the program**, verify that all machines are running the same version of MPI/Pro. Also, you might have provided MPI/Pro with an invalid network password. (Windows only)

Check that the directory structures on subsequent machines are identical to the host machine's structure. (This can occur while running the SCRIPT option.)

Verify that you have the same login on all machines.

Verify that the username exists in the same domain on all machines. (Windows only)

Verify that the pathname to the distributed solver executable in the config81.dds file is correct.

Verify that you have gone through the MPI/Pro Password Registration process on all machines. (Windows, Linux only)

The distributed solver did not complete successfully. Check your solver and MPI settings.

For Windows systems: The **machines** file must be present in either the ANSYS working directory or in the directory set by the **MPI_HOME** environment variable. Be certain this file does not have an extension. This file contains a list of the host machines, and the first machine on the list must be the local host.

The distributed solver will not run on a Windows system

Windows requires that each machine being used must have a working directory with a path identical to that on the host machine.

Slave Machine Name:_desktop failed.

User name unable to CreatProcess for \\pathname\<*`solver executable`*>

The directory name is invalid.

Local Machine Name: Process exited with code 1

Local Machine Name: Process exited with code 1

Check that the directory structures on subsequent machines are identical to those on the host machine. (Windows only)

Machine Name: Access violation

This can be caused when more than one host tries to access the same machine while using an older version of MPI/Pro. The version of MPI/Pro provided with ANSYS Release 8.1 resolves this problem. (Windows, Linux only)

The distributed solver execution script not found.

Looking for: .\<*`solver`*>.bat or M:\<*`solver`*>.bat

Verify that the **<*`solver`*>.bat** file exists and is in the correct directory. (Windows only)

The results are nonsensical.

Verify that the **KMP_LIBRARY** environment variable is not set to SERIAL. The ANSYS installation unsets this environment variable; however, other software installed or modified after ANSYS was installed could reset this environment variable. (Windows only)

Solver aborts

If you try to access more nodes than available, the solver will hang. (IBM only)

ansdds process aborts without warning

This can happen on a single system when at least two processors have been specified. There may be insufficient memory allocated, if virtual swap is in use. Display the vswap setting for the system (from command level, type: /sbin/swap -1) and, if necessary, reallocate memory. (SGI only)

*** MPI has run out of PER_PROC message headers.

In some cases, the default settings for environment variables **MPI_MSGS_PER_PROC** and **MPI_REQUEST_MAX** may be too low and may need to be increased. See the MPI documentation for SGI for more information on settings for these and other environment variables. (SGI only)

Other systems may have additional environment variables that need to be adjusted for running very large models as well. If you encounter problems with system-dependent variables, please see your systems support staff or contact your MPI vendor.

no_mp_jobs: Not enough resources available

This can happen when you request more processors than are available. This happens via **ansdds81** if **-np** is greater than the number of available processors. Use

```
mpinfo -N
```

to check the number of available processors. (Sun only)

Distributed solver failure

If you are running on a SUN cluster, you may require a shared file system to enable file availability to ANSYS from nodes within the cluster when running the distributed solver. (Sun only)

Index