

Tabular B.C. & the Function Editor

**Tabular & Function Boundary Conditions
Function Editor Usage**

Overview

- **Tables and functions provide various useful functionality within ANSYS**
 - **Tables** are similar to general arrays but are used to interpolate between values, whereas arrays are discrete values only
 - **Functions** are an extension of tables, used to provide users with means of expressing an equation directly
- **Tables and functions are used in many areas of ANSYS, most notably in definition of loads and boundary conditions**
 - **Loads and b.c. can vary with time, temperature, and/or space**
 - Most analyses support tabular b.c., including Flotran
 - *ANSYS/LS-DYNA uses concept of curves (EDCURVE) instead*
 - **Tables can also define element real constant properties**
 - For example, CONTA171-174 can have thermal contact conductance pressure- or temperature-dependent
 - SURF151/152, FLUID116, and SHELL181 also support tables in some real constant/shell section definitions.

Uses of Tables and Functions

Boundary Condition	Primary Variable	Command
Thermal Analyses		
Fixed Temperature	TIME, X, Y, Z	D,,TEMP
Heat Flow	TIME, X, Y, Z, TEMP	F,,HEAT
Film Coefficient (Convection)	TIME, X, Y, Z, TEMP, VELOCITY	SF,,CONV
Bulk Temperature (Convection)	TIME, X, Y, Z	SF,,,TBULK
Heat Flux	TIME, X, Y, Z, TEMP	SF,,HFLU
Heat Generation	TIME, X, Y, Z, TEMP	BFE,,HGEN
Uniform Heat Generation	TIME	BFUNIF
Structural Analyses		
Displacements	TIME, X, Y, Z, TEMP	D,(UX,UY,UZ,ROTX,ROTY,ROTZ)
Forces and moments	TIME, X, Y, Z, TEMP	F,(FX,FY,FZ,MX,MY,MZ)
Pressures	TIME, X, Y, Z, TEMP	SF,,PRES
Temperature	TIME, X, Y, Z	BF,,TEMP
Electrical Analyses		
Voltage	TIME, X, Y, Z	D,,VOLT
Current	TIME, X, Y, Z	F,,AMPS
Fluid Analyses		
Pressure	TIME, X, Y, Z	D,,PRES
Flow	TIME, X, Y, Z	F,,FLOW
FLOTRAN Analyses		
Nodal DOF	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	D,,(VX,VY,VZ,PRES,TEMP,ENK E, ENDS,SP01-SP06)
Nodal DOF for ALE formulation	TIME, X, Y, Z, TEMP, VELOCITY, PRES, Xr, Yr, Zr	D,,(UX,UY,UZ)
Heat Flux	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	SF,,HFLU
Film Coefficient	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	SF,,CONV
Element Heat Generation	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	BFE,,HGEN
Nodal Heat Generation	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	BF,,HGEN
Nodal Body Force	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	BF,,FORCE
Radiation	TIME, X, Y, Z, TEMP, VELOCITY, PRESSURE	SF,,RAD

Besides the boundary conditions listed on the left, there are some element real constants and section properties that can refer to tables:

CONTA171-174: TCC (thermal conductance) and RDVF (radiation view factor)

SURF151-152: OMEG (rotational speed)

FLUID116: real const 7-10 (angular velocity and slip factor at nodes I and J)

SHELL181: shell thickness via SECFUN

Revision History

- **Tabular boundary conditions, first introduced at 5.5, are continually being enhanced to provide greater functionality at each revision.**
 - **For example, at 6.0, tabular structural loads can vary spatially or with respect to temperature, whereas, previously, it could only be time-dependent.**
- **Function boundary conditions, introduced at 5.7, *supplement* tabular b.c. by allowing the user to define a function rather than a table to relate variable dependencies**
 - **At 6.0, the Function Editor has been enhanced to allow for plotting and listing of function values.**

Tabular Loads and B.C.

Example Usage

- A simple example of defining and using a tabular boundary condition may help illustrate some of the salient features
 - In the next few slides, both command and GUI method will be shown
 - This example is of a *simple* time-dependent loading, and it does not show *all* of the capabilities/features of tabular b.c.
 - There are essentially three steps in using tabular loads/b.c.:
 - Create a table parameter
 - Fill in values of the table parameter
 - Apply table parameter as a load to the model

Steps Required in Using Tabular B.C.

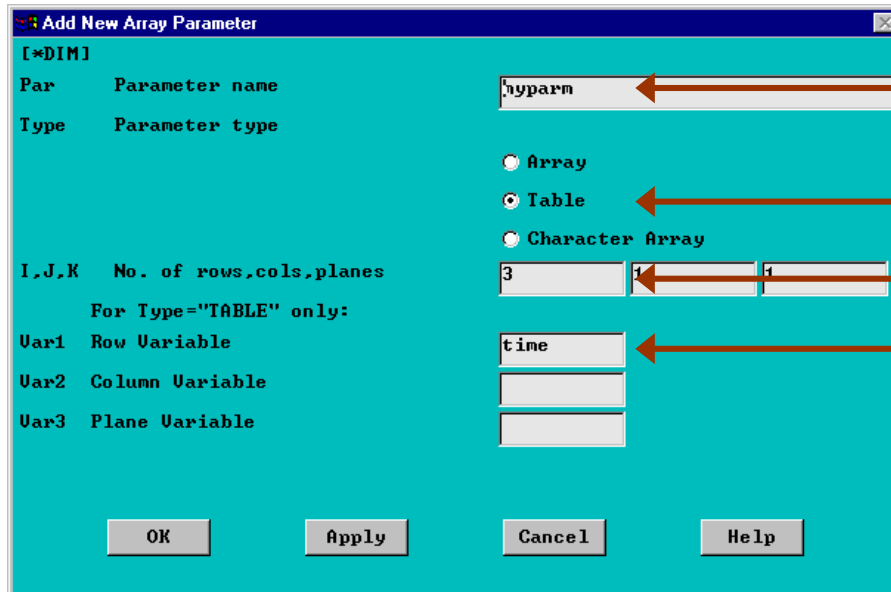
1. Create a table parameter

– Command Method:

`*DIM,myparm,TABLE,3,1,1,TIME`

– GUI Method:

Utility Menu > Parameters > Array Parameters > Define/Edit ...
Click on [Add] Button



Enter name of parameter

Select "Table" option

Enter table length

Enter name of primary variable, such as "time", "temp", "x", "y", and/or "z" (see **DIM* command online help for list of all variable names allowed)

Steps Required in Using Tabular B.C.

Notes:

At 6.0, parameter names can be up to 32-characters long

- Alphanumeric with an underscore
- Cannot contain a space or begin with a number

Tables can have up to 3 *primary variables*
(3-dimensional table)

- Specify the primary variable as the ‘row variable’ name
(Primary variable names are predefined, such as “time”, “temp”, “x”, “y”, or “z”)
- Tables (not functions) can also have independent variables
(see online help references at end for further information)
- Tables can have up to $2^{31} - 1$ rows, columns, and planes
(This specifies the limit to the table size)

Steps Required in Using Tabular B.C.

2. Enter values for the table parameter

– Command Method:

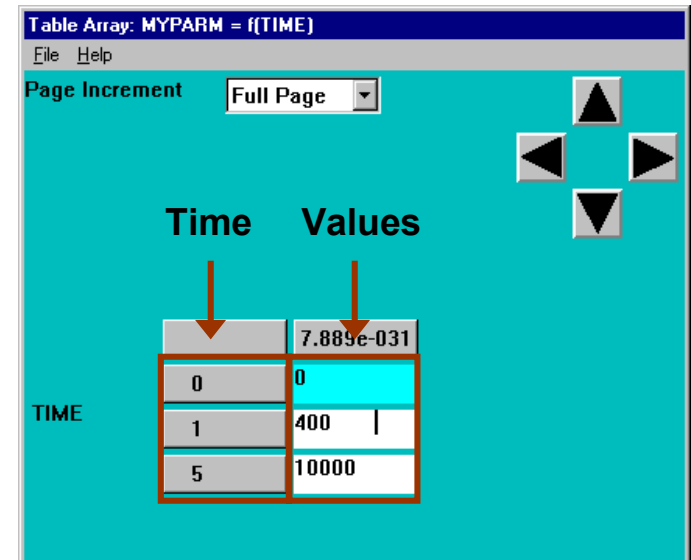
```
MYPARM(1,0,1) = 0  
MYPARM(2,0,1) = 1  
MYPARM(3,0,1) = 5  
MYPARM(1,1,1) = 0  
MYPARM(2,1,1) = 400  
MYPARM(3,1,1) = 10000
```

– GUI Method:

*In dialog box, click on [Edit] Button
Fill in values as shown on right*

The table is defined in such a way that the ‘index’ column (0 column) represents the primary variable, in this case, “time”. The actual ‘vector’ values (1 column) represent the boundary condition values of interest.

In this way, ANSYS can the value at a “time” of “3.2” by interpolating between “400” and “1e4”.



Steps Required in Using Tabular B.C.

3. Apply a load, such as pressure, on the model

– Command Method:

`SFL,all,PRES,%MYPARM%`

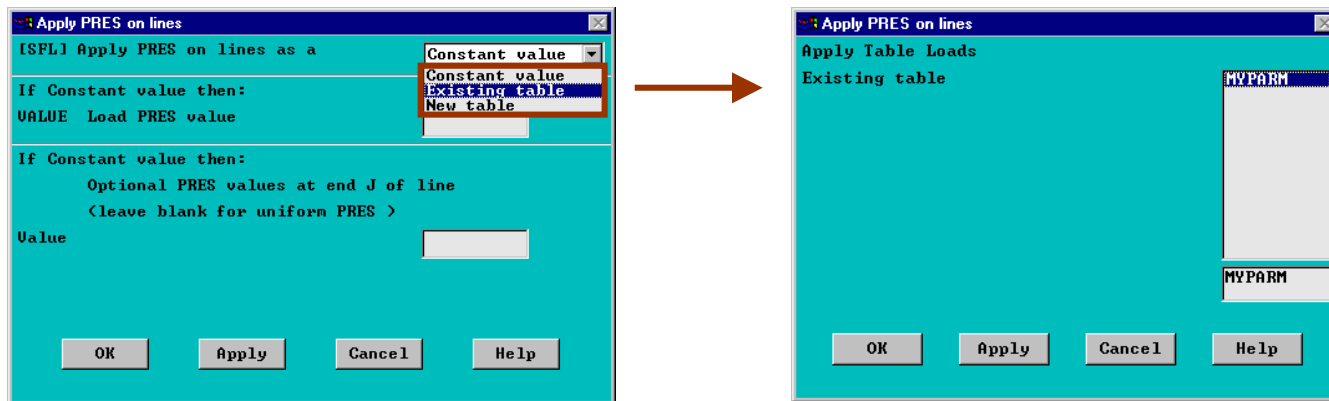
– GUI Method:

Main Menu > Solution > -Loads- Apply > -Structural- Pressure
> On Lines +

Select a line, then click on [OK].

In the resulting dialog box (right), select [Existing Table],
then click on [OK].

Select "MYPARM" table, then click on [OK]



Steps Required in Using Tabular B.C.

Notes:

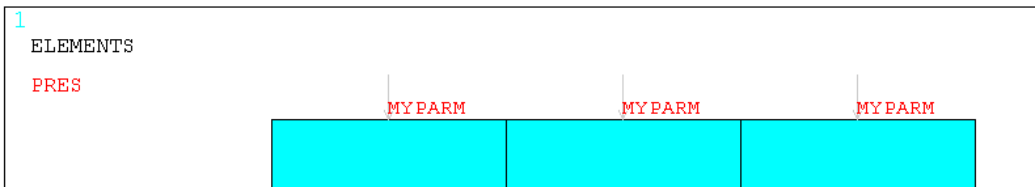
The table parameter, when applied via commands, needs to be enclosed with percent signs “%”. This distinguishes the table parameter from a scalar one.

From the previous slide, it is evident that, when using the GUI, the table parameter does not have to be defined beforehand. The user can, when applying a boundary condition, create the table on-the-fly.

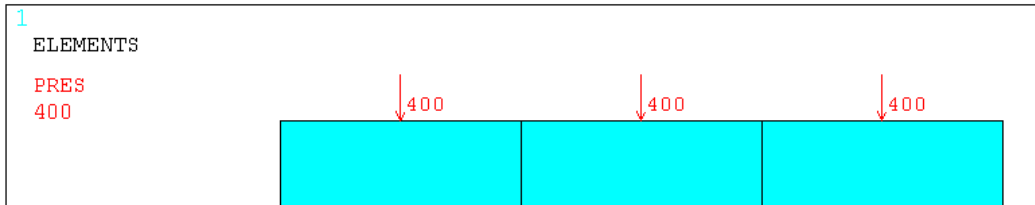
- In this example, we created the parameter beforehand, just to be consistent between GUI and command methods.

Verifying Tabular B.C.

- Applied tabular B.C. can be listed or plotted on the screen
 - If prior to solution, only table name will be shown
 - If in postprocessing, table values can also be plotted for *current load step*
 - **Command Method:**
`/PNUM,TABN,1`
`/PNUM,SVAL,1`
`/PSF,PRES,,2,,1`
 - **GUI Method:**
Utility Menu > PlotCtrls > Numbering ...



Example with table names shown for pressure load



Example with numeric contour values shown for pressure load

Why Use Tabular B.C.?

- **Spatially varying loads**
 - While users can still use SFGRAD or SFFUN to define varying surface loads, tabular loads allow a more general, consistent, and easier way of defining spatial variance.
 - SFGRAD/SFFUN are limited to surface loads. General APDL routines would be needed for other types of boundary conditions.
- **Temperature-dependent loads**
 - Users can still use MP,HF to define temperature-dependent film coefficients. However, tabular loads/b.c. provide more general capabilities, even for structural applications (at 6.0).
 - Defining HF material property, then referencing that convection material parameter with a “-material_number” is unintuitive.
 - Tabular loads provide a more general way of allowing for any temperature-dependent loading – not just convection – including heat flow, heat flux, heat generation, and structural loads.

Why Use Tabular B.C.?

- **Time-dependent loads**
 - Although users can use APDL (e.g., batch input) or LSWRITE & LSSOLVE to define time-dependent loads, tabular loads are preferable for several reasons:
 - Users comfortable with the GUI do not necessarily want to use lots of APDL to define time-varying loads (load steps). Tabular loads are more convenient.
 - LSWRITE has been used in the past, but the jobname.lsxx files only contain FE data, a problem when users need to remesh. Solid model loads are not stored with this load step method, whereas tabular loads can be applied to both FE and solid model. Also, the load step files are separate text files which need to be kept, but tabular loads are part of the database.

Other Useful Information on Tabular Loads

- ***Independent variables*** can be defined by embedding a table within a table.
 - See Ch. 2.6.14.2 “Defining Independent Variables” in the *Basic Analysis Guide* for more details
- There are other commands which help input or manipulate table values
 - ****TREAD*** allows filling up a table from an external source, such as a tab-delimited text file generated from Excel
 - This makes input much easier, especially if the data is already in Excel or a third-party program
 - Ch. 3.11.5.5 “Filling a TABLE Array From a Data File Using *TREAD” in the *APDL Programmer’s Guide*
 - ****TOPER*** allows manipulation of tables
 - Currently (at 6.0), tables can be added or multiplied only, unlike *VOPER or *MOPER, which have more functionality.
 - See *TOPER command help in *ANSYS Command Reference*

Function Editor

Overview of Function B.C.

- ***Functions*** are extensions of tables because, instead of defining data points, an equation is used instead.
 - If data is obtained from an existing Excel spreadsheet or 3rd party program, *tabular loads* will usually be more suitable. Use ***TREAD** to read in text data directly into tables.
 - If loading is known beforehand as a given equation/function, then *function loads* will be the preferred method. The *actual equation* can be input, instead of a piecewise linear curve.
- Internally, the function is defined as a *special type of table*, although the format is hidden from the user.
 - If using either the batch or GUI method, it is recommended to use the *Function Editor* to define functions.
 - Users preferring batch input methods can copy/paste the log file contents to use a function in their input files.

Background on the Function Editor

- When using the Function Editor to define a function, there are essentially three steps involved:
 - Define and save the function using the Function Editor
 - Load the function into a table parameter
 - Apply a load to the model, referencing the newly created table
- Using a simple example, the next few slides will discuss use of the function editor.
 - See Ch. 2.6.15.1 “Using the Function Editor” in the *Basic Analysis Guide* for more details.

Steps for Using the Function Editor

1. Open the Function Editor

– GUI Method:

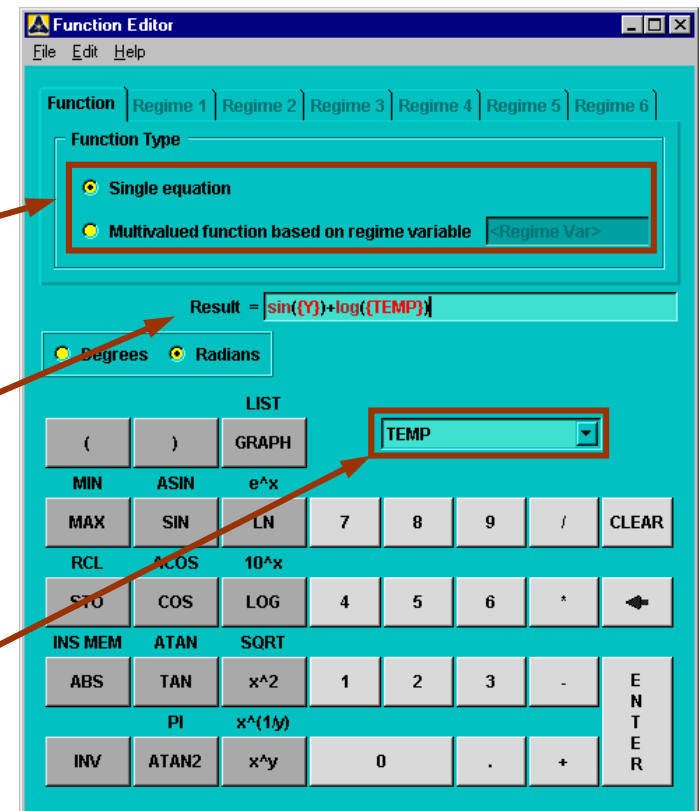
Utility Menu > Parameters > Functions > Define/Edit ...

The Function Editor is an easy-to-use calculator interface. The user can define any type of function, using the keypad.

Either a single equation or 'regime' can be defined. Up to 6 regimes allow for different 'sets' of equations to be defined, although the user must ensure that the regimes are continuous.

Predefined mathematical & trigonometric functions are typically shown in **dark red**, where the arguments are in parentheses.

Primary variables are typically shown in **light red** and are enclosed in curly brackets. Primary variables can be selected from the pop-up menu shown on right.



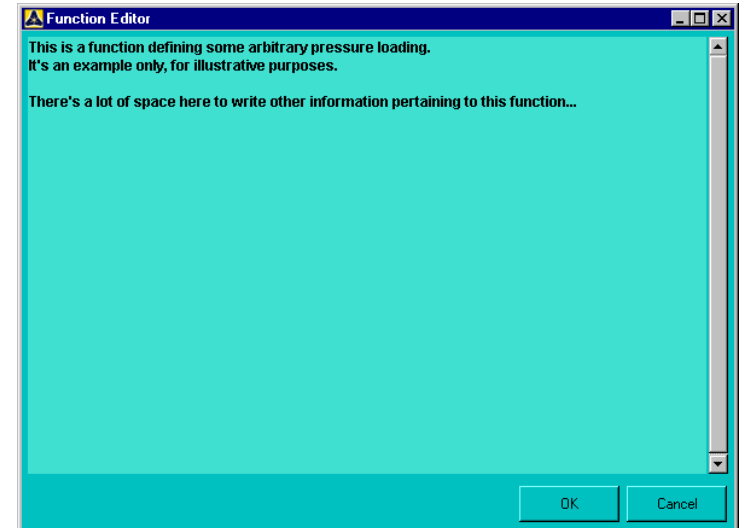
Steps for Using the Function Editor

Comments can also be written for each function

- GUI Method:

Function Editor > File > Comments

Since functions are stored in a separate ASCII text file, these comments provide useful information on details of the equation. Function files can be shared among users, similar to material and/or section properties saved to files.



2. Save the generated function

- GUI Method:

Function Editor > File > Save

Function files end with the extension *.func, although this is arbitrary. The file is actually an XML file along with some APDL commands.

Steps for Using the Function Editor

3. Load the save function into a table parameter

– GUI Method:

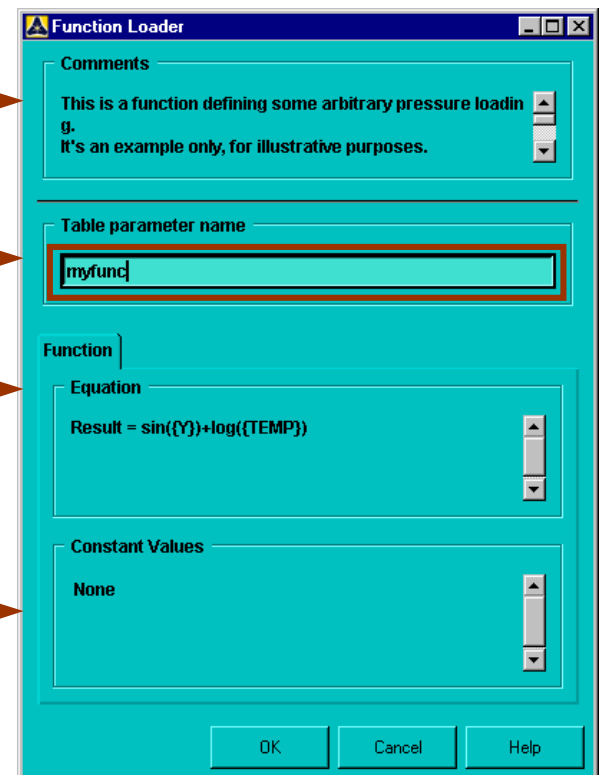
Utility Menu > Parameters > Functions > Read from File ...
Select the *.func file to read in.

Any comments written will appear in the “Comments” header.

Enter a table parameter name to use. In this example, we define a table name “MYFUNC”

The function will be shown in the “Equation” section.

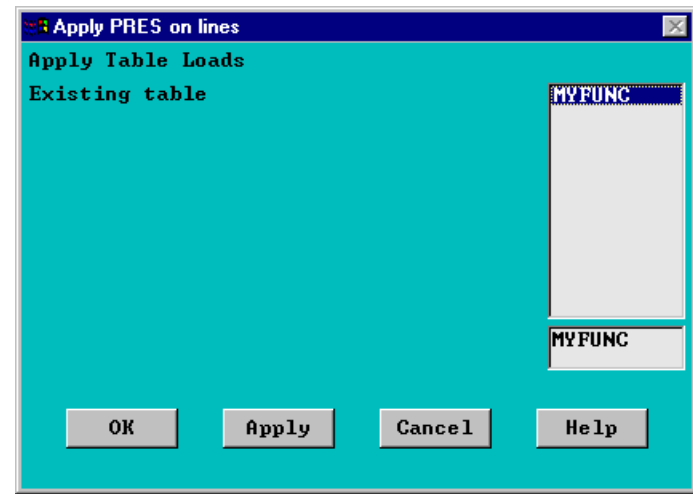
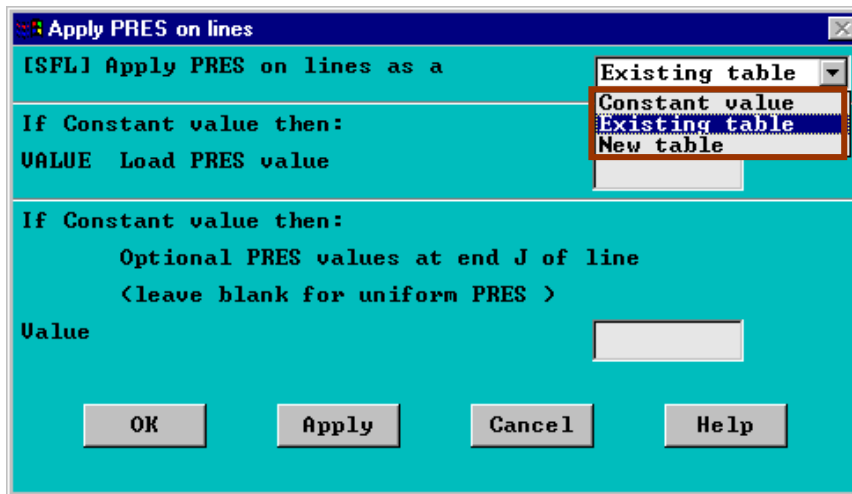
If there are any constant parameters or material parameters, these would need to be filled out in the “Constant Values” section. For example, if the function included a dependence on the thermal conductivity, the material reference number would need to be entered in this section.



Steps for Using the Function Editor

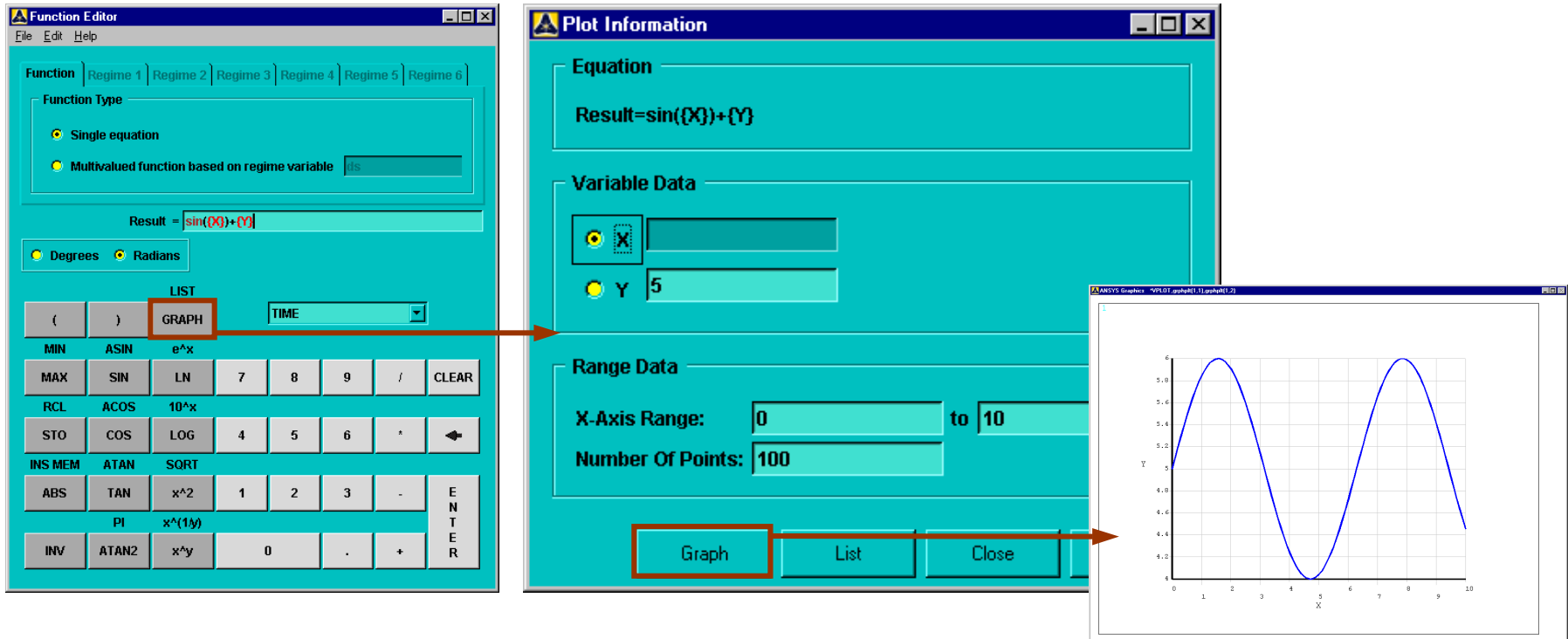
4. Use the table parameter when applying loads/b.c.

Similar to the case of regular tabular loads, as you apply the boundary condition, use an “Existing table” and then select that table parameter name created earlier (in this case, “MYFUNC”).



Verifying Function B.C.

- Similar to tabular b.c., function b.c. can also be plotted with the boundary condition symbols on geometry.
- Functions can also be plotted or listed in the Function Editor (at 6.0) to verify the equation, as shown below.



Using Functions in Batch Mode

- After reading a function into a table parameter (step 3 of previous example), the commands necessary to define the table have been copied in the log file.

An example from the previous slide is shown on the right.

These APDL commands can be used in an input file to generate the function table. To apply the function to a load or b.c., enclose the function table name in parentheses “%”, as usual.

Note that the function definition is in a specific format, not meant to be directly input by the user.

```
*DEL, _FNCNAME
*DEL, _FNCMTID
*SET, _FNCNAME, 'myfunc'
! /INPUT, C:\docs\temp\temp.func
*DIM, %_FNCNAME%, TABLE, 6, 4, 1
!
! Begin of equation: sin({Y})+log({TEMP})
%_FNCNAME%(0,0,1)= 0.0, -999
%_FNCNAME%(2,0,1)= 0.0
%_FNCNAME%(3,0,1)= 0.0
%_FNCNAME%(4,0,1)= 0.0
%_FNCNAME%(5,0,1)= 0.0
%_FNCNAME%(6,0,1)= 0.0
%_FNCNAME%(0,1,1)= 1.0, -1, 9, 1, 3, 0, 0
%_FNCNAME%(0,2,1)= 0.0, -2, 6, 1, 5, 0, 0
%_FNCNAME%(0,3,1)= 0, -3, 0, 1, -1, 1, -2
%_FNCNAME%(0,4,1)= 0.0, 99, 0, 1, -3, 0, 0
! End of equation: sin({Y})+log({TEMP})
!-->
```


Summary

- **Tabular and function boundary conditions are quite powerful yet easy to implement, especially for complex load/b.c.**
 - If using externally-prepared data, use *tabular b.c.* and read into ANSYS via *TREAD.
 - If using a known equation, use *function b.c.* defined by the Function Editor.
 - Verify boundary conditions with /PNUM on the mesh or solid model. When postprocessing, current values can also be plotted. For function b.c., the equations can also be graphed in the Function Editor.
- **While not covered in this introductory memo, tables have other uses, including defining specific real constants/section properties for some elements.**
 - See CONTA171-174, SURF151-152, FLUID116, and SHELL181

Things to Keep in Mind

- **There are some slight differences in the capabilities of tabular and function b.c., so please keep these in mind**
 - **Currently, only tabular b.c. support the use of independent variables (i.e., nested tables).**
 - **Function b.c. support some additional primary variables, namely material properties, which tables presently do not.**
 - **Refer to references at end of this memo for more details**
- **In transient analyses, ensure that automatic time-stepping does not ‘skip’ over max/min load values.**
 - **For thermal analyses, use an array with TSRES to ensure that ‘peak’ excitation is captured**
 - **For structural analyses, TSRES is currently not supported. Use a minimum number of substeps or several load steps to ensure that ‘peak’ load values are recorded.**

References

- Ch. 2.6.14 “Applying Loads Using TABLE Type Array Parameters” in the *Basic Analysis Procedures Guide*
- Ch. 3.11 “Array Parameters” in the *APDL Programmer’s Guide*
- *DIM command notes in the *ANSYS Commands Reference*