目　　录

第一章 APDL是什么？ 1

第二章 在工具条上添加命令 2

2.1 修改工具条 2
2.2 嵌套工具条缩写 4

第三章 使用参数 5

3.1 参数 5
3.2 参数命名规则 5
3.2.1 从*STATUS 命令中隐藏参数 6
3.3 定义参数 6
3.3.1 在运行过程中给参数赋值 6
3.3.2 在启动时给参数赋值 6
3.3.3 赋 ANSYS 提供的值给参数 7
3.3.3.1 *GET 命令的用法 7
3.3.3.2 内嵌获取函数的用法 8
3.3.4 排列显示参数 10
3.4 删除参数 11
3.5 字符参数的用法 11
3.6 数字参数值的置换 12
3.6.1 防止置换 12
3.6.2 字符参数值的置换 12
3.6.2.1 强制置换 12
3.6.2.2 字符参数有效的其它地方 13
3.6.2.3 字符参数的限制 14
3.7 数字或字符参数的动态置换 14
3.8 参数公式 15
3.9 带参数的函数 15
3.10 保存、恢复、写参数 16
3.11 数组参数 17
3.11.1 数组的基础知识 17
3.11.2 数组参数示例 18
3.11.3 TABLE 类型数组参数 19
3.11.4 定义和列表显示数组参数 20
3.11.5 给数组元素赋值 20
3.11.5.1 给单独的数组元素赋值 20
3.11.5.2 填充数组向量 21

3.11.5.3 交互式编辑数组 21
3.11.5.4 使用*VREAD 命令用数据文件填充数组 22
3.11.5.5 使用* TREAD 命令用数据文件填充 TABLE 类型数组 23
3.11.6 插入值 26
3.11.6.1 把获取值存入数组参数或恢复数组参数值 27
3.11.6.2 列出数组参数 28
3.11.7 写数据文件 29
3.11.7.1 数据格式描述符 29
3.11.8 对数组参数的运算 31
3.11.8.1 对向量的运算 31
3.11.8.2 矩阵运算 33
3.11.8.3 用于向量和矩阵运算的命令 35
3.11.9 用图形表示数组参数向量 37

第四章 作为宏语言的APDL 41

4.1 什么是 APDL 宏 41
4.2 产生宏 41
4.2.1 宏文件命名规则 41
4.2.2 宏的搜索路径 42
4.2.3 在 ANSYS 中生成宏 43
4.2.3.1 使用*CREATE 43
4.2.3.2 使用 *CFWRITE 43
4.2.3.3 使用 Utility Menu>Macro>Create Macro 44
4.2.4 用文本编辑器生成宏 44
4.2.5 使用宏库文件 45
4.3 运行宏和宏库文件 45
4.4 局部变量 46
4.4.1 传递变量到宏 46
4.4.2 宏内的局部变量 47
4.4.3 宏外部的局部变量 47
4.5 在 APDL 中控制程序流 47
4.5.1 宏嵌套：在宏内调用子程序 47
4.5.2 无条件分支：GOTO 48
4.5.3 条件分支：*IF 命令 48
4.5.4 重复一个命令 50
4.5.5 循环： Do 循环 50
4.6 控制函数快速参考 50
4.7 在宏中使用 _STATUS 和 _RETURN 参数 52
4.8 在组和组件中使用宏 53
4.9 复习宏例子 54

第五章 GUI用户界面 57

5.1 提示用户输入某个参数的值 57
5.2 用户提示对话框 58

5.3 用宏显示消息
59

5.4 在宏中生成并维护状态条 60
5.5 在宏中进行拾取操作 61
5.6 在宏中调用对话框 62

第六章 加密宏 63

6.1 准备加密宏 63
6.2 生成加密宏 63
6.3 运行加密宏 64

第一章 APDL是什么？

APDL 即 ANSYS 参数化设计语言（ANSYS Parametric Design Language），它是一种解释性语言，可用来自动完成一些通用性强的任务，也可以用于根据参数来建立模型。APDL 还包括其它许多特性，诸如重复执行某条命令，宏，if-then-else 分支，do 循环，标量、向量及矩阵操作等。

APDL 不仅是设计优化和自适应网格划分等经典特性的实现基础，而且它也为日常分析提供了很多便利。本指南介绍了 APDL 的基本特性-参数，宏，分支，循环，重复和参数数组，并包含有一些简单的例子。一旦你很好地掌握了这种语言，你将会发现 APDL 无所不能。

第二章 在工具条上添加命令

可以随时利用 ANSYS 函数和宏往工具条上添加命令（在 4 .1 部分讲述如何生成宏）。通过定义缩写来达到这一目的。缩写就是 ANSYS 命令、GUI 函数名或宏名的化名（最多可包含 8 个字符）。例如，MATPROP 可以是一个列出材料特性的宏的缩写，SAVE_DB 是 SAVE 命令的缩写，QUIT 是函数 Fnc_/EXIT（调出"Exit from ANSYS"对话框）的缩写。

ANSYS 程序提供了两种途径来利用缩写：1.在命令行的起始处键入缩写（执行缩写代表的宏、命令等）。2.通过 ANSYS GUI 中工具条上的按钮来执行宏或命令。

下图中的工具条包含代表已存在缩写的按钮。

图 2-1 显示缺省按钮的工具条

一旦缩写，如 SAVE_DB，已预先被定义，那么在工具条中该缩写及其执行的功能就可用了。一个工具条最多可包含 100 个缩写（可嵌套工具条来扩展这一数目）。可以根据需要重新定义或删除缩写，但是缩写不能自动保存，必须明确地保存到一个文件中，并且每次执行 ANSYS 任务时都要重新加载。

2.1 修改工具条

生成缩写的途径有：通过*ABBR 命令；通过 Utility Menu > Macro > Edit Abbreviations 或 Utility Menu >MenuCtrls > Edit Toolbar 菜单项。最好通过菜单项生成缩写，原因是：

- 单击 OK 按钮就自动更新工具条（若用*ABBR 命令，则还需要用 Utility Menu >MenuCtrls > Update Toolbar 菜单项来使新的缩写显现在工具条上）。
- 可以很方便地按需要编辑缩写。

*ABBR 命令的语法及相应的对话框是：

*ABBR, Abbr, String

其中：

Abbr 表示将要显示在工具条上的缩写名，最多可包含 8 个字符。

String 为 Abbr 所代表的宏或命令的名称。如果 String 是一个宏的名称，该宏必须在宏的搜索路径之中。有关宏的更多信息在 4.1 部分。如果 String 涉及到 ANSYS 的拾取菜单或对话框（通过 UIDL），就需要指定为"Fnc_string"。例如，在定义上图"QUIT," "POWRGRPH," 和 "ANSYSWEB"缩写中，"Fnc_/QUIT," "Fnc_/GRAPHICS," 和 "Fnc_HomePage"都是唯一的 UIDL 函数名，用来确定与 QUIT, POWRGRPH,和 ANSYSWEB 缩写对应的 ANSYS 的拾取菜单或对话框。有关 UIDL 的更多信息在 5.6 部分。String 最多可包含 60 个字符，但不能含有以下字符：字符 "$" ，命令 C***, /COM, /GOPR, /NOPR, /QUIT, /UI, 或 *END 。

缺省的 ANSYS 工具条包含如下预定义的缩写：

*ABBR, SAVE_DB, SAVE

*ABBR, RESUM_DB, RESUME

*ABBR, QUIT, Fnc_/EXIT

*ABBR, POWRGRPH, Fnc_/GRAPHICS

*ABBR, ANSYSWEB, Fnc_HomePage

例如，要加一个按钮到工具条中来调用宏文件 mymacro.mac，如下图所示在 Utility Menu >MenuCtrls > Edit Toolbar 对话框中输入值。

图 2-2 加新缩写

新按钮被添加到如下图所示的工具条按钮条中。

图 2-3 新缩写的按钮

当执行的 ANSYS 任务发生改变时，工具条按钮也会改变。但是，工具条按钮被保存在数据库中，因此在 ANSYS 任务中执行任何一条"resume"命令都将仍然包含原来的工具条按钮。要保存定制的按钮定义，必须通过 Utility Menu >MenuCtrls > Save Toolbar 菜单项明确地把它们保存到文件中，并通过 Utility Menu >MenuCtrls > Restore Toolbar 菜单项（*ABBRES 命令）在每个 ANSYS 任务中进行恢复。可以在宏中编程完成这项功能。

注意－如果在同名文件中已经存在某些缩写，*ABBSAV 命令将覆盖它们。

缩写文件的格式就是一些用来产生缩写的 APDL 命令序列。因此，如果要编辑很多按钮或要改变其顺序，通过一个文本编辑器来实现是最方便的了。例如，下面的文件就是由保存缺省工具条按钮而产生的：

```
/NOPR
*ABB,SAVE_DB ,SAVE
*ABB,RESUM_DB,RESUME
*ABB,QUIT       ,Fnc_/EXIT
*ABB,POWRGRPH,Fnc_/GRAPHICS
*ABB,ANSYSWEB,Fnc_HomePage
/GO
```

*ABB 命令（*ABBR 的缩写形式）定义按钮。第一行的/NOPR 使下面的操作不被写入 log 文件，最后一行的/GO 使后面的操作被写入 log 文件。

2.2 嵌套工具条缩写

上面讲的保存-恢复特性使嵌套缩写成为可能。通过在一个按钮下嵌套缩写，可以定义专用的工具条（假如有很多缩写放在一个工具条下，将会引起混乱，很难找到正确的按钮）。为了嵌套缩写，可以简单地定义一个恢复缩写文件的缩写。例如，下面的命令定义 PREP_ABR 作为一个缩写，该缩写从文件 prep.abbr 中恢复缩写。

*ABBR,PREP_ABR,ABBRES,,PREP,ABBR

PREP_ABR 将作为一个按钮显示在工具条上。单击它，现在的按钮将被文件 prep.abbr 中所定义的那套按钮代替。

通过定义缩写来恢复文件，并在这些文件中包含缩写，就可以在某个特定的 ANSYS 任务中定义不受数目限制的缩写。甚至可以通过嵌套一些缩写文件把这一方法扩展到产生自己的菜单层次。在实现菜单层次的过程中，最好在每一个文件中添加一个"return"按钮的缩写以通过菜单返回。

第三章 使用参数

3.1 参数

参数是 APDL 的变量（它们更象 Fortran 变量，而不象 Fortran 参数）。不必明确声明参数类型。所有数值变量（整形或实形）都以双精度数储存。被使用但未声明的参数都被赋一个接近 0 的值，或"极小值"，大约为2^{-100}。例如，若参数 A 被定义为 A=B，但 B 没被定义，则赋给 A 一个极小值。

ANSYS 包含两种类型的参数：标量和数组。本章前一部分的内容适用于这两种类型。3.11 后的内容只适用于数组类型的参数。

括在单引号中的字符串（最多 8 个字符）可以赋给参数。APDL 提供的数组参数的类型有：数值，字符和表（一种特殊的数值数组，能自动产生插值）。

可以用一个参数（代替数值或字符串）作为任何一个 ANSYS 命令的变量，该参数被求值并被赋给命令变量。例如，把值 2.7 赋给参数 AA，然后执行命令：

N,12,AA,4

ANSYS 程序将把该命令解释为：

N,12,2.7,4　　(定义节点 12 为点 X=2.7，Y=4)。

3.2 参数命名规则

参数名称必须以字母开头，且只能包含字母、数值和下划线。

下面列出一些有效和无效的参数名：

有效参数名：

ABC

PI

X_OR_Y

无效参数名：

NEW_VALUE (超过 8 个字符)

2CF3 (以数值开头)

M&E (含非法字符"&")

在命名参数时，注意以下几点：

- 要避免参数名与经常使用的 ANSYS 标识字相同，如：
  自由度（DOF）标识字(TEMP, UX, PRES 等)；
  常用标识字(ALL, PICK, STAT 等)；
  用户定义标识字（如用 ETABLE 命令定义的标识字）；
  数组类型标识字(如 CHAR, ARRAY, TABLE 等)
- 记住，名称为从 ARG1 到 ARG9 和从 AR10 到 AR99 的参数被保留为局部参数。通常，局部参数用于宏中（见 4.4）。
- 参数名不能与用*ABBR 命令定义的缩写相同。有关缩写的更多内容在 2.1 中。

- 参数名不能以下划线(_)开头。这类参数名只能用于 GUI 和应用于 ANSYS 的宏中。

3.2.1 从*STATUS命令中隐藏参数

3.3.4 部分讲述了通过*STATUS 命令列表显示参数。可以通过参数的命名约定来从*STATUS 命令中隐藏参数：名称中以下划线(_)结束的任何参数，在执行*STATUS 命令都将不被显示。

当为很多其他用户开发 APDL 宏时，这一功能是十分有用的。可以用这一功能来建立宏，而 ANSYS 用户和其它的宏编程人员不能列表显示你定义的参数。

3.3 定义参数

除非特别说明，以下几节的内容对标量和数组类型参数都适用。从 3.11 后的内容只适用于数组类型的参数。

定义参数的方法主要有：可以把值赋给参数，也可以提取 ANSYS 提供的值，再把这些值赋给参数。还可以用*GET 命令或各种内嵌获取函数从 ANSYS 中提取值。下面的部分对其进行详细说明。

3.3.1 在运行过程中给参数赋值

可以用*SET 命令定义参数。如下面的例子：

```
*SET,ABC,-24
*SET,QR,2.07E11
*SET,XORY,ABC
*SET,CPARM,'CASE1'
```

也可以用"="作为一种速记符来调用*SET 命令（这更方便），其格式为 Name=Value，这里 Name 是指参数名，Value 是指赋给该参数的数值或字符。对于字符参数，赋给的值必须被括在单引号中，并不能超过 8 个字符。下面的例子说明"="的用法：

```
ABC=-24
QR=2.07E11
XORY=ABC
CPARM='CASE1'
```

在 GUI 中，可以直接在 ANSYS 输入窗口或标量参数对话框的"Selection"域（通过 Utility Menu>Parameters>Scalar Parameters 菜单项访问）中输入"="。

3.3.2 在启动时给参数赋值

当从操作系统的命令行启动 ANSYS 时，可以定义参数作为变量：仅仅只需在 ANSYS 的运行命令（与系统有关）之后按-Name Value 的格式输入参数定义即可。例如，下面定义了两个参数(parm1 and parm2)，分别赋值 89.3 和 -0.1：

```
ansys55 -parm1 89.3 -parm2 -0.1
```

最好不要在启动时指派一个或两个字符参数名，以避免与 ANSYS 命令行选项发生冲突。

注意－ UNIX 操作系统把单引号和有些非文字数字字符作为专用符号。定义字符参数时，必须在单引号前插入后斜线(\)以免误会。例如，下面定义两个字符参数，分别赋值 `filename' 和 `200' ：

ansys55 -cparm1 \'filename\' -cparm2 \'200\'

如果不是用命令行形式启动 ANSYS，可以通过 Interactive 或 Batch 菜单项（仍采用上面描述的-Name Value 格式）来定义参数。

假如在启动时要定义很多参数，更加方便的做法是在 start5x.ans 文件中或一个能用 /INPUT 命令加载的单独文件中进行参数定义。

3.3.3 赋ANSYS 提供的值给参数

ANSYS 提供了两种方法来从 ANSYS 中提取数据：

1. *GET 命令，从某个特定的项目中提取数据并赋给某个特定的参数；
2. 内嵌获取函数，可在运行时使用。每个获取函数从某个特定的项目中提取特定的数据。

3.3.3.1 *GET命令的用法

*GET 命令（Utility Menu>Parameters>Get Scalar Data）从某个特定的项目（一个点、一个单元、一个面等）中提取 ANSYS 提供的数据并赋给某个用户命名的参数。各种关键词、标识字和数字结合在一起来确定被提取的项目。例如，*GET,A,ELEM,5,CENT,X 返回单元 5 的质心的 X 坐标值并赋给参数 A。

*GET 命令的使用格式为：

*GET,Par,Entity,ENTNUM,Item1,IT1NUM,Item2,IT2NUM

这里，

- Par 是将被赋值的参数名；
- Entity 是被提取项目的关键词，有效的关键词是 NODE, ELEM, KP, LINE, AREA, VOLU 等，在 ANSYS Commands Reference（ANSYS 命令参考手册）中的*GET 部分对之有完整的说明；
- ENTNUM 是实体的编号（若为 0 指全部实体）；
- Item1 是指某个指定实体的项目名。例如，如果 Entity 是 ELEM, 那么 Item1 要么是 NUM (选择集中的最大或最小的单元编号) ，要么是 COUNT (选择集中的单元数目)。在 ANSYS Commands Reference（ANSYS 命令参考手册）中的*GET 部分对每种实体的 Item1 值有完整的说明。

可以把*GET 命令看成是对一种树型结构从上至下的路径搜索，即从一般到特殊的确定。

可用下面的例子来说明*GET 命令的用法。下面的第一条命令用于获得单元 97 的材料属性（MAT 参考号）并赋给单元 BCD：

```
*GET,BCD,ELEM,97,ATTR,MAT        ! BCD = 单元 97 的材料号
*GET,V37,ELEM,37,VOLU            ! V37 = 单元 37 的体积
*GET,EL52,ELEM,52,HGEN           ! EL52 = 在单元 52 生成的热值
*GET,OPER,ELEM,102,HCOE,2        ! OPER =单元 102 面 2 上的热系数
*GET,TMP,ELEM,16,TBULK,3         ! TMP = 单元 16 面 3 上的体积温度
*GET,NMAX,NODE,,NUM,MAX          ! NMAX = 最大激活节点数
```

*GET,HNOD,NODE,12,HGEN ! HNOD = 在节点 12 生成的热值
*GET,COORD,ACTIVE,,CSYS ! COORD = 激活的坐标系值

3.3.3.2 内嵌获取函数的用法

对于某些项目，可以用内嵌的获取函数来代替*GET 命令。获取函数返回项目的值并直接用于当前运行之中。这样就不必先把值赋给参数，然后再在运行中调用该参数，从而可以省去起中间作用的参数。例如，要计算两个节点的 X 坐标的平均值，可以采用*GET 函数：

1. 使用下面的命令来把节点 1 的 X 坐标值赋给参数 L1：

   *GET,L1,NODE,1,LOC,X

2. 再使用*GET 命令来把节点 2 的 X 坐标值赋给参数 L2；
3. 计算中间值 MID=(L1+L2)/2.。

更简便的方法是使用节点坐标的获取函数 NX(N)，该函数返回节点 N 的 X 坐标值。这样就可以不用中间参数 L1 和 L2。如下所示：

MID=(NX(1)+NX(2))/2

获取函数的参数可以是参数也可以是其它的获取函数。例如，获取函数 NELEM(ENUM,NPOS) 返回在单元 ENUM 上 NPOS 处的节点编号，则联合函数 NX(NELEM(ENUM,NPOS))返回该节点的 X 坐标值。

下面的表总结了所有可用的获取函数：

获取函数	提取值
实体状态:	
NSEL(N)	节点 N 的状态(-1=未被选择, 0=未定义, 1=被选择)
ESEL(E)	单元 E 的状态(-1=未被选择, 0=未定义, 1=被选择)
KSEL(K)	关键点 K 的状态(-1=未被选择, 0=未定义, 1=被选择)
LSEL(L)	线 L 的状态(-1=未被选择, 0=未定义, 1=被选择)
ASEL(A)	面 A 的状态(-1=未被选择, 0=未定义, 1=被选择)
VSEL(V)	体 V 的状态(-1=未被选择, 0=未定义, 1=被选择)
选择下一个实体：	
NDNEXT(N)	选择节点编号大于 N 的下一个节点
ELNEXT(E)	选择单元编号大于 E 的下一个单元
KPNEXT(K)	选择关键点编号大于 K 的下一个关键点
LSNEXT(L)	选择线编号大于 L 的下一条线
ARNEXT(A)	选择面编号大于 A 的下一个面
VLNEXT(V)	选择体编号大于 V 的下一个体
定位：	
CENTRX(E)	单元 E 的质心在总体笛卡儿坐标系中的 x 坐标值
CENTRY(E)	单元 E 的质心在总体笛卡儿坐标系中的 y 坐标值
CENTRZ(E)	单元 E 的质心在总体笛卡儿坐标系中的 z 坐标值
NX(N)	节点 N 在当前激活坐标系中的 x 坐标值
NY(N)	节点 N 在当前激活坐标系中的 y 坐标值
NZ(N)	节点 N 在当前激活坐标系中的 z 坐标值
KX(K)	关键点 K 在当前激活坐标系中的 x 坐标值
KY(K)	关键点 K 在当前激活坐标系中的 y 坐标值

KZ(K)	关键点 K 在当前激活坐标系中的 z 坐标值
LX(L,LFRAC)	线 L 的长度百分数为 LFRAC (0.0 to 1.0)处的 X 坐标值
LY(L,LFRAC)	线 L 的长度百分数为 LFRAC (0.0 to 1.0)处的 Y 坐标值
LZ(L,LFRAC)	线 L 的长度百分数为 LFRAC (0.0 to 1.0)处的 Z 坐标值
就近定位：	
NODE(X,Y,Z)	距点 X,Y,Z 最近的被选择的节点的编号（在当前激活坐标系中；符合条件的关键点中编号最小者）
KP(X,Y,Z)	距点 X,Y,Z 最近的被选择的关键点的编号（在当前激活坐标系中；符合条件的关键点中编号最小者）
距离：	
DISTND(N1,N2)	节点 N1 和节点 N2 之间的距离
DISTKP(K1,K2)	关键点 K1 和关键点 K2 之间的距离
DISTEN(E,N)	单元 E 的质心和节点 N 之间的距离。质心由单元上选择的节点确定。
角度：	
ANGLEN(N1,N2,N3)	两条线之间的夹角（由三个节点确定，其中 N1 为顶点）。单位缺省为弧度。
ANGLEK(K1,K2,K3)	两条线之间的夹角（由三个关键点确定，其中 K1 为顶点）。单位缺省为弧度。
离某实体最近：	
NNEAR(N)	最接近节点 N 的节点
KNEAR(K)	最接近关键点 K 的关键点
ENEARN(N)	最接近节点 N 的单元。单元位置由选择的节点确定。
面积：	
AREAND(N1,N2,N3)	由节点 N1, N2, 和 N3 围成的三角形的面积
AREAKP(K1,K2,K3)	由关键点 K1,K2, 和 K3 围成的三角形的面积
ARNODE(N)	与节点 N 相连的被选择单元在节点 N 上分配的面积。对于二维平面实体，返回与节点 N 相连边界的面积；对于轴对称实体，返回与节点 N 相连边表面的面积；对于三维体实体，返回与节点 N 相连面的面积。
法向：	
NORMNX(N1,N2,N3)	节点 N1, N2, 和 N3 确定平面的法线与 X 轴的夹角的余弦值
NORMNY(N1,N2,N3)	节点 N1, N2, 和 N3 确定平面的法线与 Y 轴的夹角的余弦值
NORMNZ(N1,N2,N3)	节点 N1, N2, 和 N3 确定平面的法线与 Z 轴的夹角的余弦值
NORMKX(K1,K2,K3)	关键点 K1,K2, 和 K3 确定平面的法线与 X 轴的夹角的余弦值
NORMKY(K1,K2,K3)	关键点 K1,K2, 和 K3 确定平面的法线与 Y 轴的夹角的余弦值
NORMKZ(K1,K2,K3)	关键点 K1,K2, 和 K3 确定平面的法线与 Z 轴的夹角的余弦值
关联：	
ENEXTN(N,LOC)	与节点 N 相连的单元。若有很多单元与节点 N 相连，则由 LOC 定位。列表结束时返回零。
NELEM(E,NPOS)	单元 E 中在 NPOS (1-20)位置上的节点号。
表面：	
ELADJ(E,FACE)	与单元 E 的某个表面号(FACE)邻近的单元。面号与面载荷关键号

	相同。仅仅考虑那些有相同维数和形状的单元。若邻近的单元多于一个，则返回-1，若无邻近单元，返回 0。
NDFACE(E,FACE,LOC)	单元 E 的某个表面(FACE)上的 LOC 处的节点。面号与面载荷关键号相同。LOC 指表面上的节点位置（对于 IJLK 表面，LOC=1 指节点 I，2 指节点 J 等）
NMFACE(E)	包含选定节点的单元 E 的表面号。面号输出就是面载荷关键号。如果一个面上出现多个载荷关键号(例如线单元和面单元), 该面上的最小载荷关键号将被输出。
ARFACE(E)	对于二维平面实体和三维体实体，返回包含选定节点的单元 E 的表面面积。对于轴对称单元，返回总表面积（360 度）。
自由度结果：	
UX(N)	节点 N 在 X 向的结构位移
UY(N)	节点 N 在 Y 向的结构位移
UZ(N)	节点 N 在 Z 向的结构位移
ROTX(N)	节点 N 绕 X 向的结构转角
ROTY(N)	节点 N 绕 Y 向的结构转角
ROTZ(N)	节点 N 绕 Z 向的结构转角
TEMP(N)	节点 N 上的温度
PRES(N)	节点 N 上的压力
VX(N)	节点 N 在 X 向的流动速度
VY(N)	节点 N 在 Y 向的流动速度
VZ(N)	节点 N 在 Z 向的流动速度
ENKE(N)	在节点 N 上的湍流动能 (FLOTRAN)
ENDS(N)	在节点 N 上的湍流能量耗散 (FLOTRAN)
VOLT(N)	节点 N 处的电压
MAG(N)	在节点 N 上的磁标势
AX(N)	在节点 N 上的 X 向磁矢势
AY(N)	在节点 N 上的 Y 向磁矢势
AZ(N)	在节点 N 上的 Z 向磁矢势

3.3.4 排列显示参数

一旦定义了参数，就可以用*STATUS 命令把它们排列显示出来。如果仅用*STATUS 命令（没有附加参数），将列表显示目前所有已定义的参数。下面的例子说明了该命令的用法及典型的排列显示：

```
*STATUS
PARAMETER STATUS-                (     5 PARAMETERS DEFINED)
NAME            VALUE            TYPE       DIMENSIONS
ABC        -24.0000000        SCALAR
HEIGHT      57.0000000         SCALAR
QR         2.070000000E+11    SCALAR
X_OR_Y     -24.0000000        SCALAR
CPARM        CASE1              CHARACTER
```

通过 Utility Menu>List>Other>Parameters 或 Utility Menu>List>Status>Parameters>All Parameters 菜单项也可以得到参数的列表显示。

注意－以下划线(_)开头或结尾的参数不能由*STATUS 命令显示出来。

可以通过给*STATUS 命令定义附加参数来单独显示某些参数。下面的例子说明了如何显示参数 ABC 的状态：

```
*STATUS,ABC
PARAMETER STATUS- abc        (    5 PARAMETERS DEFINED)
NAME          VALUE          TYPE      DIMENSIONS
ABC       -24.0000000       SCALAR
```

也可以通过 Utility Menu>List>Other>Named Parameter 或 Utility Menu>List>Status>Parameters>Named Parameters 菜单项指定参数的列表显示。

注意－虽然 ANSYS 最多允许有 1000 个参数，但是由于 GUI 和 ANSYS 宏需要用到一些参数，所以用户可用参数不到 1000 个。用户界面定义的参数（内部参数）数目可由*STATUS 命令列出。*GET,par,PARM,,MAX 命令返回所有已定义参数的数目。

3.4 删除参数

可通过两种途径来删除参数：

1. 使用"="命令，其右边为空。例如，使用该命令来删除参数 QR：
   QR=
2. 使用*SET 命令(Utility Menu>Parameters>Scalar Parameters)，但不给参数赋值。例如，使用该命令来删除参数 QR：
   *SET,QR,

令某个数值参数为 0 并没有删除该参数。同样，令某个字符参数为空的单引号(` `)或单引号中为空格也没有删除该参数。

3.5 字符参数的用法

一般地，字符参数用来提供文件名和扩展名。先把文件名赋给某个字符参数，然后，在需要用到文件名的地方用对应的参数来代替。同样，文件扩展名也可以先赋给某个字符参数，然后，在需要用到文件扩展名的地方用对应的参数代替（如作为 Ext 命令的参数）。这样，在批处理模式中，只需在输入文件中简单地改变字符参数的初始值就改变了用于多重运行的文件名。

注意－字符参数最多只能有 8 个字符。

下面说明字符参数的一般用法：

- 作为命令的参数（若该参数为文字数字类型）；
- 在使用*USE 命令时，作为代表宏名的参数（Utility Menu>Macro>Execute Data Block）；
  NAME='MACRO' ! MACRO 为宏文件名
  *USE,NAME ！调用 MACRO 宏
- 作为被调用宏的参数，该宏可由*USE 命令调用或作为”未知命令”宏。允许使用下面的任何方法：
  ABC='SX'

```
*USE,NAME,ABC
或
*USE,NAME,'SX'

DEF='SY'
NEWMACRO,DEF        ！调用已经存在的宏文件 NEWMACRO.MAC
或
NEWMACRO,'SY'
```

3.6 数字参数值的置换

只要在有关数字命令的地方用到参数，该参数值都会被自动置换。假如没有给该参数赋值（即该参数还没被定义），程序会自动赋给它一个接近 0 的值(2^{-100})，通常不会发出警告。

注意－大多数情况下，某参数在一个命令中使用之后，再被定义，不会再更新该命令。(除命令 /TITLE, /STITLE, *ABBR, 和 /TLABEL 之外。详细说明见 3.6.2.1 部分。)例如：

```
Y=0
X=2.7
N,1,X,Y        ！节点 1 在 (2.7,0)
Y=3.5          ！重新定义参数 Y 不会更新节点 1
```

3.6.1 防止置换

可以通过把参数名括在单引号(')中（如'XYZ'）来防止参数被置换，这时被使用的是文字串，所以，这个特性仅对非数字参数有用。

反过来，也可以通过把用于标题、子标题和文件名的参数名括在百分号(%)中，对其强迫进行置换。例如，

/TITLE, TEMPERATURE CONTOURS AT TIME=%TM%

在这个标题中，参数 TM 的数值被置换了。注意－一旦该标题被使用，参数即被置换。

3.6.2 字符参数值的置换

在文字数字命令域中使用字符参数通常会自动地导致该字符参数值的置换。下面说明强制置换和对字符参数的限制。

3.6.2.1 强制置换

和数字参数一样，也能在某些不会发生置换的情况下强迫字符参数进行置换。只要把字符参数名括在百分号(%)中就可以达到该目的。以下命令可以实现对字符参数的强制置换：

- /TITLE 命令 (标题域)，为各种打印输出指定标题。
- /STITLE 命令 (标题域)，指定子标题，同/TITLE。 (不能在 GUI 中直接得到 /STITLE

- /TLABEL 命令 (文本域)，为注释指定文本串。
- /SYP 命令 (ARG1 - ARG8 域)，传递命令（包括参数）到操作系统。 (不能在 GUI 中直接得到/SYP 命令)
- *ABBR 命令 (缩写域)，定义缩写。

强制置换在以下类型的域中也有效：

- 任何文件名或扩展名命令参数。这些参数应用到诸如/FILENAME, RESUME, /INPUT, /OUTPUT, 和 FILE 等命令中（在这些域中也允许直接参数置换）。
- 任何 32 位字符域：典型的例子是目录路径，它被用于很多命令。（在这些域中也允许直接参数置换）
- 在任何命令名域作为命令名。也可在域 1 中作为一个"未知命令"的宏名。例如：

```
R='RESUME'
%R%,MODEL,DB
```

以下命令输入方式说明了如何对一个子标题和目录名进行强制置换：

```
A='TEST'
B='.RST'
C='/ANSYS'
D='/MODELS/'
/STITLE,,RESULTS FROM FILE %C%%D%%A%%B%
SUBTITLE    1 =
RESULTS FROM FILE /ANSYS/MODELS/TEST.RST
/POST1
FILE,A,RST,%C%%D%          ！从文件/ANSYS/MODELS/TEST.RST 读结果
```

3.6.2.2 字符参数有效的其它地方

除已经讨论的一般应用之外，在某些特定的场合，应用字符参数会带来更多的便利。下面的内容描述了所涉及的命令和用法。

- ***ASK**

该命令用于用户给字符标量参数赋值时弹出的提示字符串（最多 8 个字符，且括在单引号中）。该命令不能由 GUI 直接得到。

- ***CFWRITE**

该命令把 ANSYS 命令写到由*CFOPEN 打开的文件中。可用于写一个分配给该文件的字符参数。例如，*CFWRITE,B='FILE' 是有效的。*CFWRITE 和*CFOPEN 命令不能由 GUI 直接得到。

- ***IF 和 *ELSEIF**

字符参数可用于这两个命令的 VAL1 和 VAL2 参数。对于 Oper 参数，使用字符参数时，只有 EQ (等于) 和 NE (不等于)标识字是有效的。*IF 和 *ELSEIF 命令不能由 GUI 直接得到。例如：

```
CPARM='NO'
*IF,CPARM,NE,'YES',THEN
```

- ***MSG**

该命令的 VAL1 到 VAL8 参数均为字符参数。数据描述符%C 用于在格式行中指明字符

数据（必须接在*MSG 命令之后）。%C 与 FORTRAN 中的描述符 A8 类似。*MSG 命令不能由 GUI 直接得到。

- **PARSAV 和 PARRES**

前一个命令把字符参数保存到一个文件中（通过 PARSAV 命令或 Utility Menu>Parameters> Save Parameters 菜单项），后一个命令从文件中恢复参数（通过 PARRES 命令或 Utility Menu>Parameters> Restore Parameters 菜单项）

- ***VREAD**

该命令(Utility Menu>Parameters>Array Parameters>Read from File)用于从某个文件中读取字符参数并生成一个字符数组参数。FORTRAN 中的字符描述符(A)用于*VREAD 命令后的格式行中。

- ***VWRITE**

该命令(menu path Utility Menu>Parameters>Array Parameters>Write to File)可用来以某种格式化的顺序把字符参数数据写到一个文件中。FORTRAN 中的字符描述符(A)可用于*VWRITE 命令后的格式行中。

3.6.2.3 字符参数的限制

虽然字符参数和数字参数有很多相同的功能，但是在有些场合字符参数是无效的：

- 在*SET, *GET, *DIM, 和 *STATUS 命令中，Par 参数对应的字符参数是不能被置换的。
- 对于字符数组参数，不能应用交互式编辑方式（*VEDIT 命令）。
- 向量运算命令，诸如*VOPER, *VSCFUN, *VFUN, *VFILL, *VGET 和*VITRP，不能用于字符数组参数。
- 对字符参数进行运算时，*VMASK 和*VLEN 命令只能应用于*VWRITE 和*VREAD 命令中。
- 字符参数不能用于包括加、减、乘等运算的参数公式中。

3.7 数字或字符参数的动态置换

应用/TITLE, /STITLE, *ABBR, 和 /TLABEL 命令时会发生参数的动态置换。动态置换允许使用参数被修改后的值，即使使用该参数的命令还没有被调用。

例如：

```
XYZ='CASE 1'
/TITLE,This is %XYZ%
APLOT
```

标题"This is CASE 1" 将显示在面区域。

若改变 XYZ 的值，那么在接下来的绘图中即使没有调用/TITLE 命令，也将显示新的标题。

```
XYZ='CASE 2'
```

标题 "This is CASE 2" 将显示在以后的绘图中。

3.8 参数公式

参数公式包括对参数和数值的运算，如加、减、乘、除等。例如：

```
X=A+B
P=(R2+R1)/2
D=-B+(E**2)-(4*A*C)        ！求值 D = -B + E2 - 4AC
XYZ=(A<B)+Y**2             ！如果 A I 小于 B，XYZ = A + Y2；
                           ！否则 XYZ = B + Y2
INC=A1+(31.4/9)
M=((X2-X1)**2-(Y2-Y1)**2)/2
```

下面是 APDL 算子的完全列表：

运算符号	操作
+	加
-	减
*	乘
/	除
**	求幂
<	小于
>	大于

也可以如上例一样使用圆括号。ANSYS 运算的顺序如下所示：

1. 圆括号中的运算 (最里面最优先)
2. 求幂 (从右到左)
3. 乘和除 (从左到右)
4. 一元联合 (例如 +A 或-A)
5. 加和减 (从左到右)
6. 逻辑判断 (从左到右)

因此一个诸如 Y2=A+B**C/D*E 的公式按如下顺序求值：最先求 B**C，第二步/D，第三步*E，最后+A。为了更清楚，可以在公式中使用圆括号。圆括号最多可嵌套 4 层，在每套圆括号中最多可有 9 次运算。一般来说，在公式的运算符之间不要有空格。特别是在*之前不能有空格，这是因为如果这样，接下来的输入行（以*开头）将被作为一条命令来解释，而不再是公式的一部分了。

3.9 带参数的函数

一个带参数的函数是数学运算的程序序列，并返回一个值，例如 SIN(X), SQRT(B), 和 LOG(13.2)。下面的表完整地列出了当前可用的 ANSYS 函数：

标准 FORTRAN 77 函数

ABS(x)	X 的绝对值
SIGN(x,y)	X 的绝对值，但取 y（正负）符号。y=0 时结果取正号。
EXP(x)	X 的指数值
LOG(x)	X 的自然对数值(ln (x))
LOG10(x)	X 的常用对数值(log10(x))

SQRT(x)	X 的平方根值
NINT(x)	X 的整数部分
MOD(x,y)	x/y 的余数部分。若 y=0，则返回 0
RAND(x,y)	在 x 到 y 范围内产生随机数（一致分布）（x 为下限，y 为上限）
GDIS(x,y)	生成平均值为 x 且偏差为 y 的正态分布的随机数
SIN(x),COS(x), TAN(x)	X 的正弦、余弦及正切值。X 的缺省单位为弧度，但可用*AFUN 命令转化为度数。
SINH(x), COSH(x), TANH(x)	X 的双曲线正弦、余弦及正切值。
ASIN(x), ACOS(x), ATAN(x)	X 的反正弦、反余弦及反正切值。对于 ASIN 和 ACOS ，X 必须在 -1.0 和 +1.0 之间。输出的缺省单位为弧度，但可用*AFUN 命令转化为度数。对于 ASIN 和 ATAN，输出值的范围在-pi/2 到+pi/2 之间；对于 ACOS，输出值的范围在 0 到 pi 之间
ATAN2(y,x)	y/x 的反正切值。输出的缺省单位为弧度，但可用*AFUN 命令转化为度数。输出值的范围在-pi 到+pi 之间。
VALCHR (CPARM)	返回 CPARM 的数字值。(如果 CPARM 是一个数值则返回 0.0)
CHRVAL (PARM)	数字参数 PARM 的字符值。小数位置数取决于数值大小。
UPCASE (CPARM)	把 CPARM 转化为大写
LWCASE (CPARM)	把 CPARM 转化为小写

下面是一些带参数函数的例子：

```
PI=ACOS(-1)                  ! PI = -1 的反余弦值, PI 的精确度由机器确定
Z3=COS(2*THETA)-Z1**2
R2=SQRT(ABS(R1-3))
X=RAND(-24,R2)               ! X =  在-24 和 R2 的随机值
*AFUN,DEG                    ！ 把角度的单位转换为度数
THETA=ATAN(SQRT(3))          ! THETA 等于 60 度
PHI=ATAN2(-SQRT(3),-1)       ! PHI  等于-120 度
*AFUN,RAD                    ！ 把角度的单位转换为弧度
X249=NX(249)                 ！ 节点 249 的 X 轴坐标
SLOPE=(KY(2)-KY(1))/(KX(2)-KX(1))
                             ！ 连接关键点 1 和 2 的线的斜率
CHNUM=CHRVAL(X)              ! CHNUM = X 的字符值
UPPER=UPCASE(LABEL)          ! UPPER =  参数 LABEL 的大写字符
```

3.10 保存、恢复、写参数

如果要在其他 ANSYS 任务中使用当前定义的参数，可以先把它们写入一个文件中，然后读取（恢复）该文件。读取文件时，可以全部代替当前定义的参数，也可以只把它们加到当前定义的参数中（会覆盖已存在的参数）。

可用 PARSAV 命令把参数写入一个文件中(Utility Menu>Parameters>Save Parameters)。

参数文件是 ASCII 文件，主要由 APDL *SET 命令组成，该命令用来定义各种参数。用下面的例子来说明参数文件的格式：

```
/NOPR
```

```
*SET,A          ,   10.00000000000
*SET,B          ,   254.3948750000
*SET,C          ,'string   '
*SET,_RETURN , 0.0000000000000E+00
*SET,_STATUS ,   1.000000000000
*SET,_ZX        ,'            '
/GO
```

可用 PARRES 命令从一个文件中读取参数(Utility Menu>Parameters>Restore Parameters) 。

若需要，最多可用 FORTRAN 实数格式写 10 个参数或数组到一个文件中。可以利用这一特性来写用于其他程序、报告等的输出文件。其对应的命令为*VWRITE(Utility Menu>Parameters>Array Parameters>Write to File)。关于该命令的内容在 3.11.7 部分。

3.11 数组参数

除数量参数（一个值）之外，还可以定义数组参数（多个值）。ANSYS 数组可以是一维（一列），二维（行和列）或三维（行、列和面）。

ANSYS 提供了三种数组类型：

ARRAY

这种类型与 FORTRAN 77 的数组类似，是缺省的数组类型。和 FORTRAN 77 的数组一样，行、列和面的下标从 1 开始，为连续的整形数，数组元素为整形或实形数。该类型数组最多可有 10242-1 行、255 列和 7 面。

CHAR

字符数组，每个元素包含不超过 8 个的文字数字字符，行、列和面的下标从 1 开始，为连续的整形数。该类型数组最多可有 10242-1 行、255 列和 7 面。

TABLE

这是一种特殊的数字数组类型，通过它，ANSYS 可以计算在数组中明确定义的元素之间的值（通过线性插值）。而且可以为每一行、列和面定义数组下标，下标为实数（不是整形数）。数组元素可以是整数，也可以是实数。在后面的讨论中可以看到，这一特性为数学运算提供了一个非常有力的工具。该类型数组最多可有 65,535 行、255 列和 7 面。

3.11.1 数组的基础知识

以下面的一个二维数组（ARRAY 或 CHAR）为例：它有 m 行长和 n 列宽，即其维数为 m 乘以 n。每行由行下标 i 确定，i 在 1 到 m 之间；每列由列下标 j 确定，j 在 1 到 n 之间。组成数组的数就是数组元素。每个元素由(i,j)确定，其中 i 是其行数，j 是其列数。

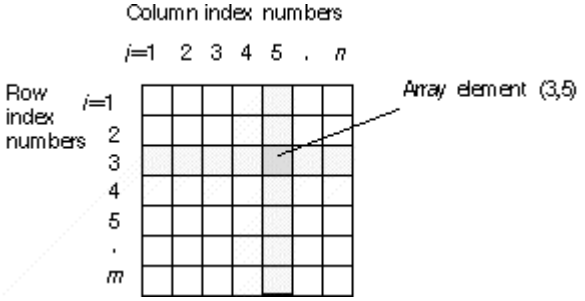


图 3-1 二维数组的图形表示

可以把以上的定义扩展到三维数组参数，三维数组有 m 行长、n 列宽和 p 个面。面下标为 k，变化范围从 1 到 p。每个数组元素由(i,j,k)确定。下图说明三维数组：

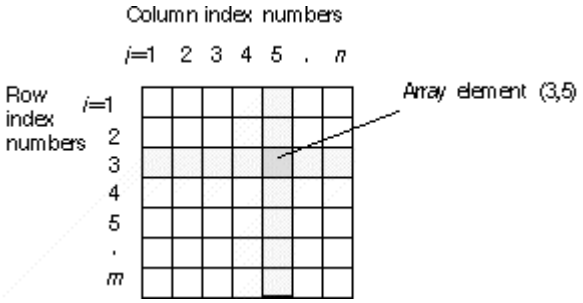


图 3-2 三维数组的图形表示

3.11.2 数组参数示例

ARRAY 类型的参数由离散的数组成，为了方便，这些数被安放在表格形式中。如下例：

$$\text{NTEMP} = \begin{bmatrix} -47.6 \\ -5.2 \\ 25.0 \\ 86.5 \\ 107.9 \\ 168.7 \\ 225.0 \end{bmatrix} \qquad \text{EVOLUM} = \begin{bmatrix} 0.085 \\ 0.01 \\ 0.265 \\ 1.00 \\ 0.832 \\ 0.52 \\ 1.032 \\ 0.002 \\ 0.697 \\ 0.01 \end{bmatrix}$$

$$\text{COMPSTRS} = \begin{bmatrix} 12152 & 814 & -386 & 202 & -82 & -1108 \\ 14848 & 1057 & -704 & 117 & -101 & -555 \\ 15490 & 1033 & -713 & 15 & -76 & 235 \\ 13899 & 786 & -348 & -103 & -45 & 848 \\ 10813 & 420 & -66 & -211 & -17 & 1065 \\ 7151 & 109 & 111 & -272 & 11 & 1052 \end{bmatrix}$$

参数 NTEMP 为表示选定节点处的温度的数组；NTEMP(1)=-47.6 表示节点 27 处的温度，NTEMP(2)=-5.2 表示节点 43 处的温度，等等。同样地，参数 EVOLUM 为表示单元体积的数组，参数 COMPSTRS 为表示节点压力的数组，每一列代表一个特定的方向(如 X, Y, Z, XY, YZ, XZ)。

CHAR 类型的数组参数的结构与 ARRAY 类型参数相似，只是其元素为字符串（最多 8 个字符）。以下为两个字符数组参数的例子：

$$\text{FILNAM} = \begin{bmatrix} \text{JOB1} \\ \text{JOB2} \\ \text{JOB3} \\ \text{JOB4} \\ \text{JOB5} \end{bmatrix} \qquad \text{EXTENS} = \begin{bmatrix} \text{LOG} \\ \text{ERR} \\ \text{DB} \\ \text{LIB} \\ \text{MAC} \end{bmatrix}$$

3.11.3 TABLE类型数组参数

一个TABLE类型的数组参数由安放在表格形式中的数(文字数字无效)组成,与ARRAY类型较为相似。但是，有三点主要区别：

- ANSYS 能计算在表数组中明确定义的元素之间的任意值（通过线性插值）；
- 表数组包含 0 行 0 列，与一般数组不同，下标值可以是实数。唯一的限制是下标值必须是增加的（不能减少）数值。必须通过行和列的下标值来明确声明某个值，否则，将被赋一“极小值”(7.888609052E-31)；
- 面的下标值放在该面的 0,0 处。

下图用来说明一个 TABLE 数组。下面插图显示了一个可以进行数值检索的表数组。 注意：检索定义是从"0"行和列值开始的。

如上例所示，初始化一个表数组时，必须设置：

- 每个面的 0,0 元素值为该面的下标值；
- 面 1 中 0 行里的列下标值。只有想从数组中得到数据时才会用到这些值。给数组中的元素赋值时，采用传统的行列下标即可。
- 面 1 中 0 列里的行下标值。同样地，只有想从数组中得到数据时才会用到这些值。给数组中的元素赋值时，采用传统的行列下标即可。

在任何面中都可以设置或改变行列的下标值，这些下标值将应用在所有面中。

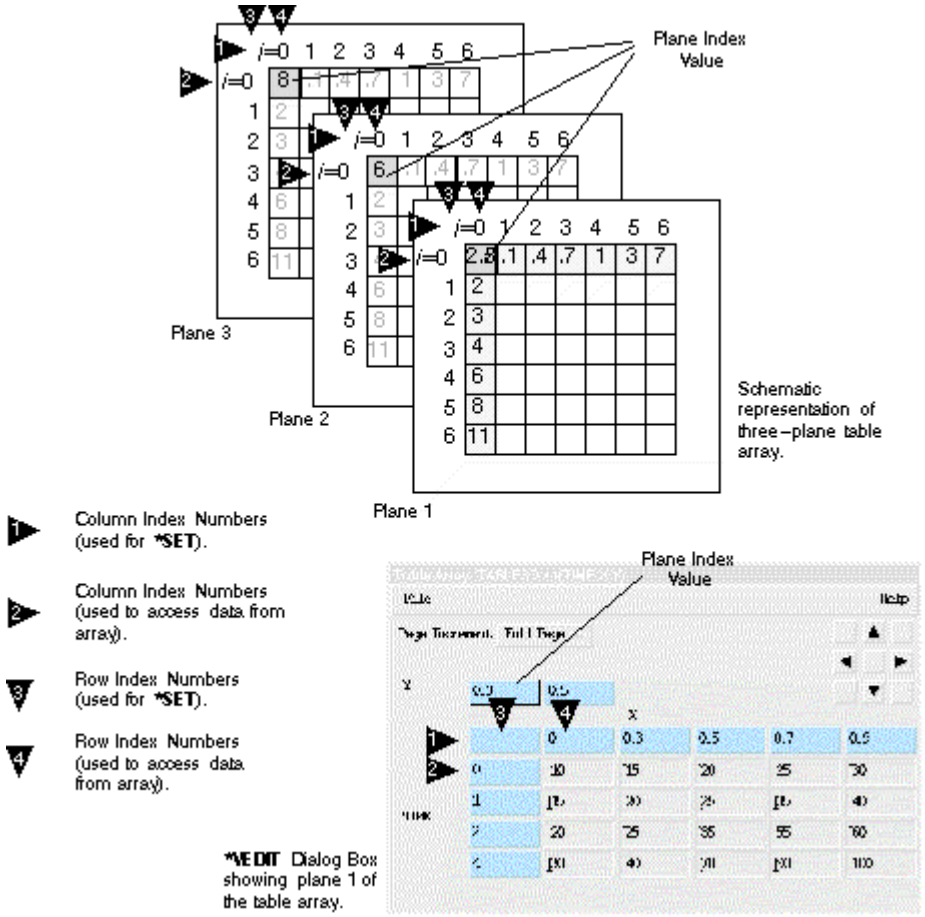


图 3-3 表数组曲线

3.11.4 定义和列表显示数组参数

要定义一个数组参数，首先必须通过*DIM 命令(Utility Menu>Parameters>Array Parameters>Define/Edit)来声明其类型和维数。

下面的例子说明如何使用*DIM 命令：

```
*DIM,AA,,4                ! 类型 ARRAY 为缺省类型,维数为 4[x1x1]
*DIM,XYZ,ARRAY,12         ! ARRAY 类型数组, 维数为 12[x1x1]
*DIM,FORCE,TABLE,5        ! TTABLE 类型数组, 维数为 5[x1x1]
*DIM,T2,,4,3              ! 维数为 4x3[x1]
*DIM,CPARR1,CHAR,5        ! CHAR 类型数组, 维数为 5[x1x1]
```

注意－ ARRAY 和 TABLE 类型的数组元素被初始化为 0（除 TABLE 类型的 0 行和 0 列之外，它们被初始化为“极小值”）。 CHAR 类型的数组元素被初始化为一个空格。

3.11.5 给数组元素赋值

可以通过以下途径给数组元素赋值：

- 通过*SET 命令或"="给单独的数组元素赋值；
- 用指定的或计算出的值来填充数组中的某个向量（列）（如*VFILL 命令）；
- 通过*VEDIT 对话框交互地给元素赋值；
- 从某个 ASCII 文件中读取值（*VREAD 或 *TREAD 命令）。

3.11.5.1 给单独的数组元素赋值

可以通过*SET 命令或"="给单独的数组元素赋值。除要定义一列数据（每个"="命令最多可定义 10 个数组元素值）外，和在标量数组中的用法相同。例如，定义一个维数为 12x1 的数组参数 XYZ，就需要使用两次"="命令。在下面的例子中，第一条命令定义了 8 个数组元素，第二条命令定义了剩下的 4 个数组元素。

```
XYZ(1)=59.5,42.494,-9.01,-8.98,-8.98,9.01,-30.6,51
XYZ(9)=-51.9,14.88,10.8,-10.8
```

$$XYZ = \begin{bmatrix} 59.5 \\ 42.494 \\ -9.01 \\ -8.98 \\ -8.98 \\ 9.01 \\ -30.6 \\ 51 \\ -51.9 \\ 14.88 \\ 10.8 \\ -10.8 \end{bmatrix}$$

Notice that the starting location of the array element is indicated by the row index number of the parameter (1 in the first command, 9 in the second command).

下面的例子说明了如何定义维数为 4x3 的数组参数 T2 的元素值，T2 在前面已经由*DIM 定义：

```
T2(1,1)=.6,2,-1.8,4          ! 定义(1,1),(2,1),(3,1),(4,1)
T2(1,2)=7,5,9.1,62.5         ! 定义(1,2),(2,2),(3,2),(4,2)
```

T2(1,3)=2E-4,-3.5,22,.01　　！定义(1,3),(2,3),(3,3),(4,3)

$$T2 = \begin{bmatrix} 0.6 & 7.0 & 0.0002 \\ 2.0 & 5.0 & -3.5 \\ -1.8 & 9.1 & 22.0 \\ 4.0 & 62.5 & 0.01 \end{bmatrix}$$

下面的例子定义 TABLE 类型的参数 FORCE 的元素值，FORCE 前面已定义过。

FORCE(1)=0,560,560,238.5,0

FORCE(1,0)=1E-6,.8,7.2,8.5,9.3

$$FORCE = \begin{array}{c} \\ 1E-6 \\ 0.8 \\ 7.2 \\ 8.5 \\ 9.3 \end{array} \begin{array}{c} 0 \\ \begin{bmatrix} 0.0 \\ 560.0 \\ 560.0 \\ 238.5 \\ 0.0 \end{bmatrix} \end{array}$$

The first "=" command defines the five array elements of the TABLE array FORCE. The second and third "=" commands redefine the index numbers in the j=0 column and i=0 row.

字符数组参数也能用 "=" 命令来定义。每个值最多可有 8 个字符，且必须括在单引号中。例如：

*DIM,RESULT,CHAR,3　　!维数为(3,1,1)的字符数组参数

RESULT(1)='SX','SY','SZ'　!给参数 RESULT 赋值

注意到，当定义一个数字数组参数时，要给出数组元素的起始位置（本例中，指定了行下标值 1）。

注意－ CHAR 不能被作为一个字符参数名，因为在*DIM 命令中会和 CHAR 标识字发生冲突。当 CHAR 被用于*DIM 命令中的第三个参数时，ANSYS 将替换赋给参数 CHAR 的字符串。

3.11.5.2 填充数组向量

可以使用*VFILL 命令(Utility Menu>Parameters>Array Parameters> Fill)来“填充” 一个 ARRAY 或 TABLE 向量（列）。

可以在 ANSYS 命令参考手册（ANSYS Commands Reference）中找到关于*VFILL 命令详细的语法参考信息。下面的例子说明了*VFILL 命令的用途：

*DIM,DTAB,ARRAY,4,3　　！定义维数为 4 x 3 的数字数组

*VFILL,DTAB(1,1),DATA,-3,8,-12,57　！四个数值被赋给向量 1（列 1）

*VFILL,DTAB(1,2),RAMP,2.54,2.54　！用起始值为 2.54，增量为 2.54 的数来

　　！填充向量 2（列 2）

*VFILL,DTAB(1,3),RAND,1.5,10　　！用 1.5 和 10 之间的随机数来填充向量 3

　　！结果依赖于随机数的产生

$$DTAB = \begin{bmatrix} -3 & 2.54 & 2.79901284 \\ 8 & 5.08 & 6.11392418 \\ -12 & 7.62 & 6.70205516 \\ 57 & 10.16 & 4.11487684 \end{bmatrix}$$

3.11.5.3 交互式编辑数组

*VEDIT(Utility Menu>Parameters>Array Parameters>Define/Edit)命令只适用于交互式模式，该命令会弹出一个数据输入对话框，通过该对话框可以编辑一个 ARRAY 或 TABLE（不

能是 CHAR）数组。该对话框提供了一些便利的特性：

- 为数组元素值提供一个电子数据表格形式的编辑器；
- 为大数组提供导向控制；
- 把某一行或列设置为某一指定值的初始化功能（仅对 ARRAY 有效）；
- 为移动行或列数据而设置的删除、拷贝和插入功能（仅对 ARRAY 有效）；

按对话框中的 Help 按钮可以得到该对话框的全部用法说明。

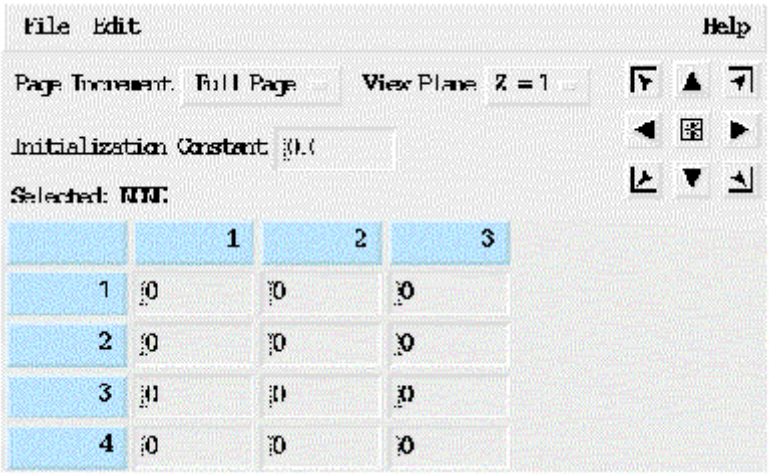


图 3-4ARRAY 类型数组的*VEDIT 对话框例子

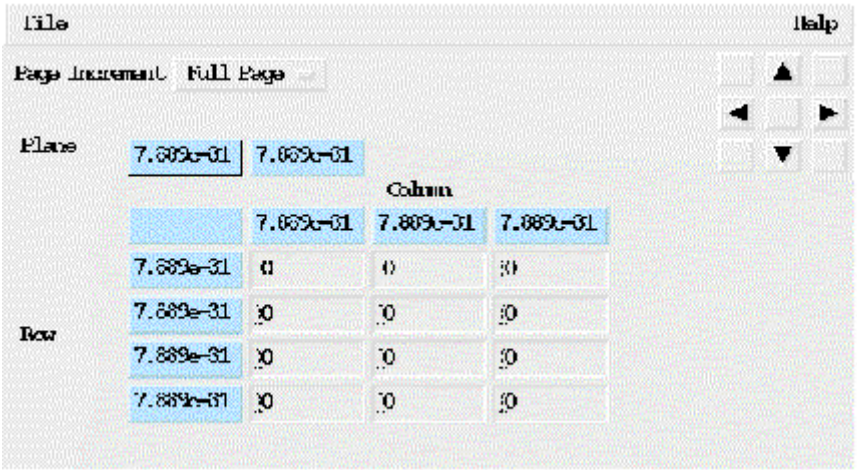


图 3-6 TABLE 类型数组的*VEDIT 对话框例子

3.11.5.4 使用*VREAD命令用数据文件填充数组

可以使用*VREAD 命令用数据文件填充数组(Utility Menu>Parameters>Array Parameters>Read from File)。该命令从一个 ASCII 数据文件读取信息，并由指定下标处开始写入数组中。可以通过数据描述符来控制从文件中读取的信息的格式。数据描述符必须括在圆括号中，并放在*VREAD 命令后。关于数据描述符的更多内容在 3.11.7.1 部分。数据描述符控制从每个记录中读取的数据数目，数据宽度和数据中小数点的位置。

例如，有下面的数据文件：

1.5　　7.8　12.3

15.6 -45.6 42.5

和一个维数为 3 x 2 的数组 EXAMPLE，执行下面的命令（宏或输入命令的一部分）：

*VREAD,EXAMPLE(1,1),,,2

(3F6.1)

结果为

$$\text{EXAMPLE} = \begin{bmatrix} 1.5 & 15.6 \\ 7.8 & -45.6 \\ 12.3 & 42.5 \end{bmatrix}$$

注意一不能直接在输入窗口中执行*VREAD 命令。但是，通过 Utility Menu>Parameters>Array Parameters>Read from File 弹出的对话框可以指定数据描述符并交互式执行该命令。

3.11.5.5 使用* TREAD命令用数据文件填充TABLE类型数组

设置完 TABLE 类型数组后，有两种选择来指定其元素的值：跟其他类型的数组一样，按照你的需要增加数值到表数组中，或者从一个外部文件的数据表中读取。

要从一个外部文件的数据表中读取数据，首先仍然要定义 TABLE 数组，指定行、列和面的数目及标识字，然后通过*TREAD 命令(Utility Menu>Parameters>Array Parameters>Read from File)读取包含数据表的 ASCII 文件。同时，还要指定在文件开头和数据表的第一行之间需跳过的行数(NSKIP)。

当从外部文件中读取数据时，要记住：

- 包含数据表的 ASCII 文件可以由文本编辑器或外部应用程序（如 Microsoft Excel）生成，但必须是 ASCII 形式，且由制表符分界；
- 首先必须在 ANSYS 中定义数组，记住允许下标值为(0,0)；
- 按行读入数值，直到数组中每行的所有列都已填充完; 然后，ANSYS 一行行地轮流填充它们包含的列。一定要保证你定义的数组有正确的维数。如果你在 ANSYS 中错误地定义了一个少于要求列数的数组，ANSYS 将用从数据表读入的第一行剩下的数据开始填充数组的下一行。类似地，如果你在 ANSYS 中错误地定义了一个多于要求列数的数组，ANSYS 将用从数据表另一行读入的数值填充数组的所有列 ，仅当换到下一行时才开始填充下一行。

可以从外部文件中读取数据来生成 1-D, 2-D, 和 3-D 表数组。下面的例子说明如何生成：

例 1: 1-D 表数组

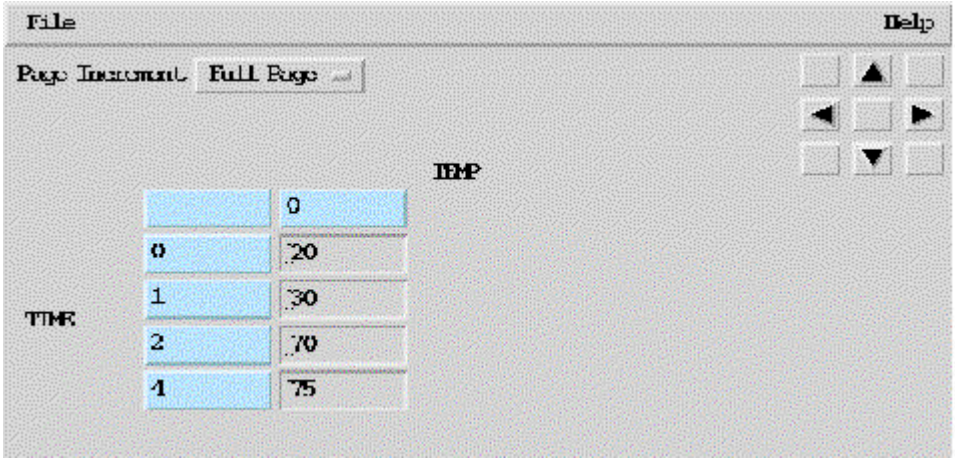
首先，用选择的应用程序（如电子制表软件，文本编辑器等）生成 1-D 表，然后把该文件保存为带制表符的文本文件。本例中，表名为"Tdata"，包含时间和温度的对应数据。该表以 ASCII 形式表示如下：

时间温度表

Time	Temp
0	20
1	30
2	70
4	75

在 ANSYS 中，用*DIM 命令 (Utility Menu>Parameters>Array Parameters>Define/Edit) 定义一个表数组参数"Tt"。指定其维数为 4 行 1 列，行标识字为 Time，列标识字为 Temp。注意，生成的数据表的数据为 4 行 1 列（第一列 -- TIME - 是行的下标值）。然后如上所述

读取该文件，指定跳过两行。该 TABLE 数组在 ANSYS 中将如下所示：



同样的例子，可通过下面的命令实现：

*DIM,Tt,table,4,1,1,TIME,TEMP

*TREAD,Tt,tdata,txt,,2

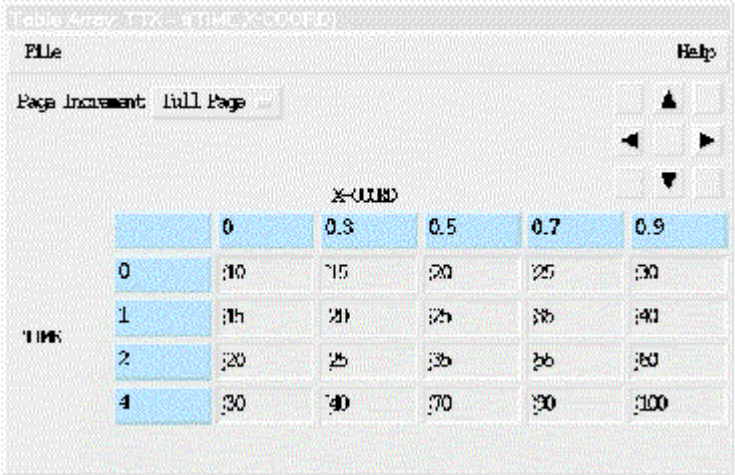
例 2: 2-D 表数组

对于这个例子，先生成（用电子制表软件，文本编辑器等）一个 2-D 表 "T2data"，其中包含作为时间函数的温度数据和 X 坐标值，然后把其读入一个名为"Ttx"的表数组参数中。该表以 ASCII 形式表示如下：

温度(时间-X 坐标)表

Time	X-Coordinate				
0	0	.3	.5	.7	.9
0	10	15	20	25	30
1	15	20	25	35	40
2	20	25	35	55	60
4	30	40	70	90	100

在 ANSYS 中，用*DIM 命令 (Utility Menu>Parameters>Array Parameters>Define/Edit) 定义一个表参数"Ttx"。指定其维数为 4 行 5 列，行标识字为 TIME，列标识字为 X-COORD。注意，生成的数据表的数据为 4 行 5 列，再加上行和列的下标值。然后如前所述读取该文件，指定跳过两行。该表数组在 ANSYS 中将如下所示：



同一个例子，可通过下面的命令实现：

*DIM,Ttx,table,4,5,,time,X-COORD

*TREAD,Ttx,t2data,txt,,2

例 3: 3-D 表数组

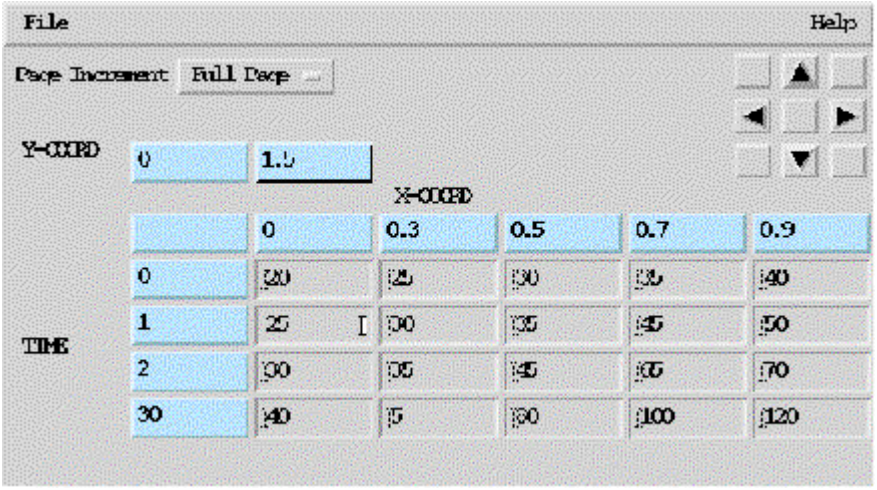
对于这个例子，先生成（用电子制表软件，文本编辑器等）一个 3-D 表 "T3data"，其中包含作为时间函数的温度数据、X 坐标值和 Y 坐标值，然后把其读入一个名为"Ttxy"的表数组参数中。该表以 ASCII 形式表示如下：

温度(时间-X 坐标)表

Time	X-Coordinate				
0	0	.3	.5	.7	.9
0	10	15	20	25	30
1	15	20	25	35	40
2	20	25	35	55	60
4	30	40	70	90	100
1.5	0	.3	.5	.7	.9
0	20	25	30	35	40
1	25	30	35	45	50
2	30	35	45	65	70
4	40	50	80	100	120

在上面的例子中，粗体字的值(在(0,0,Z)处)表示各个面。每面中，行列的下标值都是一样的，只是面的下标值及实际的数据值不同。上图中的阴影部分显示了面与面之间改变的实际的数据值。

在 ANSYS 中，通过*DIM 命令(Utility Menu>Parameters>Array Parameters>Define/Edit)定义一个表数组参数 "Ttxy"。在 3-D 表数组中，数组的维数由行、列和数据面的数目确定。第一列(TIME)是行的下标值，第一行是列的下标值。指定维数为 4 行，5 列，2 个面，行标识字为 TIME，列标识字为 X-COORD，面标识字为 Y-COORD。注意，生成的数据表的数据为 4 行 5 列 2 面，每面再加上行和列的下标值。然后如前所述读取该文件，指定跳过两行。对于第二个数据面(Y=1.5)，该 TABLE 数组在 ANSYS 中将如下所示：



同一个例子，可通过下面的命令实现：

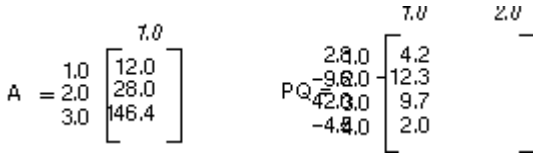
*DIM,Ttxy,table,4,5,2,TIME,X-COORD,Y-COORD

*TREAD,Ttxy,t3data,txt,,2

3.11.6 插入值

访问表数组时，ANSYS 可在已定义的值之间进行插值。

下面的例子说明了 ANSYS 如何在 TABLE 数组中进行插值：



给定 A 为 TABLE 数组参数，ANSYS 程序能计算 A(1)和 A(2)之间的任意值，如：

A(1.5) 等于 20.0 (12.0 和 28.0 的中值)

A(1.75) 等于 24.0

A(1.9) 等于 26.4

同样，如果 PQ 是一个 TABLE 数组参数

PQ(1.5,1) 等于-3.4 (2.8 和 -9.6 的中值)

PQ(1,1.5) 等于 3.5 (2.8 和 4.2 的中值)

PQ(3.5,1.3) 等于 14.88

可以利用该特性使用 TABLE 数组参数来描述函数 y=f(x)：用 j=0 列作为自变量 x 的值，j=1 列作为 y 的值。例如，对于一个如下所示由 5 个点描述的时间相关力函数：

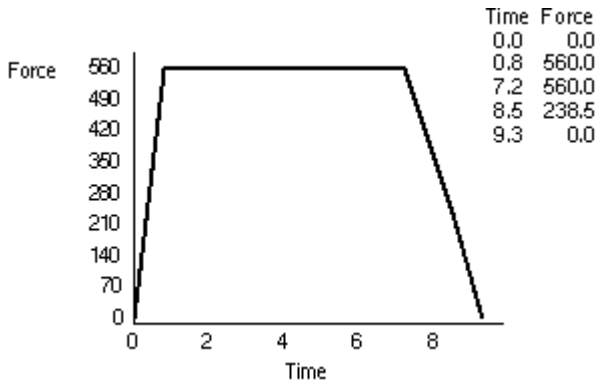


图 3-7 时间相关力函数

指定函数为一个 TABLE 数组参数，其数组元素为力的值，从 1 到 5 的行下标值是时间值 0.0 到 9.3。该参数可表示如下：

$$\text{FORCE} = \begin{array}{c} \\ 0.0 \\ 0.8 \\ 7.2 \\ 8.5 \\ 9.3 \end{array} \begin{array}{c} 1.0 \\ \left[\begin{array}{r} 0.0 \\ 560.0 \\ 560.0 \\ 238.5 \\ 0.0 \end{array}\right] \end{array}$$

ANSYS 能计算出（通过线性插值）在 FORCE 参数中没有定义的时间处的力值。在上面的例子中，ANSYS 可以计算出 FORCE(9)的值为 89.4375。如果参数位置超过了数组的维数，那么该参数的值为最后的参数值，并不采用外推法。例如，ANSYS 将给 FORCE(5,2)赋值 560.0，给 FORCE(12)赋值 0.0。

从这些例子中可以看到 TABLE 数组参数在分析中是非常有用的。其典型应用有时间历程载荷函数、响应谱曲线、压力曲线、材料-温度曲线、磁性材料的 B-H 曲线等等。运行时，TABLE 数组参数比 ARRAY 类型参数需要更多的机时。

3.11.6.1 把获取值存入数组参数或恢复数组参数值

可以通过*VGET 命令 (Utility Menu>Parameters>Get Array Data)（与*GET 命令类似）获取 ANSYS 提供的值，并把它们储存在一个数组中。

必须为*VGET 命令生成的数组参数确定起始位置。当 KLOOP 为缺省值时，循环将按顺序处理实体元。例如，*VGET,A(1),ELEM,5,CENT,X 返回单元 5 的质心的 x 坐标值，并储存在数组 A 的第一个值中，然后继续获取单元 6，7…，直到填满数组。在这个例子中，如果 KLOOP 为 4，那么就会返回质心的 x、y 和 z 坐标值。

可以通过*VPUT 命令 (Utility Menu>Parameters>Array Operations>Put Array Data)来恢复数组参数的值。

*VPUT 命令使用和上面描述的*VGET 命令相同的参数，但是作用与之相反。在 ANSYS 命令参考手册（ANSYS Commands Reference）的命令描述部分讨论了*VPUT 命令。

ANSYS 程序直接输出向量，不进行坐标系统的转换。*VPUT 能替换已经存在的数组内容，但不能产生新的内容。在数据库中被改变的自由度结果可被所有后续的操作利用。其他临时更改的结果主要用于即时打印和显示。

注意－使用该命令时要非常小心，因为它能更改整个数据库部分。*VPUT 命令不支持*VGET 列出的所有内容，因为把值放置在某些位置会使 ANSYS 数据库发生矛盾。

3.11.6.2 列出数组参数

和标量参数一样，可用*STATUS 命令来列出数组参数。下面的例子说明了*STATUS 命令的用法：

```
*STATUS
  ABBREVIATION STATUS-

    ABBREV       STRING
    SAVE_DB      SAVE
    RESUM_DB     RESUME
    QUIT         Fnc_/EXIT
    POWRGRPH     Fnc_/GRAPHICS
    ANSYSWEB     Fnc_HomePage

  PARAMETER STATUS-              (     5 PARAMETERS DEFINED)
                        (INCLUDING      2 INTERNAL PARAMETERS)

    NAME            VALUE            TYPE       DIMENSIONS
    MYCHAR          hi            CHARACTER
    MYPAR                            ARRAY        4        6        1
    MYPAR1       .987350000        SCALAR

*STATUS,XYZ(1),5,9          ！列出 XYZ 中 5 行到 9 行的参数
  PARAMETER STATUS- XYZ          (     4 PARAMETERS DEFINED)
        LOCATION              VALUE
        5       1       1     -8.98000000
        6       1       1      9.01000000
        7       1       1     -30.6000000
        8       1       1      51.0000000
        9       1       1     -51.9000000
*STATUS,FORCE(1),,,0       ！列出参数 FORCE，包括 j=0 列
PARAMETER STATUS- FORCE        (     4 PARAMETERS DEFINED)
        LOCATION              VALUE
        1       0       1     0.000000000E+00
        2       0       1     0.800000000
        3       0       1      7.20000000
        4       0       1      8.50000000
        5       0       1      9.30000000
        1       1       1     0.000000000E+00
        2       1       1      560.000000
        3       1       1      560.000000
        4       1       1      238.500000
        5       1       1     0.000000000E+00
```

```
*STATUS,T2(1,1)              ! 列出参数 T2
PARAMETER STATUS- T2            (     4 PARAMETERS DEFINED)
        LOCATION              VALUE
        1       1       1     0.600000000
        2       1       1      2.00000000
        3       1       1     -1.80000000
        4       1       1      4.00000000
        1       2       1      7.00000000
        2       2       1      5.00000000
        3       2       1      9.10000000
        4       2       1      62.5000000
        1       3       1     2.000000000E-04
        2       3       1     -3.50000000
        3       3       1      22.0000000
        4       3       1     1.000000000E-02
*STATUS,RESULT(1)       !列出参数 RESULT

PARAMETER STATUS- RESULT        ( 4 PARAMETERS DEFINED)
    LOCATION            VALUE
    1       1       1       SX      (CHAR)
    2       1       1       SY      (CHAR)
    3       1       1       SZ      (CHAR)
```

3.11.7 写数据文件

可以通过*VWRITE 命令把数组中的数据写到格式化（表格式）的数据文件中。该命令最多可带有 10 个数组向量作为参数，并把这些向量中包含的数据写入当前打开的文件(*CFOPEN 命令)中。每个向量的格式由*VWRITE 命令后紧接的 FORTRAN 77 数据描述符确定（因此不能从 ANSYS 输入窗口执行该命令）。数组向量要指定起始元素位置(如 MYARRAY(1,2,1))。可以用表达式来计算数据文件中每一行的位置。关键词 SEQU 将从 1 开始写一个连续的整数列。

数据文件中每一行的格式由数据描述符决定。命令的每一个参数都要带有一个描述符。在描述符行不要有 FORMAT 这个词。可以使用实数格式或字符格式描述符，不能使用整形或直接列表描述符。

3.11.7.1 数据格式描述符

若对 FORTRAN 的数据描述符不熟悉，本部分将从头说明如何格式化数据文件。要获得更多的信息，请参阅 FORTRAN 77 编译器文件。

一定要给*VWRITE 命令所带参数中的每一个数据项提供一个数据描述符。

通常地，对任何数字值都可以采用 F 描述符（浮点数）。F 描述符的语法为：

Fw.d

其中，w 指数据宽度，d 指小数点右边的阿拉伯数字的个数。

因此，对于一个共有 10 个阿拉伯数字且小数点后有 8 个阿拉伯数字的数据，将采用如下描述符：

F10.8

对于字符数据，可以采用 A 描述符。A 描述符的语法为：

Aw

其中，w 指数据宽度。

因此，因此，对于一个共有 8 个字符的字符数据，将采用如下描述符：

A8

下面的例子说明如何使用*VWRITE 命令和数据描述符：

给定数组 MYDATA 为

$$\text{MYDATA} = \begin{bmatrix} 2.15215183 & 3.89075020 & 5.28636971 & 7.15706483 & 13.7859423 & 87.4970443 \\ 2.30485343 & 4.04486730 & 5.40919563 & 7.68192625 & 15.5483820 & 86.5677915 \\ 2.01051819 & 3.39152436 & 5.93663807 & 7.38584253 & 18.4635868 & 45.7263566 \\ 2.36833012 & 3.32711472 & 5.63220341 & 7.22482004 & 18.7977889 & 39.7902425 \\ 2.84819512 & 4.76350638 & 5.97802354 & 7.29258882 & 14.8096356 & 62.0843906 \\ 2.22795343 & 3.48214546 & 5.54685145 & 7.90325139 & 14.0708891 & 37.6009897 \end{bmatrix}$$

下面的宏首先定义一个标量参数 X 为 25，然后打开文件 vector (*CFOPEN 命令)。然后使用*VWRITE 命令定义将要写入文件中的数据，写入的第一个向量采用 SEQU 关键词来提供行数。注意，在某些情况下，常数、标量参数和包含数组元素值的操作也可以写入文件中。

```
x=25
*cfopen,vector
*vwrite,SEQU,mydata(1,1,1),mydata(1,2,1),mydata(1,3,1),10.2,x,mydata(1,1,1)+3
(F3.0,'  ',F8.4,'  ',F8.1,'  'F8.6,'  ',F4.1,'  'F4.0,'  'F8.1)
*cfclos
```

该宏将生成如下的数据文件：

```
  1.    2.1522      3.9   5.286370   10.2    25.       5.2
  2.    2.3049      4.0   5.409196   10.2    25.       5.2
  3.    2.0105      3.4   5.936638   10.2    25.       5.2
  4.    2.3683      3.3   5.632203   10.2    25.       5.2
  5.    2.8491      4.8   5.978024   10.2    25.       5.2
  6.    2.2280      3.5   5.546851   10.2    25.       5.2
```

下面的例子使用如下定义的数组

$$\text{MYDATA} = \begin{bmatrix} 10 & 50 \\ 20 & 70 \\ 30 & 80 \end{bmatrix}$$

注意接下来的*VWRITE 命令中数据描述符的用法：

```
*vwrite,SEQU,mydata(1,1),mydata(1,2),(mydata1(1,1)+mydata1(1,2))
(' Row',F3.0,' contains ',2F7.3,'. Is their sum ',F7.3,' ?')
```

结果数据文件为：

```
Row 1. contains   10.000 50.000.    Is their sum   60.000 ?
Row 2. contains   20.000 60.000.    Is their sum   60.000 ?
Row 3. contains   30.000 70.000.    Is their sum   60.000 ?
```

3.11.8 对数组参数的运算

同参数表达式和函数允许对标量参数进行运算一样，也有一系列的命令可以对数组参数进行运算。这种运算可分为以下几类：对列（向量）的运算，矢量运算和完整矩阵(数组)的运算,矩阵运算。所有的运算都受到一套 ANSYS 规定命令的影响，这一部分详见 3.11.8.3。

3.11.8.1 对向量的运算

对向量的运算就是按某种顺序对数组元素进行一系列的诸如加、减、求正弦、求余弦、点积、叉乘等运算。虽然可以通过采用 Do 循环（见 4.5.5 部分）来达到这一目的，但是更为简便快捷的方法是采用向量操作命令-*VOPER, *VFUN, *VSCFUN, *VITRP, *VFILL, *VREAD, 和 *VGET。在这些命令中，只有*VREAD 和*VWRITE 对字符数组参数有效，其余的只能用于 ARRAY 类型或 TABLE 类型（由*DIM 定义）的数组参数。

*VFILL, *VREAD, *VGET, *VWRITE, 和 *DIM 命令在本章中已经介绍过了，本部分要讨论的其它命令包括：

***VOPER** 或 Utility Menu>Parameters>Array Operations>Vector Operations

对两个输入数组向量进行运算，输出一个数组向量。

***VFUN** 或 Utility Menu>Parameters>Array Operations>Vector Functions

对两个输入数组向量执行某函数，输出一个数组向量。

***VSCFUN** 或 Utility Menu>Parameters>Array Operations>Vector-Scalar Func

确定单个输入数组矢量属性，并将结果存放到指定的标量参数中。

***VITRP** 或 Utility Menu>Parameters>Array Operations>Vector Interpolate

通过在指定的表下标位置插入一个数组参数(TABLE 类型)来生成一个数组参数(ARRAY 类型)。

下面的例子说明了这些命令的用法。查阅 ANSYS 命令参考手册（ANSYS Commands Reference）可以得到关于这些命令的语法。在下面的例子中，数组参数(ARRAY 类型) X, Y, 和 THET 定义为：

$$X = \begin{bmatrix} -2 & 6 & 8 & 0 \\ 1 & 0 & 2 & 12 \\ 4 & -3 & -1 & 7 \\ -8 & 1 & 10 & -5 \end{bmatrix} \qquad Y = \begin{bmatrix} 3 & 2 & 5 & -6 \\ -5 & -7 & 1 & 0 \\ 8 & 0 & 0 & 11 \\ 1 & 4 & 9 & 16 \end{bmatrix}$$

$$THETA = \begin{bmatrix} 0 \\ 15 \\ 30 \\ 45 \\ 60 \\ 75 \\ 90 \end{bmatrix}$$

首先定义结果数组为 Z1。然后，*VOPER 命令把 X 的第 2 列和 Y 的第 1 列相加，二者都从第一行开始，然后把结果赋给 Z1。注意，对所有的数组参数都要指定起始位置（行和列的下标数）。

```
*DIM,Z1,ARRAY,4
*VOPER,Z1(1),X(1,2),ADD,Y(1,1)
```

$$Z1 = \begin{bmatrix} 9 \\ -5 \\ 5 \\ 2 \end{bmatrix}$$

在下一个例子中，同样首先定义结果数组为 Z2。然后，*VOPER 命令把 X 的第 1 列（从行 2 开始）和 Y 的第 1 列（从行 1 开始）相乘，然后把结果赋给 Z2（从行 1 开始）。

*DIM,Z2,ARRAY,3

*VOPER,Z2(1),X(2,1),MULT,Y(1,4)

$$Z2 = \begin{bmatrix} -6 \\ 0 \\ -88 \end{bmatrix}$$

在这个例子中，同样首先定义结果数组为 Z4。然后，*VOPER 命令计算四对向量的点积，一对为 X 和 Y 的 1 行。这些矢量的 i, j, 和 k 分量依次是 X 的列 1, 2,和 3 以及 Y 的列 2, 3,和 4。结果将写进 Z4，它的 i, j 和 k 分量分别是矢量 1, 2,和 3。

*DIM,Z4,ARRAY,4,3

*VOPER,Z4(1,1),X(1,1),CROSS,Y(1,2)

$$Z4 = \begin{bmatrix} -76 & 4 & -22 \\ -2 & -14 & 1 \\ -33 & -44 & 0 \\ -74 & 168 & -76 \end{bmatrix}$$

在下一个例子中，同样首先定义结果数组为 A3。然后，*VFUN 命令把 X 的第 2 列中的每个元素平方后赋给 A3。

*DIM,A3,ARRAY,4

*VFUN,A3(1),PWR,X(1,2),2

$$A3 = \begin{bmatrix} 36 \\ 0 \\ 9 \\ 1 \end{bmatrix}$$

在下一个例子中，同样首先定义结果数组为 A4。然后，两个*VFUN 命令分别计算 THETA 中的数组元素的余弦和正弦值，并分别赋给 A4 中的第一和第二列。注意，现在的 A4 表示一个由 7 个点（其 x, y, z 的全局坐标就是那三个向量）描述的 90 度的圆弧。该圆弧半径为 1.0，并在 z = 2.0 且与 x-y 平行的平面上。

*DIM,A4,ARRAY,7,3

*AFUN,DEG

*VFUN,A4(1,1),COS,THETA(1)

*VFUN,A4(1,2),SIN,THETA(1)

A4(1,3)=2,2,2,2,2,2,2

$$A4 = \begin{bmatrix} 1.0 & 0.0 & 2.0 \\ 0.966 & 0.259 & 2.0 \\ 0.866 & 0.5 & 2.0 \\ 0.707 & 0.707 & 2.0 \\ 0.5 & 0.866 & 2.0 \\ 0.259 & 0.966 & 2.0 \\ 0.0 & 1.0 & 2.0 \end{bmatrix}$$

在下一个例子中，同样首先定义结果数组为 A5。然后，*VFUN 命令计算 A4 所表示的

曲线在每个点处的切线向量，并进行归一化处理后赋给 A5。

*DIM,A5,ARRAY,7,3

*VFUN,A5(1,1),TANG,A4(1,1)

$$A5 = \begin{bmatrix} -0.131 & 0.991 & 0 \\ -0.259 & 0.965 & 0 \\ -0.5 & 0.866 & 0 \\ -0.707 & 0.707 & 0 \\ -0.866 & 0.5 & 0 \\ -0.966 & 0.259 & 0 \\ -0.991 & 0.131 & 0 \end{bmatrix}$$

两个附加的*VOPER 运算-积聚(GATH)和分散(SCAT)-可用于根据在一个“位置”向量中包含的位置号从一个向量中拷贝值到另一个向量。下面的例子说明了积聚运算。注意，结果数组总是预先定义。在这个例子中，积聚运算拷贝 B1 的值到 B3（通过在 B2 中指定的下标位置）。注意，B3 中的最后一个元素为 0，是其初始化值。

*DIM,B1,,4

*DIM,B2,,3

*DIM,B3,,4

B1(1)=10,20,30,40

B2(1)=2,4,1

*VOPER,B3(1),B1(1),GATH,B2(1)

$$B3 = \begin{bmatrix} 20 \\ 40 \\ 10 \\ 0 \end{bmatrix}$$

3.11.8.2 矩阵运算

矩阵运算是一种数字数组参数之间的数学运算，例如矩阵乘法、计算转置矩阵、求解联立方程组等。

本部分将要讨论的命令包括：

*MOPER 或 Utility Menu>Parameters>Array Operations>Matrix Operations

对两个输入数组参数矩阵进行矩阵运算，输出一个数组参数矩阵。矩阵运算包括：

- 矩阵相乘；
- 求解联立方程组；
- 对矩阵中的某个指定向量排序（按上升顺序）；
- 计算两个向量之间的协方差；
- 计算两个向量之间的相关性；

***MFUN** 或 Utility Menu>Parameters>Array Operations>Matrix Functions

拷贝或转置一个数组参数矩阵（接受一个输入矩阵，生成一个输出矩阵）。

***MFOURI** 或 Utility Menu>Parameters>Array Operations>Matrix Fourier

计算傅立叶级数的系数或求傅立叶级数。

下面的例子用来说明这些命令的用法。查阅 ANSYS 命令参考手册（ANSYS Commands Reference）可以得到关于这些命令的语法。

这个例子说明了*MOPER 命令的排序功能。假定数组(SORTDATA)被定义如下：

$$\text{SORTDATA} = \begin{bmatrix} 3 & 10 & 11 \\ 5 & -4 & 12 \\ 8 & -9 & 13 \\ 2 & 7 & 14 \\ 6 & 1 & 15 \end{bmatrix}$$

首先，定义数组 OLDORDER，*MOPER 命令将把行的初始顺序放在 OLDORDER 中。然后，*MOPER 命令对 SORTDATA 中的行进行排序，这样 1,1 向量就按升序排列了。

*dim,oldorder,,5

*moper,oldorder(1),sortdata(1,1),sort,sortdata(1,1)

得到的结果数组为：

$$\text{SORTDATA} = \begin{bmatrix} 2 & 7 & 14 \\ 3 & 10 & 11 \\ 5 & -4 & 12 \\ 6 & 1 & 15 \\ 8 & -9 & 13 \end{bmatrix} \qquad \text{OLDORDER} = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 5 \\ 3 \end{bmatrix}$$

若要恢复 SORTDATA 数组为初始顺序，那么就要执行下面的命令：

*moper,oldorder(1),sortdata(1,1),sort,oldorder(1,1)

在下面的例子中，*MOPER 命令用来求解联立方程组。两个数组定义如下：

$$A = \begin{bmatrix} 2 & 4 & 3 & 2 \\ 3 & 6 & 5 & 2 \\ 2 & 5 & 2 & -3 \\ 4 & 5 & 14 & 14 \end{bmatrix} \qquad B = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 11 \end{bmatrix}$$

*MOPER 命令可以用于求解联立方程组组成的方阵，方程组采用如下形式：

$$a_{k1}X_1 + a_{k2}X_2 + \ldots + a_{kk}X_k = b$$

*MOPER 命令求解的联立方程组为：

$$2X_1 + 4X_2 + 3X_3 + 2X_4 = 2$$
$$3X_1 + 6X_2 + 5X_3 + 2X_4 = 2$$
$$2X_1 + 5X_2 + 2X_3 - 3X_4 = 2$$
$$4X_1 + 5X_2 + 14X_3 + 14X_4 = 11$$

要求解以上方程组，首先定义结果数组 C。然后，用*MOPER 命令求解方程组，用 A 作为系数矩阵，B 作为 b 值组成的向量。

*DIM,C,,4

*MOPER,C(1),A(1,1),SOLV,B(1)

C 数组中的结果为：

$$C = \begin{bmatrix} -66 \\ 26 \\ 6 \\ 4 \end{bmatrix}$$

接下来的例子说明如何使用*MFUN 命令来转置数组中的数据。在这个例子中，假定数组 DATA 预先定义为：

$$\text{DATA} = \begin{bmatrix} 34 & 25 \\ 22 & 68 \\ -7 & 12 \end{bmatrix}$$

同样地，首先定义结果数组 DATATRAN。然后，用*MFUN 命令转置数据并把结果写入 DATATRAN 数组。

*DIM,DATATRAN,,2,3

*MFUN,DATATRAN(1,1),TRAN,DATA(1,1)

DATATRAN 数组中的结果为：

$$\text{DATATRAN} = \begin{bmatrix} 34 & 22 & -7 \\ 25 & 68 & 12 \end{bmatrix}$$

3.11.8.3 用于向量和矩阵运算的命令

所有向量和矩阵运算命令都受到下面这些命令的影响：*VCUM, *VABS, *VFACT, *VLEN, *VCOL, 和*VMASK. (在所有命令中，只有*VLEN 和 *VMASK，与*VREAD 或 *VWRITE 连用时，对字符数组参数有效)。可以通过*VSTAT 命令来查看这些命令的状态。这些命令中的绝大多数（及其相应的 GUI 路径）在本章中已经介绍过了，接下来将介绍剩下的命令。

除*VSTAT 命令之外，下面将要介绍的其它所有的命令都可以通过菜单 Utility Menu>Parameters>Array Operations>Operation Settings 得到。

注意－每次执行向量或矩阵运算之后，所有规定命令都将重置为缺省设置。

下面列出有效的数组规定命令：

***VCUM**

指定结果累积或不累积（覆盖已有结果）。ParR 为向量运算的结果，要么被加入一个已存在的同名参数中，要么被覆盖。缺省为不累积结果，即 ParR 覆盖掉已存在的同名参数。

***VABS**

使向量运算中的某个或所有参数取绝对值。缺省为实数值。

***VFACT**

对向量运算中的某个或所有参数乘以一个比例因子。比例因子缺省为 1.0。

***VCOL**

指定矩阵运算中列的数目。缺省为从指定起始处填满结果数组的所有位置。

***VSTAT**

列出数组参数的当前状态。

***VLEN** 或 Utility Menu>Parameters>Array Operations>Operation Settings

指定数组参数运算中被用到的行的数目。

***VMASK** 或 Utility Menu>Parameters>Array Operations>Operation Settings

指定某个数组作为屏蔽向量。

下表列出了各类规定命令及其影响到的向量和矩阵运算命令：

	***VABS**	***VFACT**	***VCUM**	***VCOL**	***VLEN** NROW,NINC		***VMASK**
***MFOURI**	No	No	No	N/A	No	No	No
***MFUN**	Yes	Yes	Yes	No	Yes	No	Yes
***MOPER**	Yes	Yes	Yes	No	Yes	No	Yes
***VFILL**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VFUN**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VGET**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VITRP**	Yes	Yes	Yes	N/A	Yes	Yes	Yes

***VOPER**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VPLOT**	No	No	N/A	N/A	Yes	Yes	Yes
***VPUT**	Yes	Yes	No	N/A	Yes	Yes	Yes
***VREAD**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VSCFUN**	Yes	Yes	Yes	N/A	Yes	Yes	Yes
***VWRITE**	No	No	N/A	N/A	Yes	Yes	Yes

下面的例子说明了一些规定命令的用法。查阅 ANSYS 命令参考手册（ANSYS Commands Reference）可以得到关于这些命令的语法。

例 1 定义结果数组 CMPR。然后与*VMASK 和 *VLEN 命令连用的两个*VFUN 命令压缩选择的数据并把结果写入 CMPR 中的指定位置。在*VFUN 命令中， COMP 运算的反运算为 EXPA 。

```
*DIM,CMPR,ARRAY,4,4
*VLEN,4,2        ！每四行执行下一个*V---- 运算, 每次跳过一行
*VFUN,CMPR(1,2),COMP,Y(1,1)
*VMASK,X(1,3)        ！使用 X 的列 3 作为下一个*V----运算的屏蔽矢量。
*VFUN,CMPR(1,3),COMP,Y(1,2)
```

$$\text{CMPR} = \begin{bmatrix} 0 & 3 & 2 & 0 \\ 0 & 8 & -7 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

例 2 用*VFACT 命令把数组向量中的值根据 NUMDP 标量参数（本例中设为 2）指定的值按比例增大或缩小。NUMDATA 数组定义如下：

$$\text{NUMDATA} = \begin{bmatrix} 2.526 \\ 2.524 \\ -6.526 \\ -6.524 \end{bmatrix}$$

```
numdp=2
*vfact,10**numdp
*vfun,numdata(1),copy,numdata(1)
*vfun,numdata(1),nint,numdata(1)
*vfact,10**(-numdp)
*vfun,numdata(1),copy,numdata(1)
```

或者更简单地执行如下命令：

```
numdp=2
*vfact,10**numdp
*vfun,numdata(1),copy,numdata(1)
*vfact,10**(-numdp)
*vfun,numdata(1),nint,numdata(1)
```

那么，作为结果的 NUMDATA 数组为

$$\text{NUMDATA} = \begin{bmatrix} 2.53 \\ 2.52 \\ -6.53 \\ -6.52 \end{bmatrix}$$

例3 通过*VLEN 和 *VMASK 命令找到小于 100 的素数的数目。生成数组 MASKVECT 时，用 1.0 表示该行值是素数，用 0.0 表示该行值不是素数。生成屏蔽向量的算法为：把所有值大于 1 的行初始化为 1.0，然后通过成倍增加因数在可能的因数范围内进行循环。*VLEN

命令设置运算的行增量为 FACTOR，执行*VFILL 命令时，行号根据该值增加。因为起始行是 FACTOR x 2，所以每次循环中行的变化为：FACTOR x 2, FACTOR x 3, FACTOR x 4 等等。

```
*dim,maskvect,,100
*vfill,maskvect(2),ramp,1
*do,factor,2,10,1
*vlen,,factor
*vfill,maskvect(factor*2),ramp,0
*enddo
*vmask,maskvect(1)
*dim,numbers,,100
*vfill,numbers(1),ramp,1,1
*status,numbers(1),1,10
```

输出结果可以由*STATUS 命令显示出来，NUMBERS 中的前 10 个元素为：

```
PARAMETER STATUS- NUMBERS     (     5 PARAMETERS DEFINED)
                        (INCLUDING        2 INTERNAL PARAMETERS)

LOCATION              VALUE
1      1      1    0.000000000E+00
2      1      1     2.00000000
3      1      1     3.00000000
4      1      1    0.000000000E+00
5      1      1     5.00000000
6      1      1    0.000000000E+00
7      1      1     7.00000000
8      1      1    0.000000000E+00
9      1      1    0.000000000E+00
10     1      1    0.000000000E+00
```

3.11.9 用图形表示数组参数向量

可以通过*VPLOT 命令用图形来显示数组向量的值。

接下来用例子说明*VPLOT 命令的一些功能。在本例中，有两个 TABLE 类型数组(TABLEVAL 和 TABLE2)和一个数字数组，分别定义如下：

$$\text{TABLEVAL} = \begin{array}{c|ccc} & 0 & 3 & 9 \\ \hline 4 & 6 & 12 \\ 7 & 8 & 6 \\ 15 & 10 & 3 \end{array} \qquad \text{TABLE2} = \begin{array}{c|c} & 0 \quad 40 \\ \hline 19 & 70 \\ 88 & 80 \\ 99 & 95 \end{array}$$

$$\text{ARRAYVAL} = \begin{bmatrix} 6 & 12 \\ 8 & 6 \\ 10 & 3 \end{bmatrix}$$

注意，既然 ARRAY 类型数组的数据是无序的，那么就用柱状图表示；TABLE 类型数组的数据是有序的，就用曲线表示。

绘图通过下面的命令得到：

*vplot,,arrayval(1,1),2

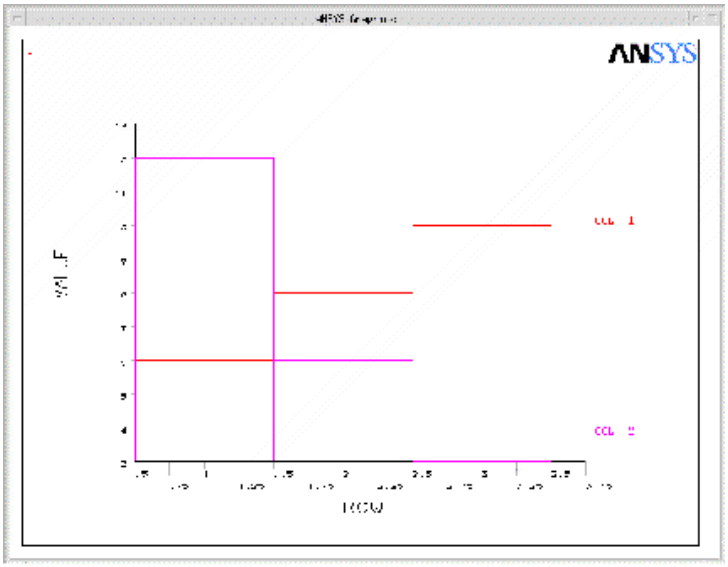


图 3-7 图例

*vplot,,tableval(1,1),2

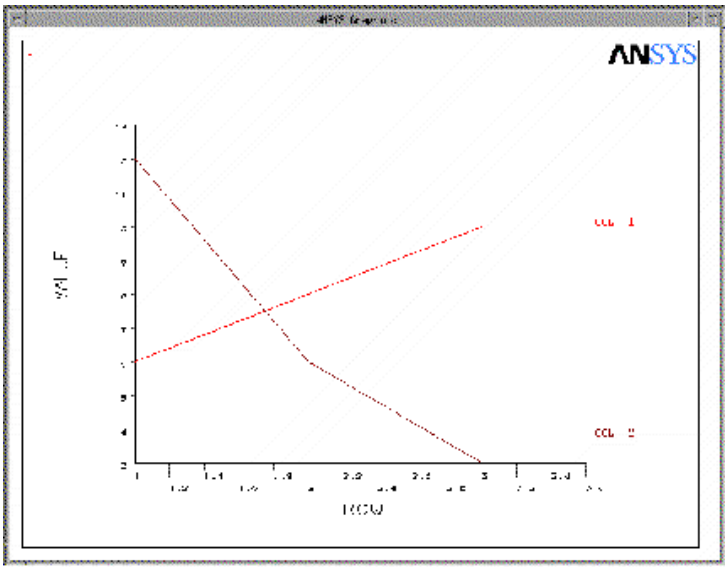


图 3-8 图例

*vplot,table2(1),tableval(1,1),2

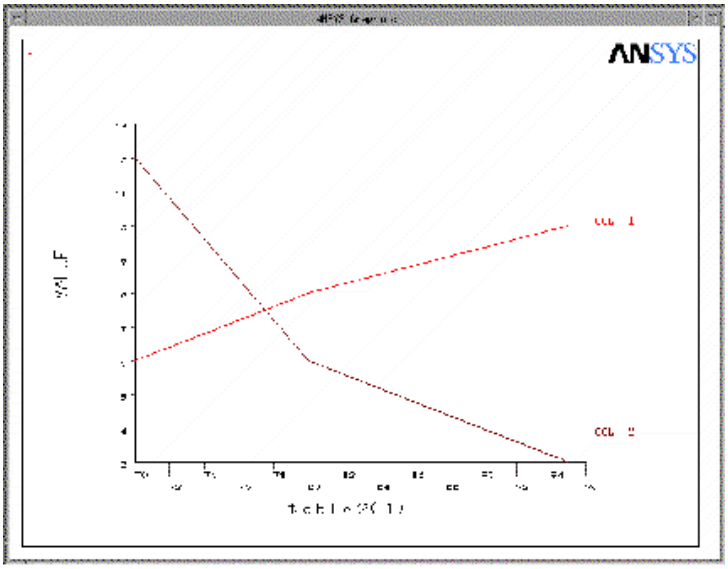


图 3-9 图例

*vplot,tableval(1,0),tableval(1,1),2

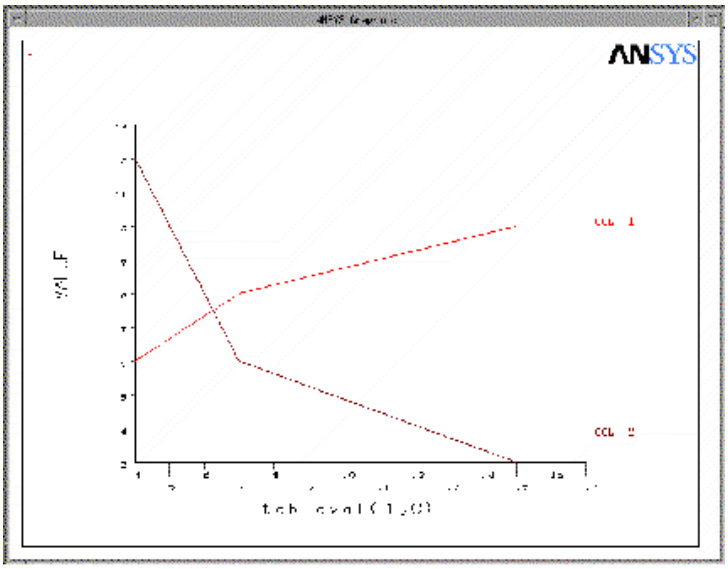


图 3-10 图例

第四章 作为宏语言的APDL

4.1 什么是APDL宏

可以在一个宏文件（有时候也被称为命令文件）中记录一个经常用到的 ANSYS 命令序列。通过宏可以生成自定义的 ANSYS 命令。例如，在磁分析中计算由于涡流引起的功率损失时，需要在后处理中执行一系列的 ANSYS 命令。通过把这些命令记录到一个宏中，那么就有了一个新的命令，通过执行它可以完成该计算所需的那一系列的 ANSYS 命令。除了执行一系列的 ANSYS 命令之外，宏还可以调用 GUI 函数或把值传递给参数。

还可以对宏进行嵌套。也就是说，一个宏能调用第二个宏，第二个宏能调用第三个宏，等等。最多可嵌套 20 层，其中包括由 ANSYS /INPUT 命令引起的任何文件转换。每一次嵌套的宏执行完毕后，ANSYS 程序仍置于前一个宏的控制之下。

下面是一个简单的宏文件的例子。在本例中，宏生成一个尺寸为 4, 3, 2 的长方形块和一个半径为 1 的球体。然后，从块的一个角处减去球体。

```
/prep7
/view,,-1,-2,-3
block,,4,,3,,2
sphere,1
vsbv,1,2
finish
```

假如这个宏取名为 mymacro.mac，那么就可以用这样一个 ANSYS 命令来执行以上的命令序列：*use,mymacro 或 (因为扩展名为.mac) mymacro。

很显然，这个宏的功能并不十分强大，但它很好地说明了宏的原理。

本章提供了一些关于产生、保存和执行宏的知识。同时论述了在产生宏的过程中，必须用到的作为解释性语言的 APDL 的有关基本知识。

4.2 产生宏

可以在 ANSYS 中产生宏，也可以通过文本编辑器(如 emacs, vi, 或 wordpad) 产生宏。假如要生成的宏很简单，那么在 ANSYS 中生成就非常方便了。假如要生成一个长的复杂的宏，或者要编辑一个已经存在的宏，那么最好使用文本编辑器。而且，可以使用文本编辑器利用一个已经存在的类似的宏或 ANSYS 日志文件来生成自己的宏。

如果生成的宏既长且复杂，那么最好从一个已经存在的类似的宏着手，或者先通过交互模式完成该任务，再以生成的日志文件作为基础生成宏。这两种方法都可以大大地缩短所需耗费的时间和精力。

4.2.1 宏文件命名规则

宏就是保存在一个文件中的 ANSYS 命令序列。宏不能与已经存在的 ANSYS 命令同名，否则，ANSYS 执行的将是内部的命令，而不是宏。下面是宏命名中所受到的限制：

- 文件名不能超过 32 个字符；
- 文件名不能以数字开头；
- 文件扩展名不能超过 8 个字符（如果想和执行 ANSYS 命令一样执行宏，该宏的扩展名应为.mac）；
- 文件名或文件扩展名中不能包含空格；
- 文件名或文件扩展名不能包含任何被当前文件系统禁止使用的字符，为了更好的移置性，还不能包含任何被 UNIX 或 Windows 文件系统禁止使用的字符。

为了确保没有使用 ANSYS 命令名，在生成宏之前应该试着象运行 ANSYS 命令一样运行准备赋给宏的名称。如果 ANSYS 返回如下所示的消息，就可以确信在当前处理器中没有该命令。为“安全”起见，应该在每一个计划要用到的处理器中都检查一下宏文件的名称。（也可以检查宏文件名是否与在线文档中的某个命令名相同，但是该方法不能查找不在文档中的命令）

图 4-1 声明执行未知命令的消息框

若使用.mac 作为扩展名，ANSYS 将和执行内部命令一样执行该宏。扩展名.MAC 用于 ANSYS 内部的宏，用户不能使用。

4.2.2 宏的搜索路径

一般地，ANSYS 按如下缺省路径搜索用户生成的宏(.mac 扩展名)：

1. 由环境变量 ANSYS_MACROLIB(如果已经定义了)指定的路径或注册路径(主目录)。该环境变量在针对各平台的 ANSYS 安装和配置指南（ANSYS installation and configuration guide）中有详细说明；
2. 由/PSEARCH 命令（如果已经定义了）指定的路径。该路径在注册路径之前，在由环境变量 ANSYS_MACROLIB 指定的路径之后被搜索；
3. 当前路径。

可以把仅供自己使用的宏放在自己的主目录下。共享的宏应放在 ANSYS 文件目录或一些可以通过环境变量 ANSYS_MACROLIB 访问的目录下：

- 对于 Windows 95 用户：必须通过环境变量指定主目录和驱动器。参看针对 Windows 的 ANSYS 安装和配置指南（ANSYS Installation and Configuration Guide for Windows）。
- 对于 Windows NT 用户：当前路径为管理员设置的缺省路径(通常为一个网络资源)，可以从管理员处得到该路径。也可以通过环境变量生成一个本地的“主目录”。该本地主目录在由域初始化文件定义的缺省路径之后被搜索。参看针对 Windows 的 ANSYS 安装和配置指南（ANSYS Installation and Configuration Guide for Windows）。

4.2.3 在ANSYS 中生成宏

在 ANSYS 中可以通过三种途径来生成宏：

- 在输入窗口执行*CREATE 命令。参数的值不确定，参数名被写入文件中。
- 使用*CFOPEN, *CFWRITE, 和*CFCLOS 命令。参数名被其当前值取代，值被写入文件中。
- 选择 Utility Menu>Macro>Create Macro 菜单项。该方法打开一个可作为简单的多行编辑器的对话框来生成宏。参数的值不确定，参数名被写入文件中。

下面部分详细解释这些方法。

4.2.3.1 使用*CREATE

在命令输入窗口执行*CREATE 命令后，ANSYS 转向控制该命令指定的文件。在执行*END 命令之前，所有的命令都是对该文件进行操作。如果有一个和指定宏文件名同名的文件存在，ANSYS 程序将覆盖掉该文件。

例如，要生成一个名为 matprop.mac 的宏，用来定义材料特性。从命令输入窗口为该宏输入的命令为：

```
*CREATE,matprop,mac,macros\
MP,EX,1,2.07E11
MP,NUXY,1,.27
MP,DENS,1,7835
MP,KXX,1,42
*END
```

*CREATE 命令带的参数用来指定文件名、扩展名和所在路径。注意，UNIX 用户必须在路径的尾部再加上一道斜线（\）。

当使用*CREATE 命令时，所有在命令中用到的参数都会被写入文件（参数的当前值不会替换参数）。若当前参数值很重要，可以通过 PARSAV 命令把参数保存到某个文件中。

4.2.3.2 使用 *CFWRITE

若要生成一个参数被其当前值所代替的宏文件，可以使用*CFWRITE 来达到这一目的。和*CREATE 不同，*CFWRITE 不能指定宏文件名，因此必须首先用*CFOPEN 命令来指定宏文件名。在命令输入窗口输入的命令行只有以*CFWRITE 命令开头，才会被写入指定文件中，否则都会被运行。和*CREATE 一样，*CFOPEN 也能指定文件名、扩展名和所在路径。下面的例子把一个 BLOCK 命令写入当前打开的宏文件中：

```
*cfwrite,block,,a,,b,,c
```

注意，参数 a,b,c 用作 BLOCK 命令的实际参数。被写入文件的是这些参数的当前值，而不是参数名。因此，本例中被写入宏文件中的句子为：

```
*cfwrite,block,,4,,2.5,,2
```

要关闭宏文件，执行*CFCLOS 命令。

注意－虽然也可以通过这种方法来生成宏，但是这些命令更适合于在宏运行期间把

ANSYS 命令写入某个文件中去。

4.2.3.3 使用Utility Menu>Macro>Create Macro

通过这个菜单项可以打开一个作为简单的多行编辑器的对话框来生成宏。但是，不能通过它来打开并编辑已存在的宏。如果在对话框的*CREATE 域输入一个已经存在的宏名，那么该已经存在的宏就会被覆盖掉。

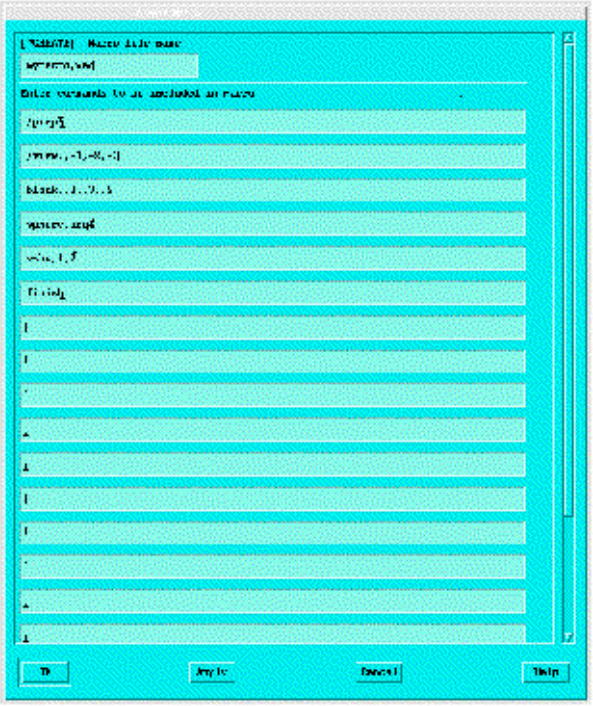


图 4-2 Utility Menu>Macro>Create Macro 对话框示例

和*CREATE 命令一样，参数不被求值，写入文件中的为参数名。注意，最后一行不能为*END 命令。

4.2.4 用文本编辑器生成宏

可以用任何喜欢的文本编辑器来生成或修改宏文件。任何 ASCII 编辑器都可以利用，而且 ANSYS 宏中的行可以由 UNIX 或 Windows 行结束符（回车换行或换行符）来终止，因此在一个平台上生成的宏可以在好几个平台上运行。

在用这种方法生成宏时，不要包含有*CREATE 和*END 命令。

图 4-3 在文本编辑器中生成的一个简单的宏示例

4.2.5 使用宏库文件

为更方便起见，ANSYS 允许把一批宏放到一个文件中，该文件被称为宏库文件。通过*CREATE 命令或文本编辑器都可以达到这一目的。由于宏库文件要比单独的宏长得多，所以使用文本编辑器是一个更好的选择。

宏库文件没有明确的文件扩展名，若有，不能超过 8 个字符。其文件名的命名规则和宏文件一样。宏库文件的结构为：

```
MACRONAME1
.
.
.
/EOF
MACRONAME2
.
.
.
/EOF
MACRONAME3
.
.
.
./EOF
```

例如，下面的宏库文件包括两个简单的宏：

```
mybloc
/prep7
/view,,-1,-2,-3
block,,4,,3,,2
finish
/EOF
mysphere
/prep7
/view,,-1,-2,-3
sphere,1
finish
/EOF
```

注意，每个宏的开始处都有一个宏名（有时也被称为数据块名），并以一个/EOF 命令结束。

宏库文件可以放在系统中的任何地方，为方便起见，最好放在宏的搜索路径中。

4.3 运行宏和宏库文件

可以通过*USE 命令来运行任何宏文件。例如，要运行在宏搜索路径中某个位置处的宏 MYMACRO（无扩展名），可通过执行以下命令来实现：

*use,mymacro

本例中，宏不带参数。假如要运行在/myaccount/macros 处的宏 MYMACRO.MACRO，那么执行的命令为：

*use,/myaccount/macros/mymacro.macro

注意，在*USE 命令中，路径、文件名和扩展名是作为一个参数输入的。

如果宏的文件扩展名为.mac，并且在宏的搜索路径中，那么就可以把它当作 ANSYS 命令来运行，在命令输入窗口输入即可。例如，要运行一个名为 mymacro.mac 的宏：

mymacro

也可以通过 Utility Menu>Macro>Execute Macro 菜单项来运行扩展名为.mac 的宏。

如果上一个宏带有参数（关于宏的参数传递内容见 4.4.1 部分），那么在命令行输入的内容为：

mymacro,4,3,2,1.5

或

*use,mymacro.mac,4,3,2,1.5

Utility Menu>Macro>Execute Macro 菜单项提供输入参数的域。

运行存在于宏库中的宏的方法也是相似的。首先，用*ULIB 命令指定库文件。例如，要运行存在于/myaccount/macros 路径处的 mymacros.mlib 文件中的宏，先指定库文件：

*ulib,mymacros,mlib,/myaccount/macros/

选择完宏库之后，就可以通过*USE 命令来运行任何包含在该库中的宏。同样也可以在*USE 命令中指定该宏所带参数。

注意－在执行*ULIB 命令后，不能用*USE 命令去访问没有包含在指定宏库文件中的宏。

4.4 局部变量

APDL 提供两套特殊命名的标量参数用作局部变量：

- 一套传递命令行变量到宏；
- 一套在宏内部使用。该局部变量仅仅在该宏中有效。

下面部分将详细讨论这两种类型的变量。

4.4.1 传递变量到宏

共有 19 个标量参数可以被用来从宏运行的命令行中传递变量到宏中。这些标量参数可以被许多宏重新使用，也就是说，它们的值只在每个宏的范围内有效。该参数名为从 ARG1 到 AR19，可用于以下项目：

- 数值；
- 文字数字字符串（括在单引号中，最多 8 个字符）；
- 数字或字符参数；
- 参数表达式

注意－只有从 ARG1 到 AR18 的参数的值可以用*USE 命令传递到宏中。对于可以象 ANSYS 命令一样运行的宏(该宏文件扩展名为.mac)，可以传递从 ARG1 到 AR19 的参数的值到宏中。

例如，下面的宏需要 4 个变量 ARG1, ARG2, ARG3, and ARG4：

```
/prep7
/view,,-1,-2,-3
block,,arg1,,arg2,,arg3
sphere,arg4
vsbv,1,2
finish
```

按如下方式运行该宏：

```
mymacro,4,3,2.2,1
```

4.4.2 宏内的局部变量

每个宏最多可有 79 个标量参数用作局部变量(从 AR20 到 AR99)。这些参数只在宏内部有效。在宏嵌套中，这些参数也不会互相传递。可以将它们传进任何通过执行一个/INPUT 命令调用的文件或宏的"do 循环中"。

4.4.3 宏外部的局部变量

ANSYS 还有一套类似的从 ARG1 到 AR99 的标量参数，它们在某个输入文件中有效，但不会传递到该输入文件调用的宏中。因此，一旦宏运行完毕，控制权返回给输入文件，ARG1 到 ARG99 参数的值恢复为在该输入文件中定义的值。

4.5 在APDL中控制程序流

运行一个输入文件时，ANSYS 通常线性执行程序，也就是说，按顺序一条一条语句地执行程序。但是，APDL 提供了一套丰富的命令来控制程序流：

- 子程序调用（宏嵌套）；
- 宏内部的无条件分支；
- 宏内部的条件分支；
- 重复执行某一条命令，增加一个或多个命令参数值；
- 按指定次数循环执行宏的某部分；

下面部分详细讨论了这些程序控制命令。关于这些命令的确切语法，请参阅 ANSYS 命令参考手册（ANSYS Commands Reference）。

4.5.1 宏嵌套：在宏内调用子程序

APDL 允许最多嵌套宏 20 层，其宏嵌套功能与 FORTRAN 77 中的 CALL 语句或函数调用功能相似。最多可以传递 19 个变量给宏，每个嵌套的宏运行完毕后，程序控制权返回给调用该宏的那一层。例如，在下面的宏库文件中，宏 MYSTART 调用宏 MYSPHERE 来生成一个球体：

```
mystart
/prep7
/view,,-1,-2,-3
```

```
mysphere,1.2
finish
/eof
mysphere
sphere,arg1
/eof
```

4.5.2 无条件分支：Goto

最简单的转向命令-*GO-指示程序转到某个指定标识字行处，不执行中间的任何命令。程序继续从该指定标识字行处开始执行。例如：

```
*GO,:BRANCH1
---      ！这个程序体被跳过(不执行)
---
:BRANCH1
---
---
```

由*GO 命令指定的标识字必须以冒号(:)开头，并不能超过 8 个字符（包括冒号）。该标识字可位于同一文件中的任何地方。

注意－不鼓励使用*GO 命令。最好使用其它的分支命令来控制程序流。

4.5.3 条件分支：*IF 命令

APDL 允许根据条件执行某些供选择的程序体中的一个。条件的值通过比较两个数的值（或等于某数值的参数）来确定。

*IF 命令的语法为：

*IF, VAL1, Oper, VAL2, Base

其中，

- VAL1 是比较的第一个数值（或数字参数）；
- Oper 是比较运算符；
- VAL2 是比较的第二个数值（或数字参数）；
- 若比较的值为真，则执行 Base 指定的操作。

APDL 提供了 8 个比较运算符，关于它们的详细讨论在*IF 命令参考部分。简要地说，它们主要是：

EQ ：等于（VAL1=VAL2）；
NE ：不等于 (VAL1#VAL2).
LT ：小于 (VAL1<VAL2).
GT ：大于 (VAL1>VAL2).
LE ：小于或等于 (VAL1vVAL2).
GE ：大于或等于 (for VAL1>=VAL2).
ABLT ：绝对值小于
ABGT ：绝对值大于

通过给 Base 变量赋值 THEN，*IF 命令就变成了 if-then-else 结构(和 FORTRAN 中的该

结构类似)的开始。该结构包括：

- 一个*IF 命令，接下来是
- 一个或多个*ELSEIF 命令选项
- 一个*ELSE 命令选项
- 一个必需的*ENDIF 命令，标识字该结构的结束。

在最简单的形式中，*IF 命令判断比较的值，若为真，则转向 Base 变量所指定的标识字处。结合一些*IF 命令，将能得到和其它编程语言中 CASE 语句相同的功能。注意，不要转向某个位于 if-then-else 结构或 do 循环中的带标识字的行。

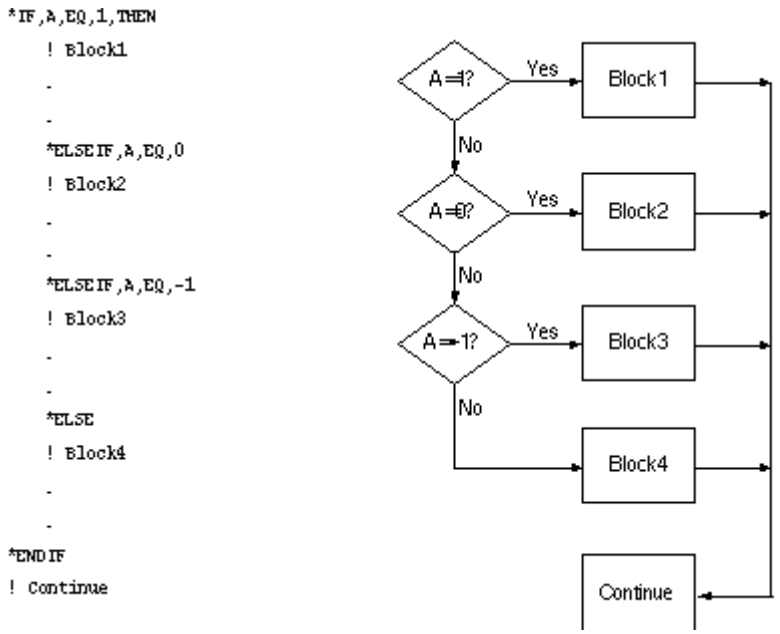
通过给 Base 变量赋值 STOP，可以离开 ANSYS。

if-then-else 结构仅仅判断条件并执行接下来的程序体或跳到*ENDIF 命令的下一条语句处（用"Continue"注释表示）：

```
*IF,A,EQ,1,THEN
      !   Block1
      .
      .
*ENDIF
! Continue
```

下面的例子用来说明一个较复杂的结构。注意，仅仅只能执行一个程序体。假如比较条件都不为真，就执行*ELSE 命令后的程序体。

图 4-4　if-then-else 结构的例子



注意－可以在 if-then-else 结构中执行/CLEAR 命令。/CLEAR 命令不会清除*IF 堆栈，*IF 层的号码仍然保留。必须用*ENDIF 来结束分支。同时，要记住/CLEAR 命令会删除所有的参数，包括在分支命令中使用的任何参数。为避免由于删除参数而引发问题，可以在/CLEAR 命令前运行/PARSAV 命令，然后在/CLEAR 命令后立刻运行/PARRES 命令。

4.5.4 重复一个命令

*REPEAT 命令是最简单的循环命令，通过它可以直接按指定的次数执行上一条命令，并按常数增加命令所带参数。例如：

```
E,1,2
*REPEAT,5,0,1
```

E 命令在节点 1 和 2 之间生成一个单元，*REPEAT 命令指示执行 E 命令 5 次（包括最初的一次），每执行一次第二个节点号加 1。结果共生成 5 个单元：1-2, 1-3, 1-4, 1-5, 和 1-6。

注意－大多数以斜线(/)或星号(*)开头的命令，以及扩展名不是.mac 的宏，都不可以重复调用。但是，以斜线(/)开头的图形命令可以重复调用。同时，要避免对交互式命令使用*REPEAT 命令，诸如那些需要拾取或需要用户响应的命令。

4.5.5 循环： Do循环

do 循环允许按指定的次数循环执行一系列的命令。*DO 和*ENDDO 命令分别是循环开始和结束点的标识字。

下面的 do 循环例子读取 5 个载荷步文件（从 1 到 5）并对 5 个文件做了同样的更改：

```
*DO,I,1,5            ! I = 1 to 5；
LSREAD,I             ！ 读取载荷步文件 I
OUTPR,ALL,NONE   ！ 改变输出控制
ERESX,NO
LSWRITE,I            ！ 重写载荷步文件 I
*ENDDO
```

也可以通过使用*IF, *EXIT, 或 *CYCLE 命令来丰富自己的循环控制。

在构造 do 循环时，要遵循以下原则：

- 不要通过在*IF 或 *GO 命令中带有:Label 来从 do 循环结构中跳出；
- 不要在 do 循环结构中用:Label 来跳到另外一行语句。可以用 if-then-else-endif 结构来代替；
- 在 do 循环结构中，第一次循环后，自动禁止命令结果输出。如果想得到所有循环的结果输出，就在 do 循环结构中使用/GOPR 或/GO (无响应行)命令。

在 do 循环结构中使用/CLEAR 命令要特别小心。/CLEAR 命令不会清除 do 循环堆栈，但是它会删除所有的参数，包括在本身的*DO 语句中的循环参数。为避免由此引发的循环值未定义的问题，可以在/CLEAR 命令前运行/PARSAV 命令，然后执行/CLEAR 命令和/PARRES 命令。

4.6 控制函数快速参考

下表描述了在宏中具有控制函数功能的 APDL 命令。

这些命令的大多数重要信息都在表中。关于这些命令的完整描述请参阅 ANSYS 命令参考手册（ANSYS Commands Reference ）。

APDL 命令	作用	用法提示
*DO	定义循环开始。紧接*DO 之后的命令（在*ENDDO 之前）反复执行，直到满足循环控制条件。	命令格式为*DO,Par,IVAL,FVAL,INC, 其中： Par 是用作循环指标的标量参数； IVAL 和 FVAL 是参数 Par 的初始和最终的值； INC 为每循环一次，IVAL 的增量。 可以通过*IF 命令来控制循环。 ANSYS 允许最多嵌套 20 层的"do"循环，但是包含/INPUT, *USE, 或"未知命令"宏的"do"循环支持少一些的嵌套层数，这是因为这些命令会进行内部的文件转换。在"do"循环中不要包含拾取操作。 在 do-loop 结构中使用/CLEAR 命令要特别小心。/CLEAR 命令不会清除 do-loop 堆栈，但是它会删除所有的参数，包括在本身的*DO 语句中的循环参数。为避免由此引发的循环值未定义的问题，可以在/CLEAR 命令前运行/PARSAV 命令，然后接着/CLEAR 命令运行/PARRES 命令。
*ENDDO	结束一个"do"循环，并开始下一个循环。	对每一个嵌套的"do"循环都要有一个*ENDDO 命令。一个循环的*ENDDO 和*DO 命令必须在同一个文件中。
*CYCLE	执行"do"循环时，ANSYS 程序绕过所有在*CYCLE 和*ENDDO 之间的命令，然后开始下一个循环。	可以有条件地使用 cycle 选项（通过*IF 命令）。 *CYCLE 命令必须和*DO 命令处于同一文件中，并且在*ENDDO 命令之前。
*EXIT	跳出"do"循环。	该命令执行紧接着*ENDDO 命令之后的下一行。一个循环的*EXIT 和*DO 命令必须在同一个文件中。可以有条件地使用 exit 选项（通过*IF 命令）。
*IF	按条件执行命令。	命令格式为：*IF,VAL1,Oper,Val2,Base,其中： VAL1 是比较的第一个数值（或数字参数）； Oper 是比较运算符：EQ (等于), NE (不等于), LT (小于), GT (大于), LE (小于或等于), GE (大于或等于), ABLT (绝对值小于) 或 ABGT (绝对值大于) VAL2 是比较的第二个数值（或数字参数）； 若比较的结果为真，则执行 Base ，若为假，则执行下一条语句。Base 可能为： - :label (跳到以标识字 label 开头的语句处) - STOP (跳出 ANSYS 程序) - EXIT (跳出当前的"do" 循环) - CYCLE (跳到当前"do"循环的结束处) - THEN (构造 if-then-else 结构) *IF 程序体最多可嵌套 10 层。不能在"do"循环或 if-then-else 结构中跳到某个:label 行。 可以在 if-then-else 结构中执行/CLEAR 命令。

		/CLEAR 命令不会清除*IF 堆栈，*IF 层的号码仍然保留。必须用*ENDIF 来结束逻辑分支。同时，要记住/CLEAR 命令会删除所有的参数，包括在分支命令中使用的任何参数。为避免由于删除参数而引发问题，可以在/CLEAR 命令前运行/PARSAV 命令，然后接着/CLEAR 命令运行/PARRES 命令。
*ENDIF	终止 if-then-else 结构	*IF 和*ENDIF 命令必须处于同一文件中
*ELSE	在 if-then-else 结构中生成最后一个可选择的程序体	*ELSE 和*ENDIF 命令必须处于同一文件中
*ELSEIF	在 if-then-else 结构中生成一个可选择的中间程序体	命令格式为*ELSEIF,VAL1,Oper,VAL2, 这里： VAL1 是比较的第一个数值（或数字参数）； Oper 是比较运算符：EQ (等于), NE (不等于), LT (小于), GT (大于), LE (小于或等于), GE (大于或等于), ABLT (绝对值小于) 或 ABGT (绝对值大于) VAL2 是比较的第二个数值（或数字参数）； 如果 Oper=EQ 或 NE, VAL1 和 VAL2 可以是字符串 (括在单引号中) 或参数。 *IF 和 *ELSEIF 命令必须处于同一文件中。

4.7 在宏中使用 _STATUS 和 _RETURN 参数

ANSYS 在运行中会产生两个参数_STATUS 和 _RETURN，可以在宏中使用它们。例如，可以在一个"if-then-else"结构中使用_STATUS 和 _RETURN 的值来判断某个 ANSYS 命令或函数产生的结果，从而决定让宏采取某些行动。

实体建模函数会产生_RETURN 参数，该参数为函数运行的结果。下表说明了各种实体建模函数产生的_RETURN 值：

命令	功能	返回_RETURN 值
关键点		
K	定义关键点	关键点号
KL	在线上生成关键点	关键点号
KNODE	在节点处生成关键点	关键点号
KBET	在两个关键点之间生成关键点	关键点号
KCENT	在圆心处生成关键点	关键点号
线		
BSPLIN	生成样条拟合的三次曲线	线号
CIRCLE	生成圆	第一条线的号码
L	在两关键点之间生成线	线号
L2ANG	生成与已有两条线成一定角度的线	线号
LANG	生成与一条线成一定角度的线	线号
LARC	生成一条圆弧	线号
LAREA	在两关键点之间生成线	线号
LCOMB	把两条线合并为一条线	线号

LDIV	把一条线分成两条或更多条线	第一个关键点号
LDRAG	通过关键点按一定路径延伸生成的线	第一条线的号码
LFILLT	两条相交线之间生成的倒角线	倒角线号码
LRCS	绕两关键点定义的轴线旋转点	线号
LROTATE	旋转关键点生成的弧	第一条线的号码
LSPA	在面上的投影线段	线号
LSTR	直线	线号
LTAN	生成一条与已有线共末点并相切的线	线号
SPLINE	生成一系列多义线	第一条线的号码
面		
A	连接关键点生成面	面号
ACCAT	连接两个或以上面生成面	面号
ADRAG	沿一定路径拖拉线生成面	第一个面的号码
AFILLT	在两个面的相交处生成倒角面	倒角面号码
AL	由线围成的面	面号
ALLP	所有的封闭链	面号
AOFFST	偏移某给定面生成面	面号
AROTAT	绕轴旋转线生成的面	第一个面的号码
ASKIN	通过引导线蒙皮生成的表面	第一个面的号码
ASUB	拷贝一个面的部分生成面	面号
体		
V	通过关键点生成体	体号
VA	面围成的体	体号
VDRAG	拖拉面生成体	第一个体的号码
VEXT	挤压面生成体	第一个体的号码
VOFFST	从给定面偏移生成体	体号
VROTATE	旋转面生成体	第一个体的号码

无论是在宏中还是在其它地方，运行 ANSYS 命令就会生成参数_STATUS。该参数反映的是所运行命令的出错状态：

0, 表示没有错误

1, 表示注意

2, 表示警告

3, 表示错误

4.8 在组和组件中使用宏

为了便于管理大模型，可以根据实体类型的不同把该模型分成一些离散的组：节点，单元，关键点，线，面或体。每一个组可以只包含某一种类型的实体。这样，就可以很方便地处理一些诸如加载、显示图形等方面的工作，并且可以分开处理模型的不同部分。

同时，也可以结合两个或以上的组，甚至组件来生成组件。组件最多可以嵌套 5 层。例如，可以构造一个包含名为 STATOR, PERMMAG, ROTOR, 和 WINDINGS 的组的组件 MOTOR。

下表描述了在构造组和组件时需要用到的命令。关于这些命令的完整描述请参阅

ANSYS 命令参考手册（ANSYS Commands Reference ）。关于组件和组的详细信息在 ANSYS 基本过程手册（ANSYS Basic Analysis Procedures Guide）第 7 章中。

命令	描述
CM,Cname,Entity	根据某种几何信息定义组。参数为： Cname -组名，最多有 8 个字符，并必须以字母开头； Entity - 四个字符的标识字，指定几何信息的类型，可能的值为 VOLU, AREA, LINE, KP (关键点), ELEM 和 NODE。
CMDELE,Name	删除一个组件或组。参数 Name 指定将删除的实体。
CMEDIT, Aname,Oper,Cnam1,... Cnam7	编辑一个已经存在的组件。当下层的实体或组件被删除时，ANSYS 程序自动更新组件。命令参数为： Aname - 要编辑的组件名； Oper - 运算符(ADD 为加组, DELE 为移走组)； Cnam1,...Cnam7 - 将被加入到组件中或从组件中删除的组件和组名。它们之间要用逗号隔开。
CMGRP, Aname,Oper,Cnam1,... Cnam8	组织组件和组到某组件中。该组件一旦定义，就可以对其使用和组一样的列出、删除、选择、不选择等命令。参数为： Aname - 组件名。最多有 8 个字符，并必须以字母开头； Cnam1,...Cnam8 -将要被加入到组件中去的已存在的组件和组名。它们之间要用逗号隔开。组件最多可以嵌套 5 层。
CMLIST,Name	列出某个组件或组包含的实体。Name 是操作对象组件或组的名称。若不指定该参数，操作对象为所有的组件和组。
CMSEL,Type,Name	选择组件和组中的实体生成子集。Type 标识字确定选择运算的类型： 1. S - 选择生成一个新的子集，为缺省值； 2. R - 从子集中再选择一个子集； 4. A - 选择一个子集添加到当前子集中。 4. U - 去掉当前子集的一部分； 5. ALL - 选择所有的组； 6. NONE - 不选择组件。

4.9 复习宏例子

下面是两个宏的例子。例子宏 offset.mac 在 PREP7 处理器中偏移选择的节点。该宏实现的功能与 NGEN 命令相似。

```
!    在 PREP7 处理器中偏移选择的节点的宏
!    下面的文件被保存为 offset.mac (必须小写)
!    用法：  offset,dx,dy,dz

/nop                          ! 禁止打印输出该宏

*get,nnode,node,,num,max      ! 得到节点号

*dim,x,,nnode                 ! 定义用于存放节点坐标的数组
```

```
*dim,y,,nnode
*dim,z,,nnode

*dim,sel,,nnode              ! 定义用于存放选择矢量的数组

*vget,x(1),node,1,loc,x      ! 得到坐标值
*vget,y(1),node,1,loc,y
*vget,z(1),node,1,loc,z

*vget,sel(1),node,1,nsel     ! 得到选择集

*voper,x(1),x(1),add,arg1    ! 偏移坐标值
*voper,y(1),y(1),add,arg2
*voper,z(1),z(1),add,arg3

! *do,i,1,nnode                    ! 存储新坐标值
!   *if,sel(i),gt,0,then         ! 处理 100,000 个节点需费时 98 秒
!      n,i,x(i),y(i),z(i)
!   *endif
! *enddo

!    ***** NEW FORM AT 5.3 *****
*vmask,sel(1)                    ! 处理 100,000 个节点需费时 3 秒
!    注意：在 5.3 版中， NNODE 要做如下处理：
n,(1:NNODE),x(1:NNODE),y(1:NNODE),z(1:NNODE)

x(1) =          ! 删除参数(清除)
y(1) =
z(1) =
sel(1) =
i=
nnode=

/go              ! 恢复打印输出
```

下面的例子为宏 bilinear.mac，用来生成两种双线性的材料。在完成静态分析之后使用该宏非常有用。Material 1 是拉伸特性， Material 2 为压缩特性。ARG1 为反复次数 (缺省为 2)。

```
/nop
_niter = arg1                            ! 设置反复次数
*if,_niter,lt,2,then
    _Niter = 2
*endif
```

```
*do,iter,1,_niter                          ！按反复次数循环
/post1
set,1,1
ar11,=elmiqr(0,14)                          ！调用 elmiqr 函数来得到单元号
*dim,_s1,,ar11                              ！定义用于存放单元的数组 s1
*dim,_s3,,ar11                              ！定义用于存放单元的数组 s3
etable,sigmax,s,1                           ! s1 在单元表 sigmax 中
etable,sigmin,s,3                           ! s3 在单元表 sigmin 中
*vget,_s1(1),elem,1,etab,sigmax             ！得到在 s1 中的单元最大应力
*vget,_s3(1),elem,1,etab,sigmin             ！得到在 s3 中的单元最大应力
*dim,_mask,,ar11                            ！定义屏蔽数组
*voper,_mask(1),_s1(1),lt,0                 !若最大应力< 0 为真
*vcum,1                                     ！单元压力求和
*vabs,0,1                                   ! s3 绝对值
*voper,_mask(1),_s3(1),gt,_s1(1)            ！若最小应力的绝对值>最大应力为真
finish

/prep7
mat,1                                       ！设置材料为拉伸特性
emod,all

*vput,_mask(1),elem,1,esel                  ！选择压缩单元
mat,2                                       ！将选择的单元改为受压
emod,all

call                                        ！选择所有单元
finish

_s1(1)=                                     ！清除所有的向量（设为 0）
_s3(1)=
_mask(1)=

/solve                                      ！返回到分析
solve
finish

*enddo                                      ！迭代结束

_niter=                                     ！清除迭代计数变量
_iter=
/gop
```

第五章 GUI用户界面

在一个 ANSYS 宏中，可以通过许多方法去访问 ANSYS 图形用户界面 (GUI)的组件：

- 可以改变和更新 ANSYS 工具条；
- 可以通过*ASK 命令提示用户输入某个参数的值；
- 可以生成对话框提示用户输入多个参数的值；
- 可以通过*MSG 命令让宏写一条输出信息；
- 可以让宏更新或移走状态条；
- 可以允许用户在宏中通过图形拾取选择实体；
- 可以调用任何对话框。

5.1 提示用户输入某个参数的值

通过在宏中包含*ASK 命令，该宏就可以提示用户输入某个参数的值。

*ASK 命令的格式为：

*ASK,Par,Query,DVAL

这里：

- Par 表示存储用户输入数据的标量参数的名称；
- Query 为文本串，ANSYS 用它来提示用户。该文本串最多可包含 54 个字符。不要使用具有特殊意义的字符，诸如"$" o 或"!"。
- DVAL 为用户用空格响应时，赋给该参数的缺省值。该值可以是一个 1-8 个字符的字符串（括在单引号中），也可以是一个数值。如果没有赋缺省值，用户用空格响应时，该参数被删除。

*ASK 在屏幕上显示询问文本，并等待响应。除非 ANSYS 在批处理模式下运行（在该模式下，响应为批处理中的下一输入行），该命令都从键盘接受响应。响应可以是一个括在单引号中的含 1-8 个字符的字符串，一个数值，一个数值或字符变量，或者是一个结果为数值的表达式。ANSYS 将把从响应中读到的值赋给参数 Par。例如：

*ask,parm1,'username (enclose the username in single quotes)'

显示如下所示的对话框，然后把用户输入的值赋给参数 PARM1。

图 5-1 *ASK 对话框例子

在宏中执行*ASK 命令时，ANSYS 把用户的响应写到文件 File.LOG 中宏名的下一行。

5.2 用户提示对话框

MULTIPRO 命令可以构造一个简单的多行提示对话框，该对话框最多可以包含 10 个参数提示。该命令允许使用 UIDL 中的 *CSET 命令来产生提示，并为每个提示指定缺省值。

MULTIPRO 命令必须和以下命令一同使用：

- 1-10 个*CSET 命令；
- 最多两个允许用户输入两行内容的*CSET 命令。

该命令语法为：

```
MULTIPRO,'start',Prompt_Num
*CSET,Strt_Loc,End_Loc,Param_Name,'Prompt_String',Def_Value
MULTIPRO,'end'
```

其中：

'start'

文字串，作为第一个参数，标识字 MULTIPRO 结构的开始。必须括在单引号中。

Prompt_Num

至少有一个*CSET 命令省略了 Def_Value 参数或 Def_Value 设为 0，才必须用到该参数。该参数为整形数，等于接下来的*CSET 提示的数目。

Strt_Loc,End_Loc

对第一个*CSET 命令，Strt_Loc 参数的初始值为 1。End_Loc 的值为 Strt_Loc+2（对第一个*CSET 命令，值为 3）。以后 Strt_Loc 的值为 End_Loc+1。

Param_Name

参数名，用来存储用户输入的值。若用户没有输入，其值为 Def_Value 的值。

'Prompt_String'

字符串，最多可包含 32 个字符，用来描述参数。必须括在单引号中。

'end'

文字串，用来结束 MULTIPRO 命令。

下面是一个 MULTIPRO 命令的典型例子：

```
multipro,'start',3
    *cset,1,3,beamW,'Enter the overall beam width',12.5
    *cset,4,6,beamH,'Enter the beam height',23.345
    *cset,7,9,beamL,'Enter the beam length',50.0
multipro,'end'
```

最多可以向该结构中添加两个能提供两行字符（共 64 个字符）提示的*CSET 命令。这种特殊的*CSET 命令的语法为：

```
*CSET,61,62,'Help_String','Help_String'
*CSET,63,64,'Help_String','Help_String'
```

其中：

'Help_String'

最多可包含 32 个字符的字符串。如果提示超过 32 个字符，可以使用第二个 Help_String 参数。

下面就是一个使用该结构的例子。注意，通过使用两个 Help_String 参数，突破了 32 个字符的限制。

```
multipro,'start',3
    *cset,1,3,dx,'Enter DX Value',0.0
```

```
*cset,4,6,dy,'Enter DY Value',0.0
*cset,7,9,dz,'Enter DZ Value',0.0
*cset,61,62,'The MYOFSET macro offsets the',' selected nodes along each'
*cset,63,64,'of the three axes.   Fill in the ',' fields accordingly.'
multipro,'end'
```

上面的结构将生成如下所示的多行提示对话框。

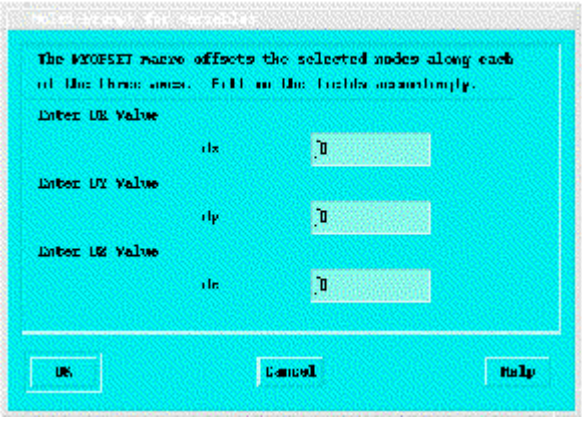


图 5-2 A 典型的多行提示对话框

可以通过检查_BUTTON 参数的值来检查按钮的状态。下面列出了按钮状态值：

_BUTTON=0 表示按下了 OK 按钮；

_BUTTON=1 表示按下了 Cancel 按钮；

目前，Help 按钮还不能工作。

5.3 用宏显示消息

通过在宏中使用*MSG 命令，可以通过 ANSYS 消息子程序来显示定制的输出消息。该命令格式如下：

*MSG,Lab,VAL1,VAL2,VAL3,VAL4,VAL5,VAL6,VAL7,VAL8

这里， Lab 是输出和终止控制的标识字，具体如下表所示：

INFO	所写的消息不带标题(缺省)。
NOTE	所写的消息带标题"NOTE"。
WARN	所写的消息带标题"WARNING"， 并把该消息写入出错文件 Jobname.ERR 中。
ERROR	所写的消息带标题"ERROR" ，并把该消息写入出错文件 Jobname.ERR 中。如果处于 ANSYS 批处理运行模式，该标识字在最初的"clean exit"处终止运行。
FATAL	所写的消息带标题"FATAL ERROR"，并把该消息写入出错文件 Jobname.ERR 中。该标识字立即终止 ANSYS 运行。
UI	所写的消息带标题"NOTE" ，并在消息对话框中显示该消息。

VAL1-VAL8 为该消息中包含的数字或字符值。该值为参数的结果。所有数值都为双浮点数。

必须在*MSG 命令之后立即指定消息的格式。消息格式最多可包含 80 个字符，包括文本串和在串之间预定义的“数据描述符”，在串和串之间将插入数字或字符数据。数据描述符为：

- %i，表示整形数据；
- %g，表示双浮点数据；
- %c，表示字符数据；
- %/, 表示一行结束。

对于前三个描述符，FORTRAN 中对应的数据描述符分别为 I9， 1PG16.9，和 A8。在每个描述符前要有一个空格。必须为每个指定数据值（最多 8 个）并按顺序提供一个数据描述符。

在*MSG 格式行中不要以*IF, *ENDIF, *ELSE, 或*ELSEIF 开头。如果消息中最后一个非空格字符是一个&，那么 ANSYS 程序把下一行当作*MSG 格式的延续。最多可以用 10 行（包括第一行）来确定格式信息。连续的空格输出将被压缩成一个空格, 并追加一个句点。产生的输出最多可以到达 10 行，每行最多包含 72 个字符(使用$/ 标识字)。

在下面的例子中，*MSG 命令显示一条包含一个字符数据、两个整形数和一个实数的消息：

```
*MSG, INFO, 'Inner' ,25,1.2,148
Radius ( %C) = %I, Thick = %G, Length = %I
```

输出消息为：

```
Radius (Inner) = 25, Thick = 1.2, Length = 148
```

注意－ 当 GUI 被激活时，/UIS,MSGPOP 命令控制消息对话框的显示。关于该命令的更多信息见 ANSYS 命令参考手册（ANSYS Commands Reference）。

5.4 在宏中生成并维护状态条

在宏中，可以通过插入命令来定义一个 ANSYS 对话框，该对话框包含一个显示运行进程的状态条、一个可用来终止运行的 STOP 按钮，或者两者都包含。

通过下面的命令来定义状态对话框：

*ABSET,Title40,Item

- Title40 是文本串，显示在带状态条的对话框中。最多可包含 40 个字符。
- Item 可取以下的值：

BAR	显示状态条，不显示 STOP 按钮
KILL	显示 STOP 按钮，不显示状态条
BOTH	状态条和 STOP 按钮都显示

通过*ABCHECK,Percent,NewTitle 命令来更新状态条，在该命令中：

- Percent 是一个在 0-100 间的整数。用来确定状态条的位置。
- NewTitle 是一个 40 个字符的字符串，表示进程信息。其内容将取代 Title40 中的字符串。

如果指定 Item 为 KILL 或 BOTH，该宏将在每次执行完*ABCHECK 命令之后检查_ERROR 参数，此时，如果用户按了 STOP 按钮，就会执行相应的动作。

可以通过*ABFINI 命令从 ANSYS GUI 中移走状态条。

下面的宏例子说明了如何使用状态条（包括 STOP 按钮）。生成的状态对话框在接下来的图中显示。注意，宏将检查_ERROR 参数，如果用户按了 STOP 按钮，就会显示"We are stopped....."消息。

```
fini
/clear,nost
/prep7
```

```
n,1,1
n,1000,1000
fill
*abset,'This is a Status Bar',BOTH
myparam = 0
*do,i,1,20
    j = 5*i
    *abcheck,j
    *if,_return,gt,0,then
       myparam = 1
    *endif
    *if,myparam,gt,0,exit
    /ang,,j
    nplot,1
    *if,_return,gt,0,then
       myparam = 1
    *endif
    *if,myparam,gt,0,exit
    nlist,all
    *if,_return,gt,0,then
       myparam = 1
    *endif
    *if,myparam,gt,0,exit
*enddo
*if,myparam,gt,0,then
*msg,ui
We are stopped.........
*endif
*abfinish
fini
```

注意－在一个循环中，调用*ABCHECK 的次数不能超过 20 次。

图 5-3 典型的状态对话框（包括状态条和 STOP 按钮））

5.5 在宏中进行拾取操作

在交互式运行 ANSYS 程序时，可以从宏中调用 GUI 拾取菜单。要达到这一目的，只需在宏中包含一条拾取命令即可。许多 ANSYS 命令(例如 K,,P)接受输入"P"以进行图形拾取。当 ANSYS 碰到这样一条命令时，就将显示正确的拾取对话框，在用户按 OK 或 Cancel

按钮后，ANSYS 将继续运行该宏。

注意，拾取命令在有些 ANSYS 处理器中不可用，因此，在调用这些命令之前必须先转换到合适的处理器中。

注意－如果宏中包含有 GUI 函数，该宏中的第一条命令应为/PMACRO 命令。该命令使宏的内容被写入日志文件中。这一点很重要，因为如果你省略了/PMACRO 命令, ANSYS 并不将任务日志文件读到 ANSYS 任务重执行环境中去。

5.6 在宏中调用对话框

当 ANSYS 程序碰到一个对话框 UIDL 函数名(如 Fnc_UIMP_Iso)时，就会显示对应的对话框。因此，可以通过在宏中把该函数名写为单独的一行来弹出对应的对话框。关闭对话框后，程序继续执行宏的下一行命令。可以通过在线帮助系统或查阅 Utility Menu > Help > Using the GUI > Menu Tree Structure 菜单项得到可用对话框函数名的完整列表。记住，有些对话框受许多因素的控制，包括被激活的 ANSYS 处理器和是否满足某些已经存在的条件。例如，首先假设存在节点，才能弹出选择节点的对话框，如果没有节点，当用户点击 OK 或 Apply 按钮时就会出错。

注意－如果宏中包含有 GUI 函数，该宏中的第一条命令应为/PMACRO 命令。该命令使宏的内容被写入日志文件中。这一点很重要，因为如果你省略了/PMACRO 命令, ANSYS 并不将任务日志文件读到 ANSYS 任务重执行环境中去。

第六章 加密宏

ANSYS 提供给宏加密的功能，从而可使宏的源文件不可读。加密宏需要解密匙才能运行。解密匙可以明确地放在宏文件（可读的 ASCII 文件）中，也可以由用户在 ANSYS 中设置为一个全局解密匙。

6.1 准备加密宏

在加密宏之前，首先要和平常一样生成和调试宏。然后，在宏的第一行和最后一行分别加一个/ENCRYPT 命令。在宏的第一行加的/ENCRYPT 命令的格式为：

/ENCRYPT,Encryption_key,File_name,File_ext,Directory_Path/

这里：

- Encryption_Key 是一个 8 个字符的密码；
- File_name 是加密宏文件名的名称；
- File_ext 是可选项，表示加密宏文件的文件扩展名。若扩展名为.mac ，该宏就可以象命令一样使用了。
- Directory_Path/ 是可选项，表示目录路径，最多可包含 60 个字符。如果不想把加密宏文件放到主目录中，就必须用到该参数。注意，路径名的最后一个字符必须是"/" (Windows 系统中为"\") ，否则，最后的目录名会加到文件名中。

下面的例子说明了/ENCRYPT 命令在宏顶部和底部的用法：

```
/encrypt,mypasswd,myenfile,mac,macros/
/nopr
/prep7
/view,,-1,-2,-3
block,,arg1,,arg2,,arg3
sphere,arg4
vsbv,1,2
/gopr
finish
/encrypt
```

宏顶部的/ENCRYPT 命令指示 ANSYS 给文件加密，并用字符串"mypasswd"作为解密匙。以上程序序列将生成一个名为 myenfile.mac 的加密宏，并放在主目录下的/macros 子目录中。底部的/ENCRYPT 命令指示 ANSYS 停止加密，并把加密宏写到指定文件中。

注意－加密宏在第二行用/NOPR 命令来禁止把 ANSYS 命令写到日志文件中。如果不想让用户从日志文件中读到宏的内容，这个方法是很有效的。在最后的/ENCRYPT 命令之前最好执行/GOPR 命令以重新激活日志记录。

6.2 生成加密宏

如上所述，在宏的顶端和底部加上/ENCRYPT 命令后，只需在 ANSYS 中运行该宏，就

生成了加密版的宏。加密版的宏使用的名称和所处路径由在该宏顶端的/ENCRYPT 命令指定。加密版的宏看起来就象这样：

```
/DECRYPT,mypasswd
013^Z,^%
02x^0Se|Lv(yT.6>?
03J3]Q_LuXd3-6=m+*f$k]?eB
04:^VY7S#S>c>
05daV;u(yY
06T]3WjZ
/DECRYPT
```

注意，现在宏中的命令就被加密了，加密内容包括在两个/DECRYPT 命令之间。解密匙就是第一个/DECRYPT 命令所带的参数。

6.3 运行加密宏

只要把加密宏放在宏搜索路径中，就可以和运行其他宏一样运行加密宏。如果希望在宏文件中不带解密匙就运行加密宏，可以在 ANSYS 中定义该解密匙为一个“全局解密匙”：首先，用参数 PASSWORD 代替/DECRYPT 命令中的解密匙参数。因此，加密宏的第一行变为：/DECRYPT,PASSWORD。在运行该宏之前，通过 ANSYS 的命令输入行执行以下命令：

```
/DECRYPT,PASSWORD,Encryption_Key
```

这里，

Encryption_Key 为用于加密文件的解密匙。

现在就可以运行加密密码。要删除当前总体解密匙, 则执行下面的 ANSYS 命令：

```
/DECRYPT,PASSWORD,
```