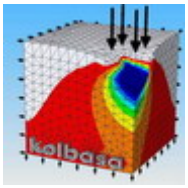


Перевод выполнил Широких А.С. (cyberkolbasa)

Данный текст может содержать ошибки, поскольку я не являюсь профессиональным переводчиком и не владею грамматикой английского языка в достаточной мере. Но, тем не менее, я старался максимально сохранить суть излагаемого материала со всеми необходимыми подробностями. Если же при прочтении этого текста кем-либо будут обнаружены грубые ошибки перевода, в результате которых содержание отдельных фраз или предложений кардинально меняет свой смысл, то прошу сообщить мне об этом в «гостевую» на сайте **http://cyberkolbasa.com1.ru** или на e-mail **cyberkolbasa@mail.ru**.

Справочник программиста APDL

Содержание

1. Введение в APDL

1.1. Что такое APDL?

APDL это Параметрический Язык Программирования ANSYS (ANSYS Parametric Design Language), язык сценариев, который Вы можете использовать, чтобы автоматизировать стандартные задачи или даже создавать вашу модель выраженную через параметры (переменные). APDL также охватывает широкий диапазон других возможностей, типа повторения команды, макроса, выполнения перехода "если-тогда-иначе", создания циклов, а также скалярных, векторных и матричных операций.

В то время как APDL - основа для сложных задач, типа оптимизационного расчета и адаптивного построения сетки, он также предлагает много удобств, которые Вы можете использовать в ваших обычных расчетах. В этом справочнике мы представим Вам основные возможности - параметры; макросы; выполнение условного перехода, циклы и повторения; параметры массивов - и покажем Вам несколько простых примеров. Поскольку Вы будете становиться более опытным в языке, Вы увидите, что приложения на APDL ограничены только вашим воображением.

Этот справочник охватывает следующие темы:

- **Работа с панелью инструментов**: Вы можете добавить часто используемые функции ANSYS или макрос к панели инструментов ANSYS, определяя сокращения, которые являются псевдонимами (восемь символов длиной) для команды ANSYS, названия{имени} функции GUI, или макро-названия{имени}.
- **Использование параметров**: Параметры - переменные APDL (они более подобны переменным ФОРТРАНа чем к параметрам Fortran). ANSYS использует два типа параметров: скаляр и массив.
- **APDL как макро-язык и создание макроса**: Вы можете сделать запись часто используемой последовательности команд ANSYS в макро-файле (их иногда называют файлами команды). Создание макроса допускает, Вы к, в действительности, создаете вашу собственную заказную команду ANSYS. В дополнение к выполнению ряда команд ANSYS, макрос может назвать{вызвать} функции GUI или передать значения в параметры.
- **Интерфейс GUI**: В пределах макроса ANSYS, Вы имеете несколько способов обратиться к компонентам ANSYS GUI (панель инструментов, диалоговое окно, команда ***ASK**, и т.д.).
- **Шифровка макроса**: ANSYS обеспечивает возможность зашифровать макрофайлы так, чтобы источник не был "удобочитаемым". Для шифрования макроса требуется запустить ключ кодирования. Вы можете поместить ключ кодирования явно (в читаемом ASCII) в макросе, или Вы можете установить это в ANSYS как глобальный ключ кодирования.

См. список всех команд APDL, обсуждаемых в этом справочнике, в **APDL Commands Reference**.

2. Работа с панелью инструментов

2.1. Добавление команд к панели инструментов

Вы можете добавить часто используемые функции ANSYS, или макрос к панели инструментов ANSYS (создание макросов охвачено, начиная с Главы 4 «APDL как макро-язык»). Вы делаете это, определяя аббревиатуру. Аббревиатура - псевдоним (восемь символов длиной) для команды ANSYS, имени функции GUI, или имени макроса. Например, MATPROP могло бы быть аббревиатурой для макроса, который перечисляет свойства материала, SAVE_DB - аббревиатура для команды сохранения, и QUIT - сокращение для функции Fnc_/EXIT (которая запускает диалоговое окно выхода из ANSYS).

Программа ANSYS обеспечивает два способа использовать аббревиатуры. Вы можете запустить аббревиатуру (и выполнить макрос, команду, и т.д.), набрав ее в командной строке. Если Вы используете ANSYS GUI, Вы можете также выполнить макрос или команду, нажимая соответствующую кнопку панели инструментов ANSYS.
Панель инструментов, которая показана на рис. 2.1, содержит кнопки, соответствующие существующим сокращениям.

Рис. 2.1. Панель инструментов.

Некоторые аббревиатуры, типа SAVE_DB, аббревиатуры, которые содержит панель инструментов и функции, которые они выполняют, уже предопределены до Вас. Одна панель инструментов может поддерживать до 100 аббревиатур (Вы можете "вложить" дополнительные панели инструментов, чтобы расширить это число). Вы можете переопределить или удалить аббревиатуры по желанию; однако, аббревиатуры автоматически не сохраняются и должны быть явно сохранены в файл и перезагружены для каждого сеанса ANSYS.

2.2. Изменение панели инструментов

Вы можете создать сокращения или через команду ***ABBR** или через **Utility Menu > Macro > Edit Abbreviations** или **Utility Menu > MenuCtrls > Edit Toolbar**. Использование одного из этих меню предпочтительнее по двум причинам:

1. Нажатие **OK** автоматически обновляет панель инструментов (использование команды ***ABBR**, требует использования **Utility Menu > MenuCtrls > Update Toolbar**, чтобы заставить вашу новую аббревиатуру появиться на панели инструментов).
2. Если потребуется Вы сможете легко отредактировать аббревиатуру.

Синтаксис для команды ***ABBR** и связанных диалогов:

***ABBR,** *Abbr, String*

Abbr

Имя аббревиатуры, которое появится на кнопке панели инструментов. Имя может содержать до восьми символов.

String

Имя макроса или команды, которую представляет *Abbr*. Если *String* - имя макроса, макрос должен быть в пределах макро-пути поиска файлов. Для получения дополнительной информации об использовании макроса, см. Главу 4 «APDL как макро-язык». Если *String* ссылается на интерактивное меню выбора (picking menu) или диалоговое окно (использующее UIDL), то определите **Fnc_string**. Например, на определениях аббревиатур

для **QUIT** и **POWRGRPH**, показанных выше, **Fnc_/QUIT** и **Fnc_/GRAPHICS** - уникальные имена функции UIDL, которые идентифицируют интерактивное меню выбора или диалоговое окно, связанного с аббревиатурами **QUIT** и **POWRGRPH** соответственно. Для получения дополнительной информации о функциях организации доступа UIDL см. «Диалоговые окна запроса из макроса». ***String*** может содержать до 60 символов, но не может включать ни одного следующего: символ «$»; команды **C*****, **/COM**, **/GOPR**, **/NOPR**, **/QUIT**, **/UI**, ***END**.

Значению по умолчанию панели инструментов ANSYS предопределены следующие сокращения:

```
*ABBR, SAVE_DB, SAVE
*ABBR, RESUM_DB, RESUME
*ABBR, QUIT, Fnc_/EXIT
*ABBR, POWRGRPH, Fnc_/GRAPHICS
```

2.2.1. Пример: Добавление кнопки панели инструментов

Например, чтобы добавить кнопку к панели инструментов, которая вызывает макро-файл **mymacro.mac**, Вам нужно ввести значения, которые показаны на рис. 2.2., в **Utility Menu > MenuCtrls >** в диалоговое окно **Edit Toolbar**.

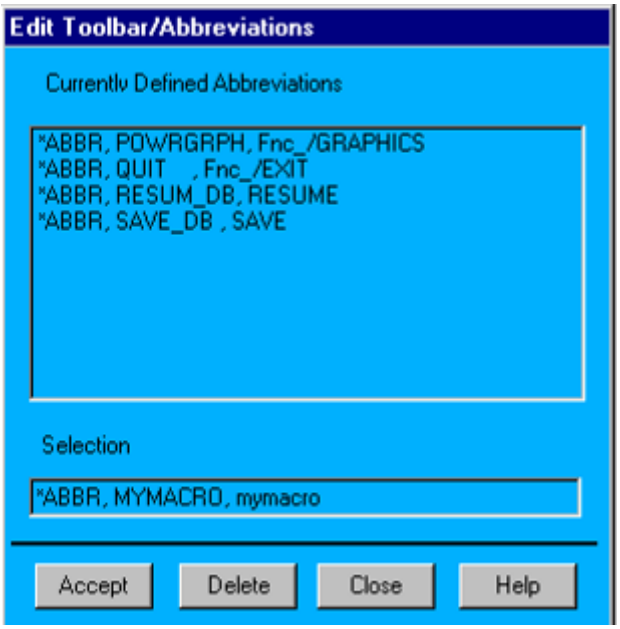


Рис. 2.2. Добавление новой аббревиатуры.

Новая кнопка вкладывается в панель как показано на рис. 2.3.

Рис. 2.3. Панель инструментов с новой кнопкой.

2.2.2. Сохранение кнопок панели инструментов

Кнопки панели инструментов не сохраняются от одного сеанса ANSYS до следующего; однако, они сохраняются и поддерживаются базой данных так, чтобы любой «RESUME» (откат) сеанса все еще содержит эти аббревиатуры. Чтобы сохранять ваши индивидуальные настройки кнопок, Вы должны явно сохранить их в файл через **Utility Menu > MenuCtrls > Save Toolbar** (команда **ABBSAV**) и восстановливать их для каждого

сеанса, используя **Utility Menu > MenuCtrls > Restore Toolbar** (команда **ABBRES**). Также Вы можете сделать это с помощью макроса.

Примечание.
Если какие-нибудь аббревиатуры уже существуют в файле с указанным именем, команда **ABBSAV** записывает поверх них.

Содержание файла аббревиатур – это команды APDL, которые используются, для создания аббревиатур. Таким образом, если Вы пожелаете отредактировать большой набор кнопок или изменить их порядок, достаточно воспользоваться текстовым редактором. Например, следующий текст является содержанием файла, который определяет значения кнопок панели инструментов по умолчанию.

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

***ABB** команды (сокращенная форма ***ABBR**) определяют кнопки. **/NOPR** выключает обращение к log-файлу (аббревиатуры в log-файл не записываются в момент считывания), в то время как **/GO** включает его (включается запись в log-файл).

2.3. Компоновка аббревиатур панели инструментов

Особенности сохранения и восстановления, описанные выше, позволяют Вам компоновать аббревиатуры. Компонуя аббревиатуры под одной кнопкой, Вы можете определять специализированные панели инструментов (если у Вас имеется много аббревиатур на одной панели инструментов, то это может затруднить поиск нужной Вам кнопки). Чтобы компоновать аббревиатуры, Вы просто определяете аббревиатуру, которая обращается в файл аббревиатур. Например, следующая команда определена как аббревиатура **PREP_ABR**, которая считывает аббревиатуры из файла prep.abbr.

```
*ABBR,PREP_ABR,ABBRES,,PREP,ABBR
```

PREP_ABR появится как кнопка на панели инструментов. Щелчок по ней заменит существующие кнопки набором кнопок, определенных в файле prep.abbr.
Определяя аббревиатуры, считывающие файлы, и включая их аббревиатуры в другие файлы, Вы можете иметь фактически неограниченное количество аббревиатур в текущем сеансе ANSYS. Вы можете даже пойти дальше и создать вашу собственную иерархическое меню, компонуя несколько файлов аббревиатур. Если Вы будете создавать такое иерархическое меню, то для того, чтобы передвигаться назад через меню, сделайте аббревиатуру «return» в каждом фале, это будет очень удобно.

3. Использование параметров

3.1. Параметры

Параметры - переменные APDL (они более подобны переменным Фортрана чем параметрам Фортрана). Вы можете явно не объявлять тип параметра. Все числовые значения (целые или вещественные) сохраняются как значения двойной точности (64-битовое представление действительного числа). Параметрам, которые используются, но не определены, автоматически присваивается значение близкое к нулю, приблизительно 2^{-100}. Например, если параметр A определен как A=B, и B не определен, то A присваивается значение близкое к нулю.

ANSYS использует два типа параметров: скаляр и массив. Первая часть этой главы содержит информацию, которая применима к обоим типам. Начиная с раздела «Параметры массива», информация касается только параметров массива.

Строки символов (восемь символов длиной) могут быть назначены для параметров, просто заключая строку в одиночные кавычки. APDL обеспечивает несколько типов параметров массива: числовой, символьный, строковый и табличный (специальный числовой тип, который автоматически интерполирует значения).

Вы можете использовать параметр (вместо числа или строки символов) как аргумент любой команды ANSYS; параметр вычисляется, и его текущее значение используется для аргумента. Например, если Вы присваиваете значение 2.7 параметру с именем AA и затем используете команду

```
N,12,AA,4
```

ANSYS интерпретирует команду как

```
N,12,2.7,4
```

(которая определяет узел №12 с координатами X=2.7 и Y=4).

Примечание

Если массив, таблица, или символьные параметры используются в пределах макроса или входного файла, то для этих параметров должны быть определена размерность (если массив или таблица), и они должны быть определены в пределах этого макроса или входного файла. Если Вы этого не сделаете, то ANSYS выдаст сообщения об ошибках, заявляющие, что эти параметры являются неопределенными. ANSYS выдаст сообщения об ошибках, даже если параметры имеют значение «ложь» в пределах невыполненного условного перехода ***IF**, поскольку замена параметра будет сделана перед выполнением условного перехода ***IF**.

3.2. Рекомендации по назначению имен параметров

Имена параметров должны:

- начинаться с буквы;
- содержать только буквы, числа, и знак подчеркивания;
- содержать не более 32 символов.

Примеры допустимых и недопустимых имен параметров.

Допустимые:

```
ABC
PI
X_OR_Y
```

Недопустимые:

MY_PARAMETER_NAME_LONGER_THAN_32_CHARACTERS (более 32 символов)
2CF3 (начинается с цифры)
M&E (содержит недопустимый символ "&")

Именуя параметры:

- Избегайте названий, которые соответствуют используемым меткам ANSYS, типа:
  - Метки степеней свободы (***DOF, TEMP, UX, PRES,*** и т.д.)
  - Convenience labels (***ALL, PICK, STAT,*** и т.д.)

    - Определяемые пользователем метки (такие как определенные с командой **ETABLE**)
    - Метки типа массива (такие как ***CHAR, ARRAY, TABLE,*** и т.д.)
- Имена параметров ***ARG1 … ARG9***, и ***AR10 … AR99*** зарезервированы для локальных параметров. Обычно локальные параметры используются в макросах. Использование этих имен как "регулярных" параметров не рекомендуется.
- Имена параметров не должны соответствовать аббревиатурам, определенным командой ***ABBR**.
- Не начинайте имя параметра с символа подчеркивания (_). Это соглашение зарезервировано для параметров, используемых GUI и поставляемыми ANSYS макросами.

Программисты APDL, работающие в организациях могут принять во внимание обозначение их параметров с замыкающим символом подчеркивания (_). Они могут отображаться как группа, используюя команду ***STATUS** и удаляться из памяти как группа через команду ***DEL**.

3.2.1. Сокрытие параметров от *STATUS

Listing Parameters выводит список параметров через команду ***STATUS**. Вы можете использовать параметр, присваивая имени соглашение "скрыть" параметры от ***STATUS**. Любые параметры, имена которых оканчиваются на символе подчеркивания (_) не будут перечислены ***STATUS**.

Эта возможность была добавлена в основном для тех, кто пишет макросы на APDL для других пользователей. Вы можете использовать эту возможность, чтобы писать свои макросы, которые ваши пользователи ANSYS и другие макро-программисты не смогут просмотреть.

3.3. Определение параметров

Если не отмечено специально, то информация в следующих нескольких разделах относится и к скаляру и к параметрам типа массив. Начиная с раздела «Массивы», информация относится только к параметрам типа массив.

Вы можете или назначить значения параметрам или извлечь значения, содержащиеся в ANSYS, и хранить эти значения в параметрах. Для того, чтобы извлечь значения из ANSYS, Вы можете использовать или команду ***GET**, или различные get-функции. Следующие разделы охватывают эти темы более подробно.

3.3.1. Назначение значений параметров во время процедур

Вы можете использовать команду ***SET**, чтобы определить параметры. Следующие примеры иллюстрируют применение этой команды:

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

Вы можете использовать "=" для определения параметров (это - самый удобный метод). Формат записи - *Имя* = *Значение*, где *Имя* – имя, назначаемое параметру, *Значение* - числовое или символьное значение, определенное для этого параметра. Значения символьных параметров должны быть заключены в одинарные кавычки и не могут превышать восемь алфавитно-цифровых символов. Следующее примеры иллюстрируют применение этого способа определения параметров:

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

В GUI, Вы можете или напечатать "=" непосредственно в окне ввода ANSYS или в поле "Selection" диалогового окна Scalar Parameter (доступ через **Utility Menu> Parameters> Scalar Parameters**).

3.3.2. Назначение значений параметров при запуске

Вы можете определить параметры как аргументы при запуске ANSYS из командной строки операционной системы. Просто напечатайте определения параметра после команды выполнения ANSYS (который является системной переменной), используя следующий формат *–Имя Значения*. Например, ниже показано определение двух параметров parm1 и parm2 со значениями 89.3 и -0.1:

```
ansys90 -parm1 89.3 -parm2 -0.1
```

Это хороший способ, чтобы избежать поручать имен параметров на один или два символа при запуске избегать конфликтов с опциями командной строки ANSYS.

Примечание

Помните, что оболочки UNIX обрабатывают одиночные кавычки и много других неалфавитно-цифровых символов как специальные символы. Определяя символьные параметры, Вы должны сказать UNIX не интерпретировать ковычки, вставляя обратную косую черту (\) перед одиночными ковычками. Следующий пример определяет параметры на два символа, имеющие значения 'filename' и '200'.

```
ansys90 -cparm1 \'filename\' -cparm2 \'200\'
```

Если Вы используете `ANSYS Launcher`, чтобы запустить ANSYS, Вы можете определить параметры через вкладку **Customization** (использующий формат *–Name Value*, описанный выше).

Если Вы будете определять большое количество параметров при запуске, то наиболее удобный способ для этого определить их в файле `start90.ans` или через отдельный файл, который Вы можете загрузить через команду **`/INPUT`** в командной строке.

3.3.3. Назначение параметрам значений содержащихся в базе данных ANSYS

ANSYS обеспечивает два мощных метода для того, чтобы отыскать значения:

- команда ***GET**, которая отыскивает значение указанного элемента и хранит его в указанном параметре.
- get-функции, которые могут использоваться в операциях. Каждая get-функция возвращает определенное значение из определенного элемента.

3.3.3.1. Использование команды *GET

Команда ***GET** (**Utility Menu> Parameters> Get Scalar Data**) отыскивает значение элемента из базы данных (узел, элемент, поверхность, и т.д.) и хранит его как определенный пользователем параметр. Различное ключевое слово, метка, и комбинации чисел идентифицируют найденный элемент. Например, ***GET,A,ELEM,5,CENT,X** возвращает x-координату центра тяжести элемента номер 5 и хранит результат как параметр A.

Синтаксис для команды ***GET**:

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

где

- *Par* - имя параметра.
- *Entity* - ключевое слово элемента, который будет сохранен. Допустимые ключевые слова - NODE, ELEM, KP, LINE, AREA, VOLU, и т.д. Полный список допустимых ключевых слов см. в описании команды ***GET**.
- *ENTNUM* - номер объекта (или ноль для всех объектов).
- *Item1* - имя элемента части объекта. Например, если *Entity* будет ELEM, то *Item1* будет или NUM (самое больший или самый меньший номер элемента в отобранном наборе) или COUNT (количество элементов в наборе). Полный список значений *Item1* см. в описании команды ***GET**.

Вы можете представить структуру команды ***GET** как путь от общего к частному.

Следующие примеры иллюстрируют использование команды ***GET**. Первая строка в примере ниже извлекает признак материала (MAT – номер материала) элемента 97 и присваивает его значение параметру BCD:

```
*GET,BCD,ELEM,97,ATTR,MAT     ! BCD = номер материала элемента 97
*GET,V37,ELEM,37,VOLU         ! V37 = объем элемента 37
*GET,EL52,ELEM,52,HGEN        ! EL52 = значение тепловыделения в элемен-
те 52
*GET,OPER,ELEM,102,HCOE,2     ! OPER = тепловой коэффициент элемента 102
с поверхности 2
*GET,TMP,ELEM,16,TBULK,3      ! TMP = средняя температура элемента 16 на
поверхности 3
*GET,NMAX,NODE,,NUM,MAX       ! NMAX = максимальный номер узла в актив-
ном наборе
*GET,HNOD,NODE,12,HGEN        ! HNOD = значение тепловыделения в узле 12
*GET,COORD,ACTIVE,,CSYS       ! COORD = номер активной системы координат
```

3.3.3.2. Использование get-функций

Предположим необходимо вычислить среднюю x-координату двух узлов. С помощью команды ***GET** это будет выглядеть следующим образом:

```
*GET,L1,NODE,1,LOC,X    ! присвоение параметру L1 x-координаты узла №1
*GET,L2,NODE,2,LOC,X    ! то же для узла №2
MID=(L1+L2)/2           ! вычисление середины
```

Используя get-функции эту операцию можно записать короче:

```
MID=(NX(1)+NX(2))/2  ! NX(n) – get-функция для получения x-координаты n-
го узла
```

Get-функции могут быть параметрами других get-функций. Например, следующая комбинация **NELEM (ENUM, NPOS)** возвращает номер узла в позиции NPOS для элемента ENUM. Комбинируя это выражение с **NX(n)** можно получить x-координату этого узла **NX(NELEM (ENUM, NPOS)).**

Полный перечень get-функций приведен в приложении B (Appendix B) справочной документации к ANSYS.

3.3.4. Листинг параметров

Используя команду ***STATUS** можно просмотреть все определенные вами параметры.

```
*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
```

Доступ к этой информации можно получить через GUI: **Utility Menu> List> Other> Parameters** или **Utility Menu> List> Status> Parameters> All Parameters.**

Примечание

Команда ***STATUS** не показывает параметры начинающиеся или оканчивающиеся на символ подчеркивания (_).

Хотя ANSYS допускает максимум 5000 параметров, для пользователя доступны меньше чем 5000 из-за GUI и макро-требований ANSYS. Число параметров, определенных пользовательским интерфейсом (внутренние параметры) можно просмотреть командой ***STATUS**. Команда ***GET, *Par*,PARM,,MAX** возвращает общее количество определенных параметров.

3.4. Удаление параметров

Вы можете удалить определенные параметры двумя способами:

- Используя символ "=" и оставляя правую часть команды пустой. Например, чтобы удалить параметр QR используйте эту команду:

  ```
  QR=
  ```

- Используя команду ***SET** (**Utility Menu> Parameters> Scalar Parameters**), но не определяйте значение для параметра. Например, чтобы удалить параметр QR через команду ***SET** введите следующую строку:

  ```
  *SET,QR,
  ```

Установка числового параметра, равного нолю не удаляет его. Точно так же установка символьного параметра, равного пробелу в одиночных кавычках (‘ ‘) удаляет его, равного пробелу без одиночных кавычек не удаляет параметр.

3.5. Использование символьных параметров

Как правило, символьные параметры используются, чтобы обеспечить имена файлам и расширениям. Желаемое имя файла может быть присвоено символьному параметру, и этот параметр может использоваться везде где требуется имя файла. Точно так же расширение файла может быть присвоено символьному параметру и использоваться соответствующим образом (как значение аргумента *Ext* команды). В пакетном режиме, это позволяет Вам легко изменять имена файла для многократных выполнений, просто заменяя начальное алфавитно-цифровое "значение" символьного параметра в вашем входном файле.

Примечание

Помните, что символьные параметры ограничены восьмью символами.

Ниже приведен список наиболее характерных применений символьных параметров.

- Как аргумент к любому доступному полю команды (то есть, где ожидается алфавитно-цифровой ввод),

- Как имя файла макроса для команды ***USE** (**Utility Menu> Macro> Execute Data Block**),

```
NAME='MACRO'    ! MACRO это имя файла макроса
*USE,NAME       ! вызов файла MACRO
```

- Как аргумент команды ***USE** и на "неизвестную команду" макроса. Любой из следующих макро-запросов допускается:

```
ABC='SX'
*USE,NAME,ABC

или

*USE,NAME,'SX'
DEF='SY'
NEWMACRO,DEF      ! вызов макроса NEWMACRO.MAC

или

NEWMACRO,'SY'
```

3.6. Замена числовых значений

Всякий раз, когда Вы используете имя параметра в числовом поле команды, его автоматически заменяют значением параметра. Если никакое значение не было присвоено параметру (то есть, если параметр не был определен), то будет присвоено значение близкое к нулю (2^{-100}), обычно без предупреждения.

Примечание

Определение параметра после того, как он использован в команде, "не обновляет" команду *в большинстве случаев*. (Исключения – команды **/TITLE, /STITLE, *ABBR, и /TLABEL**. Например:

```
Y=0
X=2.7
N,1,X,Y      ! Node 1 at (2.7,0)
Y=3.5        ! Redefining parameter Y now does not update node 1
```

3.6.1. Предотвращение замены

Вы можете предотвратить замену параметра, заключая имя параметра в одиночные кавычки (‘) например, 'XYZ'. Тогда используется буквенная строка; поэтому, эта возможность допустима только в *нечисловых* полях.

Наоборот, Вы можете вызвать замену параметра в заголовках, подзаголовках, и именах файла, заключая имя параметра символами процента (%). Например,

```
/TITLE, TEMPERATURE CONTOURS AT TIME=%TM%
```

определяет заголовок, в котором заменяют параметр TM числовым значением. Заметьте, что параметр заменяется в то время, когда заголовок используется.

3.6.2. Замена значений символьных параметров

Использование символьного параметра в алфавитно-цифровом поле команды приводит к автоматической замене его значения. Принудительная замена и символьные ограничения параметра объясняются ниже.

3.6.2.1. Принудительная замена

Как с числовыми параметрами, Вы можете вызвать замену символьного значения параметра в определенных случаях, где замена не произошла бы иначе. Это делается за-

ключением имени параметра символами процента (%). Принудительная замена символьных параметров допустима для следующих команд:

- команда **/TITLE** (поле *Title*). Определяет заголовки для различных печатанных входных данных.
- Команда **/STITLE** (поле *Title*). Определяет подзаголовки. (Вы не можете обратиться к этой команде непосредственно из GUI.)
- Команда **/TLABEL** (поле *Text*). Определяет текстовую строку для аннотации.
- Команда ***ABBR** (поле *Abbr*). Определяет аббревиатуру.

Принудительная замена также допустима в следующих типах полей:

- Любое имя файла или параметр команды расширения. Эти параметры обращаются к командам, типа **/FILNAME, RESUME, /INPUT, /OUTPUT,** и **FILE.** (Прямая замена параметра также допустима в этих полях.)
- Любое поле на 32 символа: типичный пример - имя макроса. (Прямая замена недопустима для этих полей.)
- Как имя команды в любом поле вызова команды. Также как "неизвестная команда" имени макроса в поле 1. Например:

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

Следующий пример показывает методы вызова замены для определения подзаголовка и для имени каталога.

```
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%      ! считать результат из /ANSYS/MODELS/TEST.RST
```

3.6.2.2. Другие случаи, где допустимы символьные параметры

В дополнение к более общим приложениям, которые уже обсуждались, есть некоторые определенные случаи, где символьные параметры позволяются добиться большего удобства. Ниже приведены команды, с которыми это возможно и детали их использования.

***ASK**

Эта команда запрашивает ввод алфавитно-цифровой строки (до восьми символов, заключенных в одиночные кавычки), которая становится символьным значением скалярного параметра. (Вы не можете обратиться к ***ASK** команде непосредственно в GUI.)

***CFWRITE**

Эта команда записывает команды ANSYS в файл, открывающемуся командой ***CFOPEN**. Это можно использовать, чтобы написать символьное значение параметра для этого файл. Например, ***CFWRITE, B = 'FILE'** допустимо. (Вы не можете обратиться к ***CFWRITE** и ***CFOPEN** непосредственно из GUI.)

***IF и *ELSEIF**

Символьные параметры могут использоваться для аргументов *VAL1 И VAL2* этих команд. Для аргумента *Oper*, только EQ (равный) и NE (не равный) могут использовать символьные параметры. (Вы не можете обратиться к ***IF** и ***ELSEIF** непосредственно из GUI.) Например:

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

***MSG**

Символьные параметры допускается вводить для *VAL1 ... VAL8*. Дескриптор %C используется, чтобы указать для алфавитно-цифровых данных формат вывода (который должен следовать за командой ***MSG**). %C соответствует дескриптору ФОРТРАНА A8. (Вы не можете обратиться к команде ***MSG** непосредственно из GUI.)

PARSAV и PARRES

Эти команды сохранят символьные параметры в файл (команда **PARSAV** или **Utility Menu> Parameters> Save Parameters**), и восстанавливает символьные параметры из файла (**PARRES** или **Utility Menu> Parameters> Restore Parameters**).

***VREAD**

Эта команда (**Utility Menu> Parameters> Array Parameters> Read from File)** может использоваться, чтобы читать алфавитно-цифровые символьные данные из файла и создавать массив символьных параметров. Дескриптор ФОРТРАНА (A) может использоваться в строке форматирования, которая должна следовать за командой ***VREAD**.

***VWRITE**

Эта команда (**Utility Menu> Parameters> Array Parameters> Write to File**), может использоваться, чтобы записать символьные данные параметра в файл в отформатированной последовательности. Дескриптор ФОРТРАНА (A) может использоваться в троке форматирования, которая должна следовать за командой ***VWRITE**.

3.6.2.3. Ограничения символьных параметров

Хотя символьные параметры имеют большую часть тех же самых функциональных возможностей как и числовые параметры, есть несколько случаев, где символьные параметры не допустимы.

- Символьная замена параметра не допускается для аргумента *Par* команд ***SET, *GET, *DIM,** и ***STATUS**.
- Интерактивное редактирование параметров массива (***VEDIT**) не доступно для символьных параметров массива.
- Векторные команды операции, типа ***VOPER, *VSCFUN, *VFUN, *VFILL, *VGET,** и ***VITRP,** не работают с символьными параметрами массива.
- Работая с символьными параметрами, команды детализации ***VMASK** и ***VLEN** применимы только к ***VWRITE** и ***VREAD**.
- Символьные параметры не допустимы в параметрических выражениях, которые используют сложение, вычитание, умножение, и т.д.

3.7. Динамическая замена числовых и символьных параметров

Динамическая замена параметров встречается для следующих команд: **/TITLE, /STITLE, *ABBR, /AN3D,** и **/TLABEL**. Динамическая замена позволяет пересмотренному

значению параметра использоваться, даже если команда, которая использует значение параметра, не была использована заново.

Пример:

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

Заголовок "This is CASE 1", появится в области печати.

Затем Вы можете изменить значение XYZ, и новый заголовок появится при последующих печатях, даже если Вы не использовали заново **/TITLE**.

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

Заголовок " This is CASE 2", появится при последующих печатях.

3.8. Параметрические выражения

Параметрические выражения используют операции с параметрами и числами, типа сложения, вычитания, умножения, и деления. Например:

```
X=A+B
P=(R2+R1)/2
D=-B+(E**2)-(4*A*C)      ! вычисляет D = -B + E2 - 4AC
XYZ=(A<B)+Y**2           ! вычисляет XYZ=A+Y2 если A<B;
                         ! иначе XYZ = B+Y2
INC=A1+(31.4/9)
M=((X2-X1)**2-(Y2-Y1)**2)/2
```

Ниже приведен полный список операторов APDL:

Оператор	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
**	Возведение в степень
<	Меньше
>	Больше

Вы можете также использовать круглые скобки для определенности и для того, чтобы "вкладывать" операций, как показано выше. Порядок, в котором программа ANSYS вычисляет выражение, следующий:

1. Операции в круглых скобках (самый внутренний первый)
2. Возведение в степень (справа налево)
3. Умножение и деление (слева направо)
4. Одноместная ассоциация (типа +A или -A)
5. Сложение и вычитание (слева направо)
6. Логическое вычисление (слева направо)

Таким образом выражение, типа Y2=A+B ** C/D*E будет вычислено в следующем порядке: B ** C, /D, *E , и +A. Для определенности, Вы должны использовать круглые скобки в подобных выражениях. Круглые скобки могут быть вложены до четырех уровней глубиной, и до девяти операций могут быть выполнены в пределах каждого набора круг-

лых скобок. Старайтесь избегать использовать пробелы между операторами в выражениях. В частности никогда не включите пробел перед символом умножения *, потому что остальная часть вводимой строки (начинающаяся с *) интерпретируется как комментарий и поэтому будет игнорироваться. (Не используйте этот способ для комментариев; используйте восклицательный знак (!) для этой цели.)

ABS(x)	Абсолютное значение x.
SIGN(x,y)	Абсолютное значение x со знаком y. Если y=0 то знак принимается положительным.
EXP(x)	Число e в степени x (e^x).
LOG(x)	Натуральный логарифм от x (ln (x)).
LOG10(x)	Десятичный логарифм от x ($\log_{10}(x)$).
SQRT(x)	Квадратный корень из x.
NINT(x)	Ближайшее целое к x.
MOD(x,y)	Остаток от деления x/y. Если y=0 возвращает ноль (0).
RAND(x,y)	Случайное число (равномерное распределение от x до y (x = нижняя граница, y = верхняя граница).
GDIS(x,y)	Случайная выборка Гауссовского распределения (нормальное распределение) со средним x и стандартным отклонением y.
SIN(x), COS(x), TAN(x)	Синус, косинус и тангенс от x, где x в радианах по умолчанию. Можно изменить на градусы при помощи команды ***AFUN**.
SINH(x), COSH(x), TANH(x)	Гиперболические синус, косинус и тангенс от x.
ASIN(x), ACOS(x), ATAN(x)	Арксинус, арккосинус и арктангенс от x. x должен быть между -1.0 and +1.0 для ASIN и ACOS. Результат вычисления по умолчанию в радианах, но можно изменить на градусы командой ***AFUN.** Диапазон вывода результатов от -pi/2 до +pi/2 для ASIN и ATAN, и от 0 до pi для ACOS.
ATAN2(y,x)	Арктангенс y/x с учетом знака каждого аргумента. По умолчанию результат в радианах, но можно заменить на градусы командой ***AFUN**. Диапазон вывода от -pi до +pi.
VALCHR (*CPARM*)	Числовое значение *CPARM* (если *CPARM* не числовой параметр, то возвращает 0.0).
CHRVAL (*PARM*)	Символьное значение числового параметра *PARM*. Количество десятичных знаков зависит от величины.
UPCASE *CPARM*	Прописные буквы эквивалентные *CPARM*.
LWCASE *(CPARM)*	Строчные буквы эквивалентные *CPARM*.

Ниже приведены примеры использования параметрических функций:

```
PI=ACOS(-1) ! PI = арккосинус от -1, PI вычисляется с машинной точностью
Z3=COS(2*THETA)-Z1**2
R2=SQRT(ABS(R1-3))
X=RAND(-24,R2)          ! X = случайное число от -24 до R2

*AFUN,DEG               ! изменении угловых единиц измерения на градусы
```

```
THETA=ATAN(SQRT(3))     ! вычисление THETA равной 60 градусам

PHI=ATAN2(-SQRT(3),-1)  ! PHI равно -120 градусам
*AFUN,RAD               ! изменении угловых единиц измерения на радианы

X249=NX(249)                          ! X-координата узла 249
SLOPE=(KY(2)-KY(1))/(KX(2)-KX(1))   ! уклон линии с точками 1 и 2

CHNUM=CHRVAL(X)         ! CHNUM = символьное значение X
UPPER=UPCASE(LABEL)     ! UPPER = верхний регистр символьного значения
                        ! параметра LABEL
```

3.9. Сохранение, восстановление и запись параметров

Если Вы должны использовать в настоящее время определенные параметры в другом сеансе ANSYS, Вы можете записать их в файл и затем считать (восстановить) тот файл. Когда Вы считаете файл, Вы можете или полностью заменить в настоящее время определенные параметры или добавить к ним (заменяя те, что уже существует).

Чтобы записать параметры в файл, используйте команду **PARSAV** (**Utility Menu> Parameters> Save Parameters**).

Файл параметров - файл ASCII, состоящий в значительной степени из APDL команд ***SET**, определяющих различные параметры. Следующий пример показывает формат этого файла.

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

Считать параметры из файла можно используя команду **PARRES** (**Utility Menu> Parameters> Restore Parameters**).

Если Вы желаете, Вы можете написать до десяти параметров или массив параметров, используя реальные форматы ФОРТРАНа для файла. Вы можете использовать эту возможность, чтобы записать ваш собственный выходной файл для использования в других программах, отчетах, и т.д. Чтобы сделать это, используйте команду ***VWRITE** (**Utility Menu> Parameters> Array Parameters> Write to File**).Массив параметров

3.10. Массивы

В дополнение к скалярному параметру (единственное значение), Вы можете определить параметр как массив (множественные значения). Массивы ANSYS могут быть:

- 1-D (один столбец)
- 2-D (строки и столбцы)
- 3-D (строки, столбцы и плоскости)
- 4-D (строки, столбцы, плоскости и книги)
- 5-D (строки, столбцы, плоскости, книги и полки)

ANSYS предусматривает три типа массивов:

ARRAY
Этот тип подобен массивам ФОРТРАНА 77 и по умолчанию тип массива определяется, когда задают его размерность. Как с массивами ФОРТРАНА, индексы для строк, столбцов, и плоскостей - последовательные целые числа, начинающиеся с единицы. Элементы массива могут быть или целыми числами или вещественными.

CHAR
Это - символьный массив, с каждым элементом, состоящим из алфавитно-цифрового значения, не превышающего восемь символов. Индексы для строк, столбцов, и плоскостей - последовательные целые числа, начинающиеся с единицы.

TABLE
Это - специальный тип числового массива, который позволяет ANSYS вычислять (через линейную интерполяцию) значения между этими элементами массива, явно определенными в массиве. Кроме того, Вы можете определить индексы массива для каждой строки, столбца, и плоскости и эти индексы вещественные числа (не целые). Элементы массива могут быть или целыми числами или вещественными. Как мы сможем увидеть далее, в более позднем обсуждении по TABLE-массивам, эта способность обеспечивает мощный метод для того, чтобы описывать математические функции.

STRING
Вы можете использовать команду **`*DIM`**, STRING чтобы ввести строки символов в ваши массивы. Индексные числа для столбцов и плоскостей - последовательные значения, начинающиеся с 1. Индексы строки определены позицией символа в строке. См. команду **`*DIM`** для получения дополнительной информации.

Все три типа массивов не могут превышать 2**31-1 байт. Для двоичного массива, каждый элемент данных - 8 байтов, таким образом предел на количество входных данных - (2** 31-1)/8.

Далее будут рассмотрены следующие разделы по массивам:

- Базовые массивы
- Примеры массивов
- Массивы типа TABLE
- Определение и листинг массивов
- Определение значений массива
- Запись файла данных
- Операции над массивами
- Построение графика векторных массивов
- Изменение меток кривой

3.10.1. Параметр основного массива

Рассмотрим двумерный массив (ARRAY или CHAR) как показано ниже. Где m – количество строк и n – количество столбцов; то есть, его размерность m:n. Каждая строка идентифицируется числовым индексом i, который изменяется от 1 до m, и каждый столбец идентифицирован индексом j, который изменяется от 1 до n. Параметры, которые составляют массив называются элементами массива. Каждый элемент массива идентифицируется как (i, j), где i - его номер строки, и j - его номер столбца.

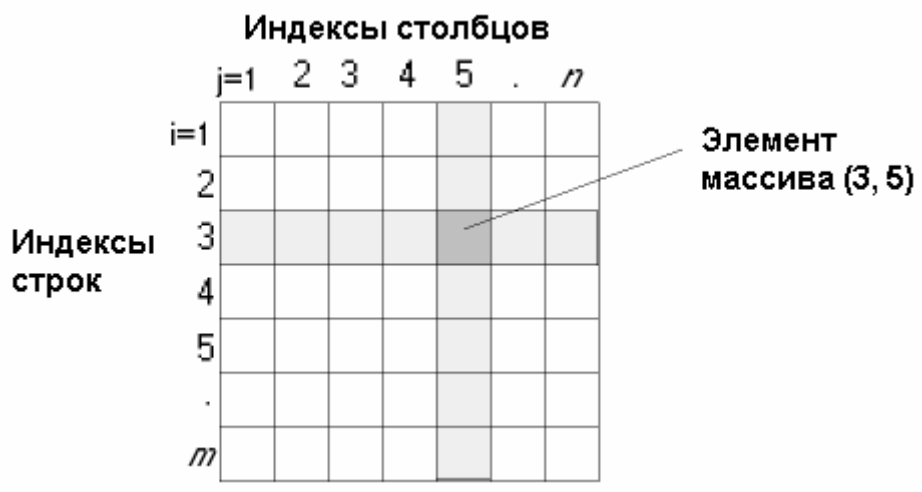


Рис. 3.1. Графическое представление двумерного массива.

Мы можем расширить эти представления трехмерным массивом, который может быть m-строк в длину, n-столбцов в ширину и p-плоскостей в глубину. Номер индекса плоскости – k, который изменяется от 1 до p. Каждый элемент массива идентифицируется как (i, j, k). Следующий рисунок показывает трехмерный массив.

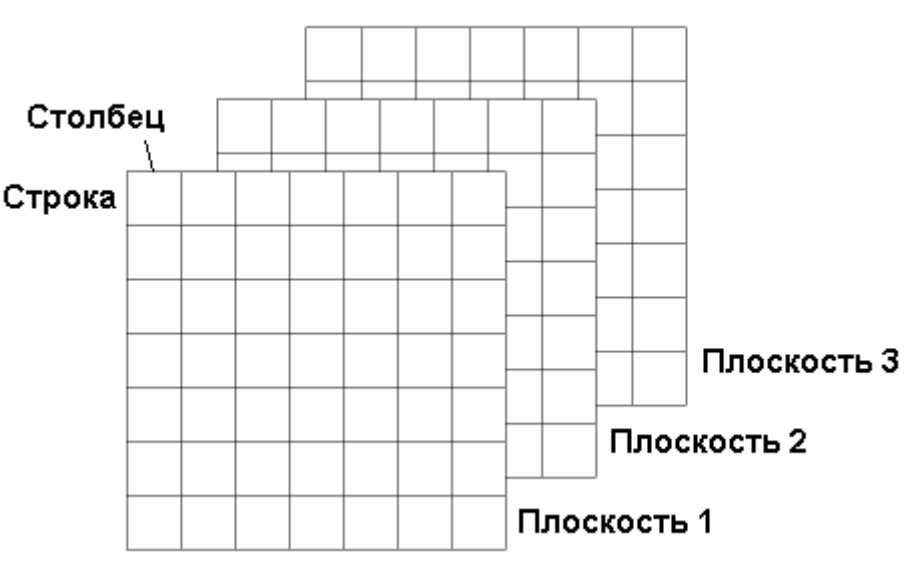


Рис. 3.2. Графическое представление трехмерного массива.

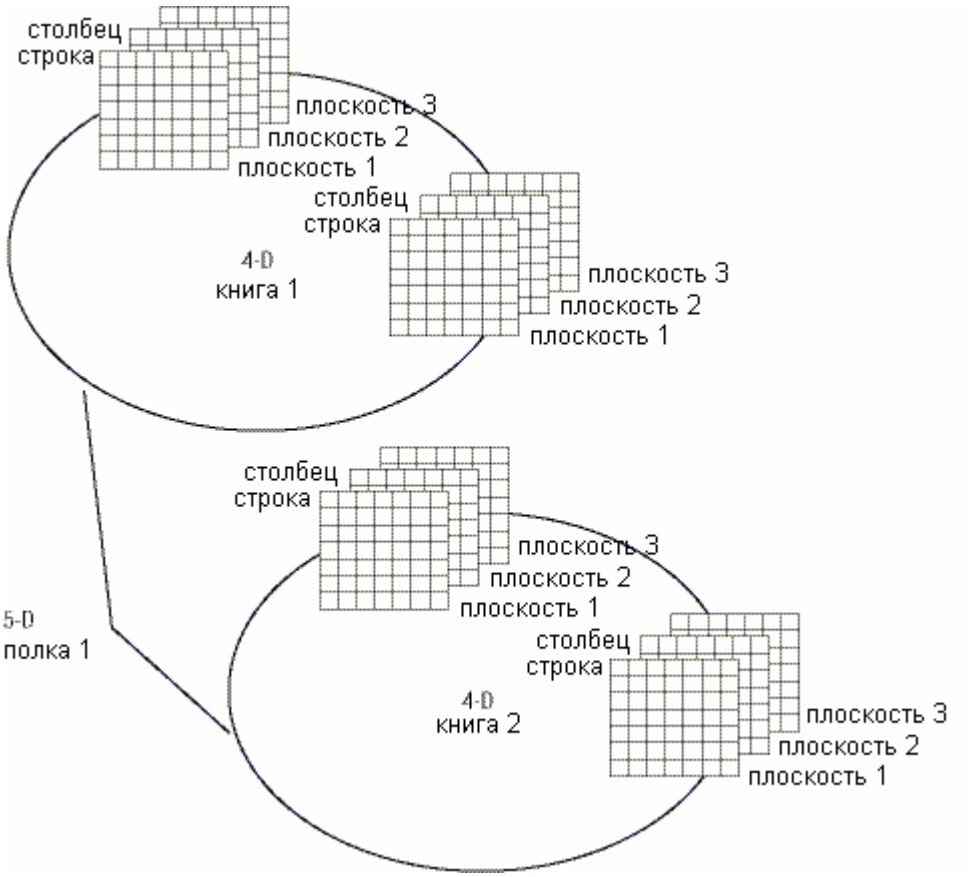


Рис. 3.3. Графическое представление пятимерного массива.

3.10.2. Примеры массивов

Элементы массивов типа ARRAY состоят из дискретных чисел, которые просто размещены в таблице. Рассмотрите следующие примеры.

$$\text{NTEMP} = \begin{bmatrix} -47.6 \\ -5.2 \\ 25.0 \\ 86.5 \\ 107.9 \\ 168.7 \\ 225.0 \end{bmatrix} \qquad \text{EVOLUM} = \begin{bmatrix} 0.025 \\ 0.01 \\ 0.265 \\ 1.00 \\ 0.832 \\ 0.52 \\ 1.032 \\ 0.002 \\ 0.697 \\ 0.01 \end{bmatrix}$$

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

Параметр NTEMP мог быть массивом температур в отобранных узлах; NTEMP(1) = -47.6 мог быть температурой в узле 27, NTEMP(2) = -5.2 мог быть температурой в узле 43, и так далее. Точно так же EVOLUM мог быть массивом объемов элементов, и COMPSTRS мог быть массивом компонент напряжения, с каждым столбцом представляющим отдельное направление (например X, Y, Z, XY, YZ, XZ).

Параметр массива типа CHAR структурирован подобно параметру ARRAY, с табличными значениями, являющимися алфавитно-цифровыми строками символов (до восьми символов). Два примера символьных параметров массива:

3.10.3. Параметр массива типа TABLE

Параметр массива типа TABEL состоит из чисел (алфавитно-цифровые значения не допустимы), размещенные табличным способом, также как тип массива ARRAY. Однако, есть три важных различия.

- ANSYS может вычислить (через линейную интерполяцию) любые значения между явно заявленными значениями элементов массива.
- Массив таблицы содержит нулевую строку и нулевой столбец, используемые для значений индекса доступа к данным, и в отличие от стандартных массивов, эти индексные значения могут быть вещественными числами. Единственное ограничение это то, что индексные значения должны быть возрастающими (никогда не убывающими) числами. Вы должны явно объявить значение индекса доступа к данным для каждой строки и столбца, иначе назначенное значение по умолчанию "очень малое число" (7.888609052E-31).
  Вы можете более удобно определить индексную отправную точку и индексировать значения через команду **`*TAXIS`**.
- Значение индекса плоскости постоянно находится в ячейке 0, 0 для каждой плоскости.

Следующий рисунок показывает массив TABLE со значениями индекса доступа к данным. Обратите внимание, что индексы определены в нулевых столбцах и строках.

Рис. 3.4. Графическое представление табличного массива (TABLE).

Как показано в вышеприведенном примере, конфигурируя массив таблицы, Вы должны установить

- Индекс каждой плоскости указывается в элементе массива с индексом 0,0.
- Столбец доступа к данным индексирует значения в элементах в 0-ых строках в плоскости 1. Обращаясь к данным из массива для всех плоскостей используются только значения индекса столбца из плоскости 1. Устанавливая значения элемента массива, Вы используете традиционный индекс строки и столбца.
- Строка доступа к данным индексирует значения в элементах в 0-ых столбцах в плоскости 1. Обращаясь к данным из массива для всех плоскостей используются только значения индекса строки от плоскости 1. Устанавливая значения элемента массива, Вы используете традиционный индекс строки и столбца.

3.10.4. Определение и листинг массивов

Чтобы определить параметр массива, Вы должны сначала объявить его тип и размерность, используя команду **`*DIM`** (**Utility Menu> Parameters> Array Parameters> Define/Edit**).

Следующие ниже примеры иллюстрируют команду **`*DIM`**, используемую, чтобы определить размерность различных типов массивов:

```
*DIM,AA,,4                ! тип ARRAY по умолчанию, размерность 4[x1x1]
```

```
*DIM,XYZ,ARRAY,12       ! тип ARRAY, размерность 12[x1x1]
*DIM,FORCE,TABLE,5      ! тип TABLE, размерность 5[x1x1]
*DIM,T2,,4,3            ! размерность 4x3[x1]
*DIM,CPARR1,CHAR,5      ! тип CHAR, размерность 5[x1x1]
```

Примечание

Элементы массива для ARRAY и TABLE принимают значение 0 (за исключением 0-ых строк и столбца для TABLE, которые принимают очень малое значение). Элементы массива CHAR принимают пустое значение.

Следующий пример показывает, как заполнить массив 5-мерными данными. Используйте 1-мерные таблицы, чтобы загрузить 5-мерную таблицу. Используйте команду ***TAXIS**, чтобы определить значения индексов таблицы.

```
*dim,xval,array,X1
*dim,yval,array,Y1
yval(1)=0,20
*dim,zval,array,10
zval(1)=10,20,30,40,50,60,70,80,90,100
*dim,tval,array,5
tval(1)=1,.90,.80,.70,.60
*dim,tevl,array,5
tevl(1)=1,1.20,1.30,1.60,1.80

*dim,ccc,tab5,X1,Y1,Z1,D4,D5,X,Y,Z,TIME,TEMP
*taxis,ccc(1,1,1,1,1),1,0,wid                    !!! X-Dim
*taxis,ccc(1,1,1,1,1),2,0,hth                    !!! Y-Dim
*taxis,ccc(1,1,1,1,1),3,1,2,3,4,5,6,7,8,9,10     !!! Z-Dim
*taxis,ccc(1,1,1,1,1),4,0,10,20,30,40            !!! Time
*taxis,ccc(1,1,1,1,1),5,0,50,100,150,200         !!! Temp
*do,ii,1,2
   *do,jj,1,2
      *do,kk,1,10
         *do,ll,1,5
            *do,mm,1,5
ccc(ii,jj,kk,ll,mm)=(xval(ii)+yval(jj)+zval(kk))*tval(ll)*tevl(mm)
            *enddo
         *enddo
      *enddo
   *enddo
*enddo
```

3.10.5. Определение значений параметра массива

Вы можете определить значения массива

- Установив индивидуальные значения массива через команду ***SET** или сокращенно через знак "=".
- Заполнив отдельные векторы (столбцы) в массиве с заданными или с вычисленными значениями (команда ***VFILL**).
- В интерактивном режиме определив значения для элементов через диалоговое окно ***VEDIT**.
- Чтением значений из файла ASCII (команды ***VREAD** или ***TREAD**).

Примечание

Вы не можете создать или редактировать 4-х или 5-мерные массивы в интерактивном режиме. Команды ***VEDIT**, ***VREAD** и ***TREAD** не применимы к ним.

3.10.5.1. Определение отдельных значений массива

Вы можете использовать команду ***SET** или знак "=". Используется также как и для скалярных параметров, за исключением того, что Вы теперь определяете столбец данных (до десяти значений элемента массива в команде "="). Например, чтобы определить параметр XYZ размерностью массива 12x1 (12 строк на 1 столбец), Вам понадобиться использовать дважды команду "=". В следующем примере первая команда определяет первые восемь элементов массива, и вторая команда определяет следующие четыре элемента массива:

```
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
```

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

Обратите внимание, что начальное местоположение элемента массива обозначено номером индекса строки параметра (1 в первой команде и 9 во второй команде).

Следующий пример показывает, как определить значения элемента для параметра массива T2 размерностью 4x3, проставленной ранее в примере использования команды ***DIM**:

```
T2(1,1)=.6,2,-1.8,4          ! эл-ты массива (1,1),(2,1),(3,1),(4,1)
T2(1,2)=7,5,9.1,62.5         ! эл-ты массива (1,2),(2,2),(3,2),(4,2)
T2(1,3)=2E-4,-3.5,22,.01     ! эл-ты массива (1,3),(2,3),(3,3),(4,3)
```

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

Следующий пример определяет значения элемента для параметра FORCE табличного массива (TABLE), обсуждаемого ранее.

```
FORCE(1)=0,560,560,238.5,0
FORCE(1,0)=1E-6,.8,7.2,8.5,9.3
```

Первая команда “=” определяет пять элементов параметра FORCE табличного массива. Вторая команда “=” переопределяет индексы.

$$\text{FORCE} = \begin{array}{c} \\ 1E-6 \\ 0.8 \\ 7.2 \\ 8.5 \\ 9.3 \end{array} \begin{array}{c} 0 \\ \left[\begin{array}{c} 0.0 \\ 560.0 \\ 560.0 \\ 238.5 \\ 0.0 \end{array}\right] \end{array}$$

Символьные параметры массива могут также быть определены, используя команду **"="**. Определяемые значения могут быть до восьми символов каждый и должны быть заключены в одиночные кавычки. Например:

```
*DIM,RESULT,CHAR,3   ! параметр символьного массива размерностью (3,1,1)
RESULT(1)='SX','SY','SZ'  ! присвоение значений параметру RESULT
```

Обратите внимание, что, определяя числовой параметр массива, начальное местоположение элемента массива должно быть определено (в этом случае, обозначен индекс строки номер 1).

Примечание

Массив типа CHAR не может использоваться как символьное имя параметра, потому что это создаст конфликт с меткой CHAR в команде ***DIM**. ANSYS заменит значение строкой символов, назначенным на параметр CHAR, когда CHAR будет введен на третьем поле команды ***DIM**.

3.10.5.2. Заполнение массива векторов

Вы можете использовать команду ***VFILL** (**Utility Menu> Parameters> Array Parameters> Fill**), чтобы "заполнять" массив ARRAY или вектор (столбец) TABLE.

Для более подробной информации о синтаксисе команды ***VFILL** см. справочную систему ANSYS. Следующий пример иллюстрирует возможности команды ***VFILL**.

```
*DIM,DTAB,ARRAY,4,3                  ! размерность числового массива 4x3
*VFILL,DTAB(1,1),DATA,-3,8,-12,57    ! четыре значения данных
                                     ! загружены в вектор 1
*VFILL,DTAB(1,2),RAMP,2.54,2.54   ! заполнение вектора 2 данными начиная
                                  ! со значения 2.54 и с шагом 2.54
*VFILL,DTAB(1,3),RAND,1.5,10 ! заполнение вектора 3 случайными числами
                             ! между 1.5 и 10.  Результат будет
                             ! изменяться из-за случайных чисел.
```

$$\text{DTAB} = \begin{bmatrix} -3 & 2.54 & 2.799901284 \\ 8 & 5.08 & 6.11292418 \\ -12 & 7.62 & 6.70205516 \\ 57 & 10.16 & 4.11487684 \end{bmatrix}$$

3.10.5.3. Редактирование массивов в интерактивном режиме

Команда ***VEDIT** (**Utility Menu> Parameters> Array Parameters> Define/Edit**), доступная только в интерактивном режиме, запускает диалоговое окно ввода данных, которое Вы можете использовать, чтобы редактировать массивы типа ARRAY или TABLE (не CHAR). Диалоговое окно обеспечивает множество удобных возможностей:

- Редактор стиля электронной таблицы для значений элементов массива.
- Навигационные управления для того, чтобы просматривать большие массивы.
- Инициализирующая функция, чтобы установить в любую строку или столбец необходимое значение (только для ARRAY).
- Функции удаления, копирования и вставки для того, чтобы перемещать строки или столбцы данных (только для ARRAY).

Полные инструкции по использованию диалоговых окон доступны при нажатии кнопки «Help» в диалоговом окне.

Примечание

Вы не можете редактировать 4-х или 5-мерные массивы типа ARRAY или TABLE в интерактивном режиме.

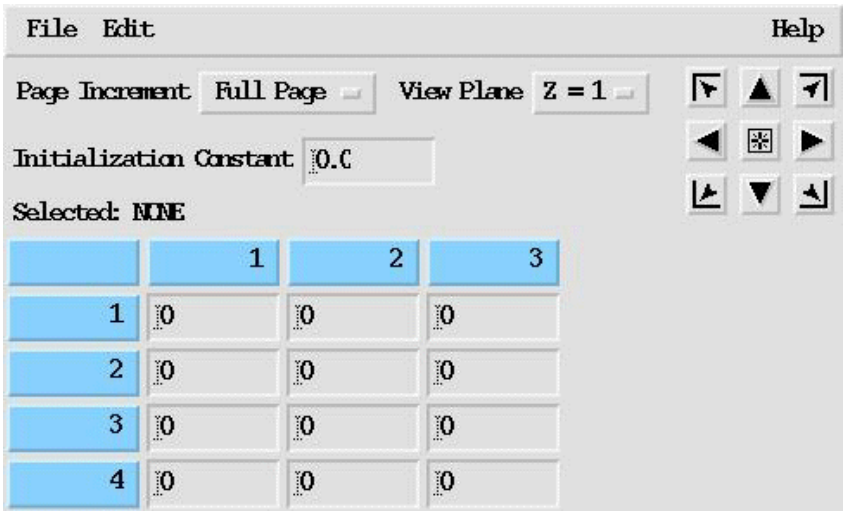


Рис. 3.4. Пример диалогового окна ***VEDIT** для редактирования массива типа ARRAY.

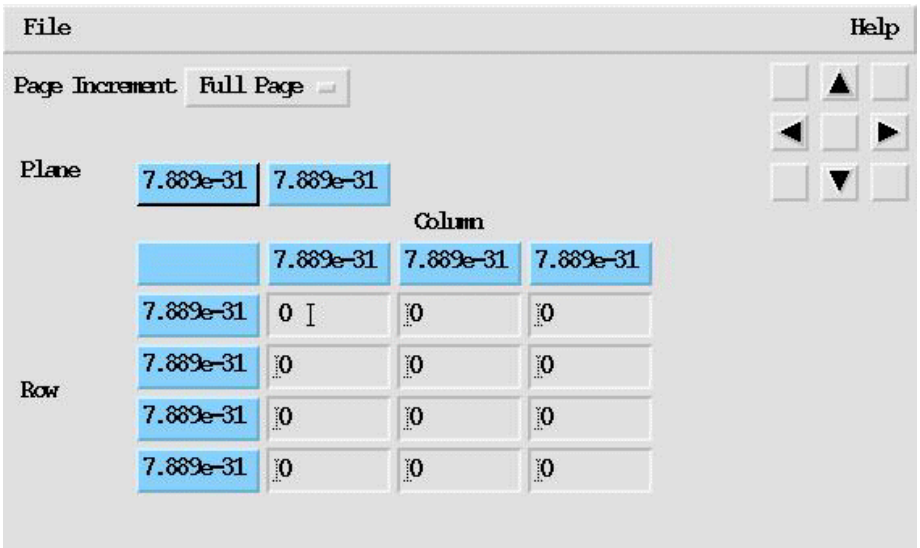


Рис. 3.4. Пример диалогового окна ***VEDIT** для редактирования массива типа TABLE.

3.10.5.4. Заполнение массива из файла данных командой *VREAD

Вы можете заполнить массив из файла данных, используя команду ***VREAD** (**Utility Menu> Parameters> Array Parameters> Read from File**). Команда считывает информацию из файла данных ASCII и начинает записывать их в массив, начиная с индекса, который Вы определили. Вы можете управлять форматом чтения информации из файла до

описателей данных. Описатели данных должны быть заключены в круглые скобки и помещены в строку после команды ***VREAD**. См. раздел «Векторные операции» для получения дополнительной информации об описателях данных. Описатели данных управляют числом полей, которые считываются при каждой записи, шириной полей данных и позицией десятичной точки в поле.

Например дан следующий файл данных с именем dataval:

```
1.5     7.8  12.3
15.6  -45.6  42.5
```

и массив по имени EXAMPLE, который определен, и которому задана размерность 2x3, следующими командами

```
*DIM,EXAMPLE,,2,3
*VREAD,EXAMPLE(1,1),dataval,,,JIK,3,2
(3F6.1)
```

в результате получаем

```
            1.5     7.8  12.3
EXAMPLE=    15.6  -45.6  42.5
```

Команда ***VREAD** не может быть использована непосредственно из окна ввода команд. Однако, **Utility Menu> Parameters> Array Parameters> Read from File** диалоговое окно предлагает способ определить описатели данных и использовать команду в интерактивном режиме.

Примечание

Вы не можете заполнить 4-х или 5-мерный массив, используя команду ***VREAD**.

3.10.5.5. Заполнение табличного массива из файла данных командой *TREAD

После того, как массив сконфигурирован, Вы имеете две опции для того, чтобы определить значения для элементов массива TABLE: Вы можете добавить значения, также как и для любого другого типа массива, или Вы можете считать в таблицу данные из внешнего файла.

Чтобы считать в таблицу данные из внешнего файла, Вы должны во первых определить массив TABLE, задав число строк, столбцов, плоскостей, и меток для каждого. Вы можете тогда считать файл ASCII, содержащий таблицу данных, используя команду ***TREAD** (**Utility Menu> Parameters> Array Parameters> Read from File**). В это время, Вы также определяете число строк, которые необходимо пропустить (NSKIP) между заголовком файла и первой строки таблицы.

Считывая данные из внешнего файла, помните:

- Файл, содержащий таблицу данных может быть создан в текстовом редакторе или внешнем приложении (типа Microsoft Excel), но он должен быть в форме ASCII разграниченный символами табуляции, чтобы читаться в ANSYS.
- Вы должны сначала определить массив в ANSYS, не забывая учитывать индексные значения (0,0).
- Значения читаются прямо поперек строк, пока все столбцы на каждой строке массива не заполнятся; тогда ANSYS переходит от одной строки к следующей и начинает заполнять в ней столбцы, и так далее. Убедитесь, что размерность массива, который Вы определили, правильна. Если Вы по ошибке определите меньше столб-

цов в массиве чем необходимо, то ANSYS начнет заполнять в следующей строке массива, используя значения, остающиеся в первой строке читаемой таблицы данных. Точно так же, если Вы определите больше столбцов в массиве чем необходимый, то ANSYS заполнит все столбцы массива, используя значения от следующей строки считываемой таблицы данных и только тогда перейдет на следующую строку и начнет ее заполнять.

Вы можете создать 1- , 2- , и 3-мерные таблицы, считывая данные из внешнего файла. Примеры того, как их создавать, последуют ниже.

Примечание

Вы не можете заполнить 4-х или 5-мерный массив типа TABLE, используя команду ***TREAD**.

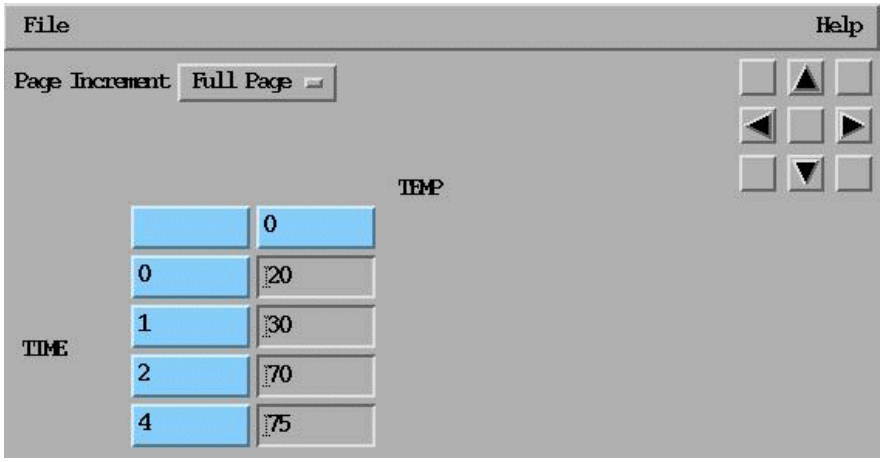
Пример 1: 1-мерная таблица.

Сначала, создайте 1-мерную таблицу, используя приложение по вашему выбору (типа электронной таблицы, редактора текста, и т.д.) и затем сохраните файл как текстовый файл в формате с символами табуляции. В этом примере, таблица называется "Tdata" и содержит данные температуры в зависимости от времени. Эта таблица в форме ASCII выглядела бы следующим образом:

Time Temperature Table

Time	Temp
0	20
1	30
2	70
4	75

В ANSYS Вы определяете параметр TABLE с именем "Tt", используя команду ***DIM** (**Utility Menu> Parameters> Array Parameters> Define/Edit**). Определите 4 строки и 1 столбец, метку строки Time и метку столбца Temp. Обратите внимание, что таблица данных, которую Вы создали, имеет четыре строки и один столбец, плюс строка и значения индекса столбца (первый столбец - TIME - является значениями индекса строки). Тогда массив TABLE считываемый из файла, который был описан ранее, с 2-мя пропущенными строками выглядел бы в ANSYS следующим образом:



Риз. 3.7. Пример диалогового окна для 1-мерной таблицы.

Тот же самый пример, выполненный через командную строку, был бы похож на следующее:

```
*DIM,Tt,table,4,1,1,TIME,TEMP
*TREAD,Tt,tdata,txt,,2
```

Пример 2: 2-мерная таблица

Для этого примера, создайте (в текстовом редакторе или в другом приложении) 2-мерную таблицу с именем "T2data", содержащую температурные данные как функцию времени и x-координаты и считайте его в параметр массива TABLE по имени "Ttx". Эта таблица в форме ASCII выглядела бы следующим образом:

Temp (time-X-coord) Table					
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

В ANSYS, Вы определяете параметр TABLE "Ttx" используя команду **`*DIM`** (**Utility Menu> Parameters> Array Parameters> Define/Edit**). Определите 4 строки, 5 столбцов, 1 плоскость, метку строки Time и метку столбца X-coord. Обратите внимание, что таблица данных, которую Вы создали, имеет четыре строки и пять столбцов данных, плюс строка со значениями индекса столбца. Затем считайте данные из файла как описано ранее, определив 2 пропущенных строки. Этот массив в ANSYS выглядел бы следующим образом:

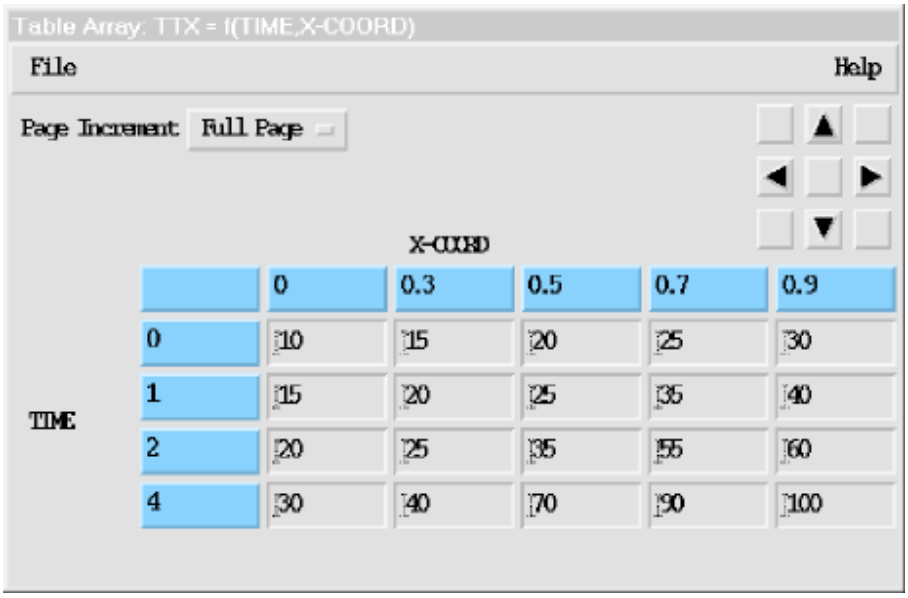


Рис. 3.8. Пример диалогового окна для 2-мерной таблицы.

Тот же самый пример, выполненный через командную строку, был бы похож на следующее:

```
*DIM,Ttx,table,4,5,,time,X-COORD
*TREAD,Ttx,t2data,txt,,2
```

Пример 3: 3-мерная таблица

Для этого примера, создайте 3-мерную таблицу по имени "T3data", содержащий температурные данные как функцию времени, x-координаты и y-координаты и считайте ее в параметр массива TABLE с именем "Ttxy". Таблица в форме ASCII выглядела бы следующим образом:

Temp (time-X-coord) Table					
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

В примере выше, полужирные значения (в позиции (0,0,Z)) указывают отдельные плоскости. Каждая плоскость данных, наряду со строкой и значениями индекса столбца, повторяется для каждой плоскости. Отличаются только значение индекса плоскости и фактические значения данных. Затененная область показывает значения, которые изменяются от плоскости к плоскости.

В ANSYS, Вы определяете параметр TABLE "Ttxy" используя команду **`*DIM`** (**Utility Menu> Parameters> Array Parameters> Define/Edit**). В случае 3-мерной таблицы, размерность таблицы проставляется согласно числу строк, столбцов, и плоскостей данных. Первый столбец (Time) - значения индекса строки, и первая строка - значения индекса столбцов. Определите 4 строки, 5 столбцов, 2 плоскости, метку строки Time, метку столбца X-COORD и метку плоскости Y-COORD. Обратите внимание, что таблица данных, которую Вы создали, имеет четыре строки и пять столбцов данных в двух плоскостях, плюс строка и столбец со значениями индексов. Затем считайте из файла данные как было описано ранее, пропустив 2 строки. Этот массив типа TABLE для второй плоскости данных (Y=1.5) будет выглядеть в ANSYS так, как показано ниже:

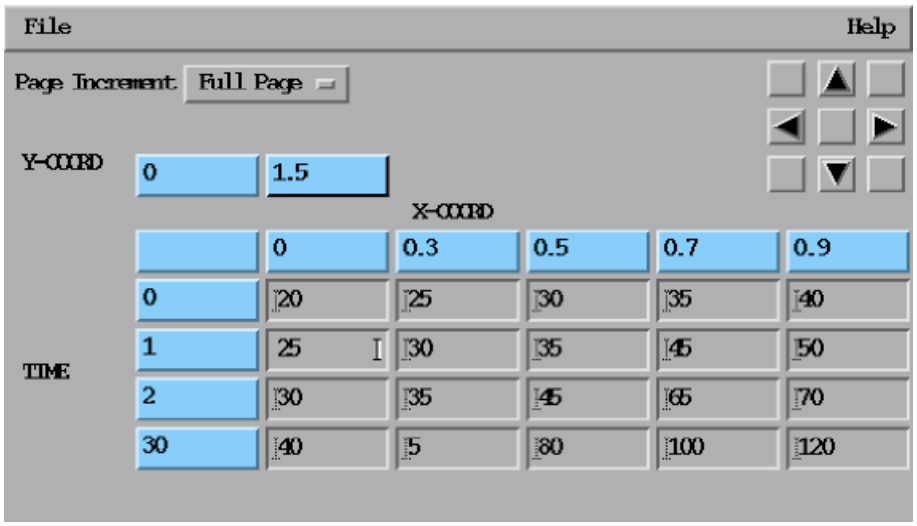


Рис. 3.9. Пример диалогового окна для 3-мерной таблицы.

Тот же самый пример, выполненный через командную строку, был бы похож на следующее:

```
*DIM,Ttxy,table,4,5,2,TIME,X-COORD,Y-COORD
*TREAD,Ttxy,t3data,txt,,2
```

3.10.5.6. Интерполяция значений

Когда Вы обращаетесь к информации из массива, ANSYS будет интерполировать значения между явно установленными.

Как пример того, как ANSYS интерполирует значения в массиве типа TABLE, рассмотрите следующее:

$$A = \begin{array}{c} \\ 1.0 \\ 2.0 \\ 3.0 \end{array} \begin{array}{c} 1.0 \\ \left[\begin{array}{c} 12.0 \\ 28.0 \\ 146.4 \end{array}\right] \end{array} \qquad PQ = \begin{array}{c} \\ 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{array} \begin{array}{c} \begin{array}{cc} 1.0 & 2.0 \end{array} \\ \left[\begin{array}{cc} 2.8 & 4.2 \\ -9.6 & -12.3 \\ 42.0 & 9.7 \\ -4.5 & 2.0 \end{array}\right] \end{array}$$

Учитывая, что A - параметр массива TABLE, программа ANSYS может вычислить любое значение между (1) и (2), например

- (1.5) вычисляется как 20.0 (половина между 12.0 и 28.0)
- (1.75) вычисляется как 24.0
- (1.9) вычисляется как 26.4

Точно так же, если PQ - параметр массива TABLE

- PQ (1.5, 1) вычисляется как 3.4 (половина между 2.8 и -9.6)
- PQ (1, 1.5) вычисляется как 3.5 (половина между 2.8 и 4.2)
- PQ (3.5, 1.3) вычисляется как 14.88

Эта особенность позволяет Вам описывать функцию, типа y=f(x), используя массив типа TABLE. Вы использовали бы j=0 столбец для значений независимой переменной x и j=1 столбца для значений y. Рассмотрите, например, функцию истории нагружения силой описанной пятью точками как показано ниже.

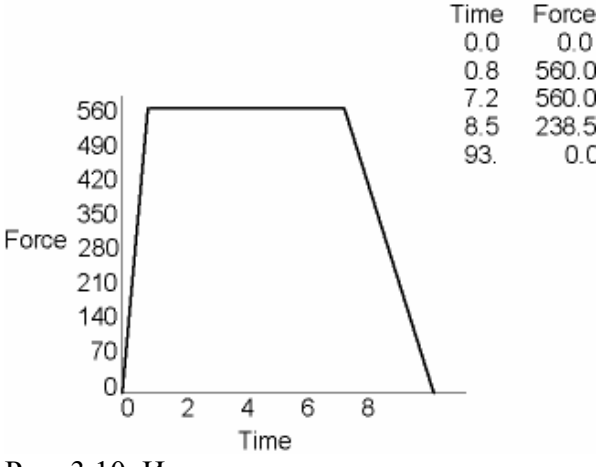


Рис. 3.10. История нагружения.

Вы можете определить эту функцию как параметр массива TABLE, элементы массива которого - значения силы, и с индексом строки от 1 до 5 - значения времени от 0.0 до 9.3. Схематично параметр будет выглядеть следующим образом:

$$\text{FORCE} = \begin{array}{c} \\ 1E-6 \\ 0.8 \\ 7.2 \\ 8.5 \\ 9.3 \end{array} \begin{array}{c} 0 \\ \left[\begin{array}{c} 0.0 \\ 560.0 \\ 560.0 \\ 238.5 \\ 0.0 \end{array}\right. \end{array}$$

ANSYS может вычислить (через линейную интерполяцию) значения силы в зависимости от времени не определенные в параметре FORCE. Для вышеупомянутого примера, ANSYS вычислит значение 89.4375 для FORCE(9). Если местоположение параметра находиться вне размерности массива, то экстраполяция значений не выполняется, а используется начальное/последнее значение массива. Например, ANSYS определит значение 560.0 для FORCE(5,2) или 0.0 для FORCE(12)

Вы можете увидеть из этих примеров, что параметры массива TABLE могут стать очень мощными инструментами в вашем анализе. Обычно этот тип массива применяется для описания функции истории нагружения, спектра кривых отклика, кривых напряжения-деформации, кривых температурнозависисмых материалов, кривых B-H для магнитных материалов, и т.д. Знайте, что параметры массива TABLE требуют больше компьютерного времени для обработки, чем тип массива ARRAY.

3.10.5.7. Поиск и восстановление значений в массиве

Вы можете использовать команду **`*VGET`** (**Utility Menu> Parameters> Get Array Data**), которая подобно команде **`*GET`**, отыскивает значения и сохраняет их в массиве.

Вы должны определить начальный номер местоположения массива для параметра массива, который создает команда **`*VGET`**. Цикл продолжается по последовательным числам до значения *KLOOP* по умолчанию. Например, **`*VGET`**,A(1),ELEM,5,CENT,X возвращает положение центра тяжести по оси X элемента номер 5 и хранит результат в первой ячейке массива A. Поиск продолжается с элементами 6, 7, и так далее пока последовательные ячейки массива не заполняться. В этом примере, если *KLOOP* = 4, то будет возвращен центр тяжести по осям X, Y, и Z.

Чтобы восстанавливать значения параметра массива, используйте команду **`*VPUT`** (**Utility Menu> Parameters> Array Operations> Put Array Data**).

Команда **`*VPUT`** использует те же самые параметры что и команда **`*VGET`** (описанная выше), но делает противоположные операции. Список допустимых значений аргументов команды **`*VPUT`** см. в описании команды в справочной системе ANSYS.

Программа ANSYS "помещает" векторные элементы без какого-либо преобразования системы координат. **`*VPUT`** может заменить существующие элементы массива, но не может создать новые элементы. Степень свободы, которая заменена в базе данных, доступна для всех последующих операций. Другие результаты изменяются временно, и доступны главным образом для немедленной последующей печати и отображения на экране.

Примечание

Используйте эту команду с большой предосторожностью, поскольку она может изменить все разделы базы данных. Команда **`*VPUT`** не поддерживает все значения аргументов, которые доступны для команды **`*VGET`**, потому что помещение значений в некоторые местоположения может сделать базу данных ANSYS противоречивой.

3.10.5.8. Листинг массива

Как и со скалярными параметрами, Вы можете использовать команду ***STATUS**, чтобы выполнить листинг массива. Следующие примеры иллюстрируют использование команды ***STATUS**:

```
*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      ! листинг параметра XYZ с 5 по 9 строку
 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    ! листинг параметра FORCE, включая столбец j=0

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)         ! листинг параметра 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)       ! листинг параметра RESULT

PARAMETER STATUS- RESULT      ( 4 PARAMETERS DEFINED)

LOCATION      VALUE
1      1      1      SX(CHAR)
2      1      1      SY(CHAR)
3      1      1      SZ(CHAR)
```

3.10.6. Запись файла данных

Вы можете записать отформатированные файлы данных (табличное форматирование) из данных, содержащихся в массивах, через команду ***VWRITE**. Команда берет до 10 векторов массива как параметры и записывает данные, содержавшиеся в тех векторах в открытый в настоящее время файл (команда ***CFOPEN**). Формат для каждого вектора определен согласно описателей данных ФОРТРАНА 77 на строке после команды ***VWRITE** (поэтому, Вы не можете использовать команду ***VWRITE** из командной строки ANSYS.)

Вектор массива определяется начальным местоположением элемента (типа MYARRAY (1,2,1)). Вы можете также использовать выражение, которое вычисляет значение для каждого поля в каждой строке файла данных. Ключевое слово SEQU определяет последовательный ряд целых чисел для столбца, начинающихся с одного.

Формат каждой строки в файле данных определяется по дескриптору (описателю) строки. Вы должны включить один описатель для каждого параметра к команде. Не включайте слова FORMAT в дескриптор строки. Вы можете использовать любой вещественный формат, или символьный формат дескриптора; однако, Вы не можете использовать целое число или list directed дескрипторы.

3.10.6.1. Формат описателей данных

Если Вы не знакомы с описателями данных ФОРТРАНА, то этот раздел даст Вам начальные представления о форматирований ваших файлов данных. Для получения дополнительной информации обратитесь к документации ФОРТРАНА 77.

Вы должны обеспечить описатель данных для каждого элемента данных, который Вы определяете как параметр в команде ***VWRITE**. Вообще, Вы можете использовать описатель F (с плавающей запятой) для любых числовых значений. Дескриптор F имеет синтаксис

Fw.d

где **w** – является шириной поля данных в символах.
d – является числом цифр справа от десятичной точки.

Таким образом, для поля, которое в ширину состоит из 10 символов и имеет восемь символов после десятичной точки, Вы использовали бы следующий описатель данных:

```
F10.8
```

Для символьных полей, Вы можете использовать дескриптор A. Описатель имеет синтаксис

Aw

где **w** – Является шириной поля данных в символах.

Таким образом, для символьного поля, которое в ширину состоит из 8 символов, описатель выглядел бы следующим образом:

```
A8
```

Следующие примеры иллюстрируют использование команды ***VWRITE** и описатели данных.

Дан массив MYDATA с определенной размерностью и заполненый следующими значениями:

$$\text{MYDATA} = \begin{bmatrix} 2.15215183 & 3.89075020 & 5.28636971 & 7.15706483 & 13.7859423 & 87.4970443 \\ 2.30485343 & 4.44486730 & 5.40919563 & 7.68192625 & 15.5483820 & 86.5677915 \\ 2.01051819 & 3.39152436 & 5.93663807 & 7.38584253 & 18.4635868 & 45.7263566 \\ 2.36833012 & 3.32711472 & 5.63220341 & 7.22482004 & 18.7977889 & 39.7902425 \\ 2.84819512 & 4.76350638 & 5.97802354 & 7.29258882 & 14.8096356 & 62.0843906 \\ 2.22795343 & 3.48214546 & 5.54685145 & 7.90325139 & 14.0708891 & 37.6009897 \end{bmatrix}$$

Следующий короткий макрос во-первых определяет скалярный параметр X равный 25 и затем открывает файл **vector** (команда ***CFOPEN**). Затем команда ***VWRITE** определяет данные, которые будут записаны в файл. В этом случае, первый вектор записывается с использованием ключевого слова SEQU, чтобы создать номера строк. Обратите внимание, что в некоторых случаях константы, скалярные параметры, и операции, которые включают значения элемента массива, записываются в файл. Обратите внимание на эти элементы содержащиеся в файле данных.

```
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
```

Макрос создает следующую картотеку данных:

```
 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
```

Второй пример использует массив заранее определенной размерности и заполненный следующим образом:

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

Обратите внимание на использование описателей в следующем примере использования команды ***VWRITE**:

```
*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,' ?')
```

Получающийся файл данных

```
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 ?
```

3.10.7. Операции над массивами

Так же как параметрические выражения и функции позволяют выполнять операции над скалярными параметрами, так и имеется ряд команд для выполнения операции над массивами. существуют следующие классы операций: операции над столбцами (векторами), известные как *векторные операции* и операции над всей матрицей (массивов), известные как *матричные операции*. Все операции выполняются рядом команд, описание которых будет рассмотрено ниже.

3.10.7.1. Векторные операции

Векторные операции это ряд простых операций таких как сложение, вычитание, синус, косинус, скалярное произведение, векторное произведение, и т.д. - повторяющиеся последовательно над элементами массива. Циклы (будут обсуждаться ниже), могут использоваться с этой целью, но более удобный и намного более быстрый путь состоит в том, чтобы использовать векторные команды - ***VOPER**, ***VFUN**, ***VSCFUN**, ***VITRP**, ***VFILL**, ***VREAD**, и ***VGET**. Из этих перечисленных векторных команд, только ***VREAD** и ***VWRITE** допустимы для символьных параметров массива. Другие команды векторных операций применяются только, если массив определен как тип ARRAY или тип TABLE (команда ***DIM**).

***VFILL**, ***VREAD**, ***VGET**, ***VWRITE**, и ***DIM** команды были введены ранее в этой главе. Другие команды, которые обсуждаются в этом разделе, включают

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

Выполняет операцию над двумя векторами на входе и в результате на выходе получается один вектор.

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

На входе выполняет функцию над одним вектором массива и в результате выводит один вектор массива.

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

На входе определяет свойства одного вектора массива и помещает результат в указанный скалярный параметр.

***VITRP Utility Menu> Parameters> Array Operations> VectorInterpolate**

Формирует параметр массива (типа ARRAY), интерполируя параметр массива (типа TABLE) по указанным индексам таблицы.

Ниже следуют примеры, которые иллюстрируют использование некоторых из этих команд. Обратитесь к справочной системе ANSYS для информации о синтаксисе этих команд. Для всех следующих примеров, параметры массива (типа ARRAY) X, Y, и THETA, были определены размерности и значения элементов массивов.

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

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

В следующем примере для результирующего массива (Z1) сначала определена размерность. Затем команда ***VOPER** складывает 2-ой столбец массива X и 1-ый столбец массива Y, оба стартуют с первой строки, и затем помещает результат в массив Z1. Обратите внимание, что стартовое местоположение (индекс строки и столбца) должно быть определено для всех параметров массива. Затем операция просчитывает последовательно вниз все элементы указанного вектора.

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

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

В следующем примере снова для результирующего массива (Z2) сначала определена размерность. Затем команда ***VOPER** умножает первый столбец X (начинающийся со строки 2) с четвертым столбцом Y (начинающийся со строки 1) и записывает результаты в массив Z2 (начинающийся со строки 1).

```
*DIM,Z2,ARRAY,3
*VOPER,Z2(1),X(2,1),MULT,Y(1,4)
```

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

В этом примере для результирующего массива (Z4) сначала определена размерность. Затем команда ***VOPER** вычисляет векторное произведение четырех пар векторов, одна пара для каждой строки X и Y. i, j, и k компоненты этих векторов - столбцы 1, 2, и 3 соответственно из массива X и столбцов 2, 3, и 4 из Y. Результаты записаны в массив Z4, у которого компоненты i, j, и k - соответственно векторы 1, 2, и 3.

```
*DIM,Z4,ARRAY,4,3
```

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

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

В следующем примере для массив результатов (A3) сначала определена размерность. Затем команда ***VFUN** возводит каждый элемент в векторе 2 из массива X в степень 2 и записывает результаты в массив A3.

```
*DIM,A3,ARRAY,4
*VFUN,A3(1),PWR,X(1,2),2
```

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

В этом примере, сначала проставлена размерность массива результатов (A4). Затем две команды ***VFUN** вычисляют косинус и синус элементов массива в массив THETA и помещают результаты в первый и второй столбцы массива A4 соответственно. Обратите внимание, что массив A4 теперь представляет дугу окружности, охватывающую 90 °, описанную семью точками (чьи x, y, и z глобальные декартовские координаты представляют собой три вектора). Дуга имеет радиус 1.0 и располагается параллельно плоскости x-y с координатой z = 2.0.

```
*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
```

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

Две дополнительных команды ***VOPER**, используют операции сборки (GATH) и разборки (SCAT), чтобы скопировать значения из одного вектора в другой основываясь на индексах элементов массивов. Следующий пример демонстрирует операцию сборки. Обратите внимание, что, как всегда, для массива результатов должна быть заранее определена размерность. В примере операция сборки копирует значение массива B1 в массив B3 (используя индексные позиции, определенные в B2). Также, обратите внимание на то, что последний элемент в массиве B3 равен 0, поскольку это его инициализированное (начальное значение по умолчанию) значение.

```
*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)
```

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

3.10.7.3. Операции с матрицами

Матричные операции это математические операции между числовыми массивами, типа матричного умножения, транспонирования и решения системы уравнений.

Команды, обсуждаемые в этом разделе, следующие:

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

Выполняет матричные операции над двумя входными параметрами массивов (матрицами) и на выходе получает одну матрицу. Матричные операции включают:

- Матричное умножение
- Решение системы уравнений
- Сортировка (в порядке возрастания) на указанном векторе в матрице
- Ковариация между двумя векторами
- Корреляция между двумя векторами

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

Копирует или транспонирует матрицу массива (принимает одну входную матрицу и производит одну матрицу на выходе).

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

Вычисляет коэффициенты или оценивает ряд Фурье.

Ниже следующие примеры иллюстрируют использование некоторых из этих команд. Обратитесь к справочной системе ANSYS для информации о синтаксисе этих команд.

Этот пример показывает возможности сортировки с помощью команды **`*MOPER`**. Для этого примера, предположим, что для массива (SORTDATA) была определена размерность и значения его элементов следующим образом:

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

Сначала для массива OLDORDER определяется размерность. Затем команда **`*MOPER`** разместит первоначальную нумерацию расположения строк в массив

OLDORDER. Далее команда ***MOPER** сортирует строки в массиве SORTDATA так, чтобы вектор 1,1 был теперь в порядке возрастания.

```
*dim,oldorder,,5
*moper,oldorder(1),sortdata(1,1),sort,sortdata(1,1)
```

Следующий массив оценивает следствие *MOPER команды:

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

Вернуть массив SORTDATA в его первоначальный порядок, Вы можете использовав следующую команду:

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

В следующем примере, команда ***MOPER** решает систему уравнений. Для следующих двух массивов были заранее определены размерности и значения их элементов:

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

Команда ***MOPER** может решить систему уравнений для квадратной матрицы. Уравнения принимают форму

$$\mathbf{a_{n1}X_1 + a_{n2}X_2 + , \ldots , + a_{nn}X_n = b_n}$$

В случае вышеупомянутых массивов, команда ***MOPER** решит следующий набор совместных уравнений:

$$\mathbf{2X_1 + 4X_2 + 3X_3 + 2X_4 = 2}$$
$$\mathbf{3X_1 + 6X_2 + 5X_3 + 2X_4 = 2}$$
$$\mathbf{2X_1 + 5X_2 + 2X_3 - 3X_4 = 3}$$
$$\mathbf{4X_1 + 5X_2 + 14X_3 + 14X_4 = 11}$$

Чтобы решить уравнения, сначала для массива результатов (C) определяем размерность. Затем команда ***MOPER** решает уравнения, используя матрицу A как коэффициенты и B как вектор значений.

```
*DIM,C,,4
*MOPER,C(1),A(1,1),SOLV,B(1)
```

Массив C теперь содержит следующие решения.

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

Следующий пример показывает использование команды ***MFUN**, чтобы транспонировать данные в массиве. Для этого примера предполагается, что для массива (DATA) была определена размерность и заполнены его элементы следующими значениями:

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

Как обычно, для массива результатов (DATATRAN) проставляем размерность, затем команда ***MFUN** транспонирует значения и записывает их в массив DATATRAN.

```
*DIM,DATATRAN,,2,3
*MFUN,DATATRAN(1,1),TRAN,DATA(1,1)
```

Ниже показаны результаты в массиве DATATRAN:

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

3.10.7.3. Перечень команд для операций с векторами и матрицами

Все векторные и матричные команды затрагивают установки следующих команд: ***VCUM**, ***VABS**, ***VFACT**, ***VLEN**, ***VCOL**, ***VMASK**. (Из всех перечисленных команд только ***VLEN** и ***VMASK**, вместе с ***VREAD** или ***VWRITE**, являются допустимыми для символьных параметров массива.) Вы можете проверить состояние этих команд с помощью команды ***VSTAT**. Большинство этих команд (и их пути GUI) были описаны ранее в этой главе. Другие – объясняются ниже.

За исключением команды ***VSTAT**, к которой Вы не можете обратиться непосредственно из GUI, все перечисленные команды доступны через **Utility Menu> Parameters> Array Operations> Operation Settings.**

Важно: Все команды перечня сбрасываются к их настройкам по умолчанию после каждой векторной или матричной операции.

***VCUM**

Определяет, будут ли результаты кумулятивными или некумулятивными (перезапись предыдущих результатов). *ParR*, результат векторной операции, или добавляется к существующему параметру того же самого имени или записывается поверх. Значение по умолчанию - некумулятивные результаты, то есть, *ParR* записывается поверх существующего параметра с тем же самым именем.

***VABS**

Применяет вычисление абсолютного значение к любым из параметров, вовлеченных в векторную операцию. Значение по умолчанию – использовать вещественное (алгебраическое) значение.

***VFACT**

Применяет масштабный коэффициент к любым из параметров, вовлеченных в векторную операцию. Значение масштабного коэффициента по умолчанию - 1.0 (полное значение).

***VCOL**

Определяет число столбцов в матричных операциях. Значение по умолчанию должно заполнить все ячейки результирующего массива от указанного стартового местоположения.

***VSTAT**

Перечисляет текущие настройки для параметров массива.

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

Определяет число строк, которые используются в операциях параметра массива.

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

Определяет параметр массива как маскирующий вектор.

Следующая таблица перечисляет различные команды перечня, а также векторные и матричные команды массива, которые они затрагивают.

	***VABS**	***VFACT**	***VCUM**	***VCOL**	***VLENNROW,NINC**		***VMASK**
***MFOURI**	No	No	No	No	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

Ниже следующие примеры иллюстрируют использование некоторых из этих команд. Обратитесь к справочной системе ANSYS для информации о синтаксисе этих команд.

В следующем примере проставлены размеры массива результатов (CMPR). Затем две команды ***VFUN**, вместе с предшествующими командами ***VMASK** и ***VLEN**, сжимают отобранные данные и записывают их в указанные местоположения в массив CMPR. В команде ***VFUN** в дополнение к операции COMP используется операция EXPA.

```
*DIM,CMPR,ARRAY,4,4
*VLEN,4,2    ! Выполняется следующая *V-команда
             ! для каждой четвертой строки,
             ! пропуская каждую вторую
*VFUN,CMPR(1,2),COMP,Y(1,1)
```

```
*VMASK,X(1,3)      ! используется столбец 3 массива X
                   ! как маска для следующей *V-команды
*VFUN,CMPR(1,3),COMP,Y(1,2)
```

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

Этот пример использует ***VFACT** команду для округления значений в векторе массива до некоторого количества десятинных знаков, определенных скалярным параметром NUMDP (равен 2 в данном примере). Для массива NUMDATA была определена размерность и значения его элементов:

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

```
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)
```

или, Вы можете использовать немного более короткую версию

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

Тогда массив NUMDATA будет выглядеть так:

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

Этот пример использует команды ***VLEN** и ***VMASK**, чтобы найти набор простых чисел меньше чем 100. Массив, MASKVECT, создан, используя 1.0, чтобы указать, что значение строки – простое число и 0.0, чтобы указать, что значение не является простым. Алгоритм, используемый, чтобы создать вектор маски должен инициализировать все строки, значение которых больше чем 1 к 1.0 и затем цикл через диапазон возможных факторов, исключая все однотипные факторы. Команда ***VLEN** устанавливает приращение строки для того, чтобы выполнить операции с FACTOR. Когда команда ***VFILL** обработана, номер строки увеличивается на это значение. Поскольку стартовая строка - FACTOR x 2, строки обрабатываются каждым циклом в следующей манере: FACTOR x 2, ФАКТОР x 3, ФАКТОР x 4, и т.д.

```
*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
```

Результат действия команды ***STATUS**, отображает первые 10 элементов в массиве NUMBERS:

```
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
```

3.10.8. Построение графиков векторных массивов

Вы можете графически отобразить значения вектора массива, используя команду ***VPLOT**.

Следующий ниже пример демонстрирует некоторые из возможностей команды ***VPLOT**. Для этого примера, два массива типа TABLE (TABLEVAL и TABLE2) и один числовой массив были определены и заполнены следующими значениями:

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

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

Ниже представлен результат применения команды ***VPLOT**. Обратите внимание, что, так как данные массива типа ARRAY неупорядочиваются, то график представлен как гистограмма; данные массива типа TABLE упорядочиваются и поэтому график представлен как кривая.

Для построения графика применяется следующая команда.

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

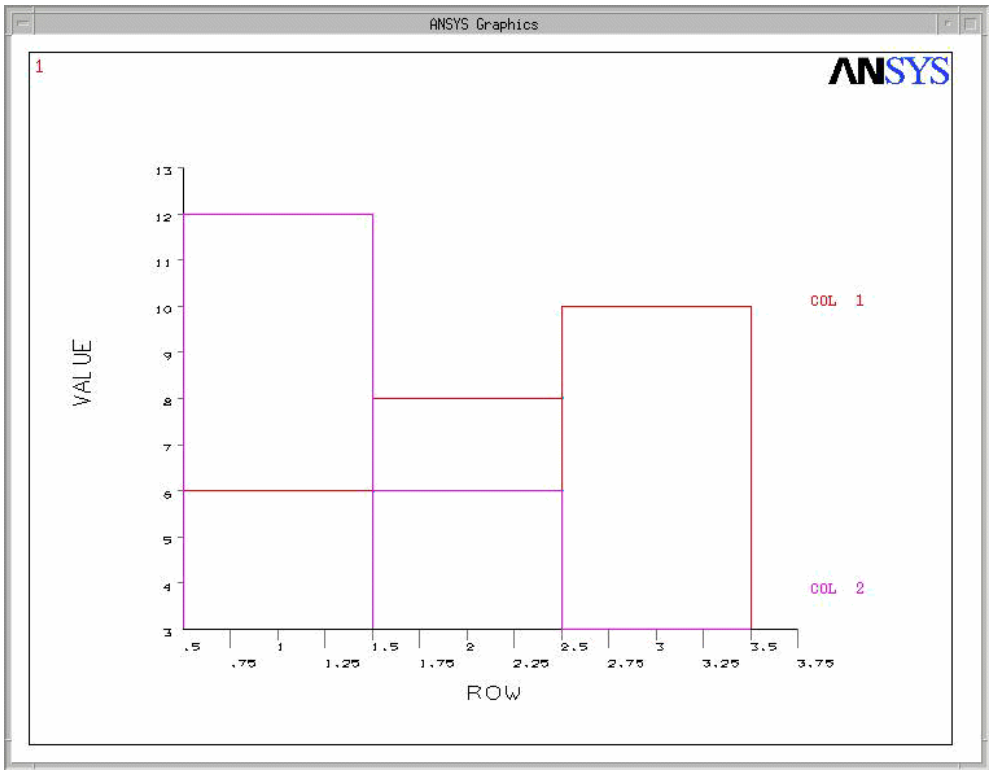


Рис. 3.11. Пример построения графика.

Для построения графика применяется следующая команда.

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

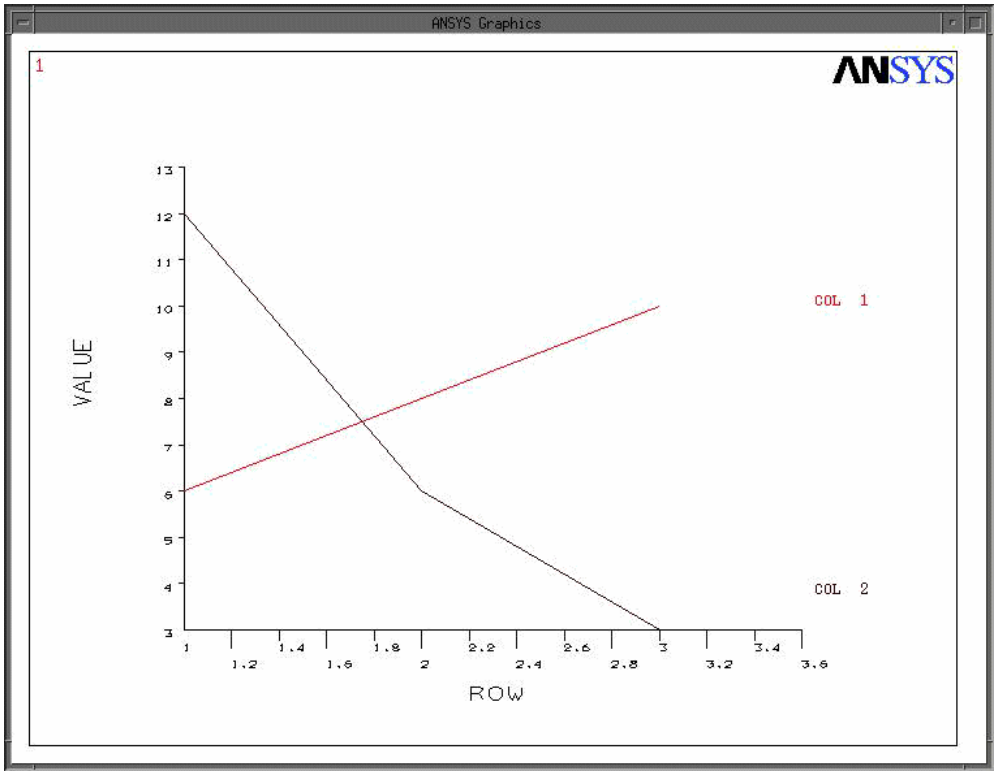


Рис. 3.12. Пример построения графика.

Для построения графика применяется следующая команда.

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

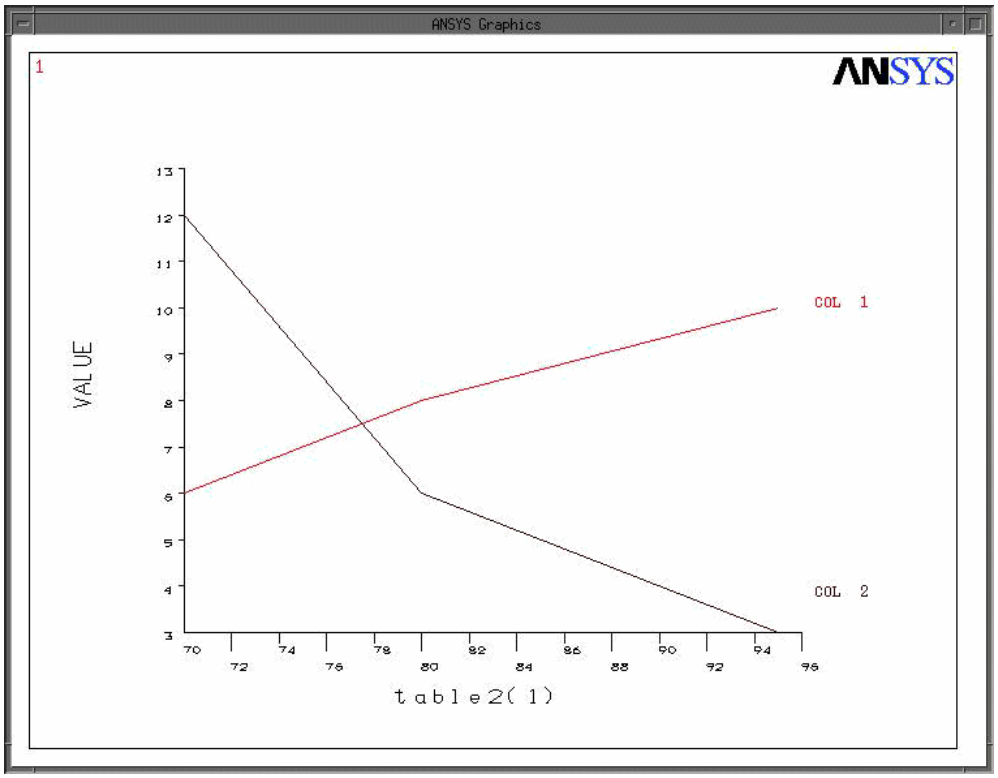


Рис. 3.13. Пример построения графика.

График (ниже) следовавший следующая команда.

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

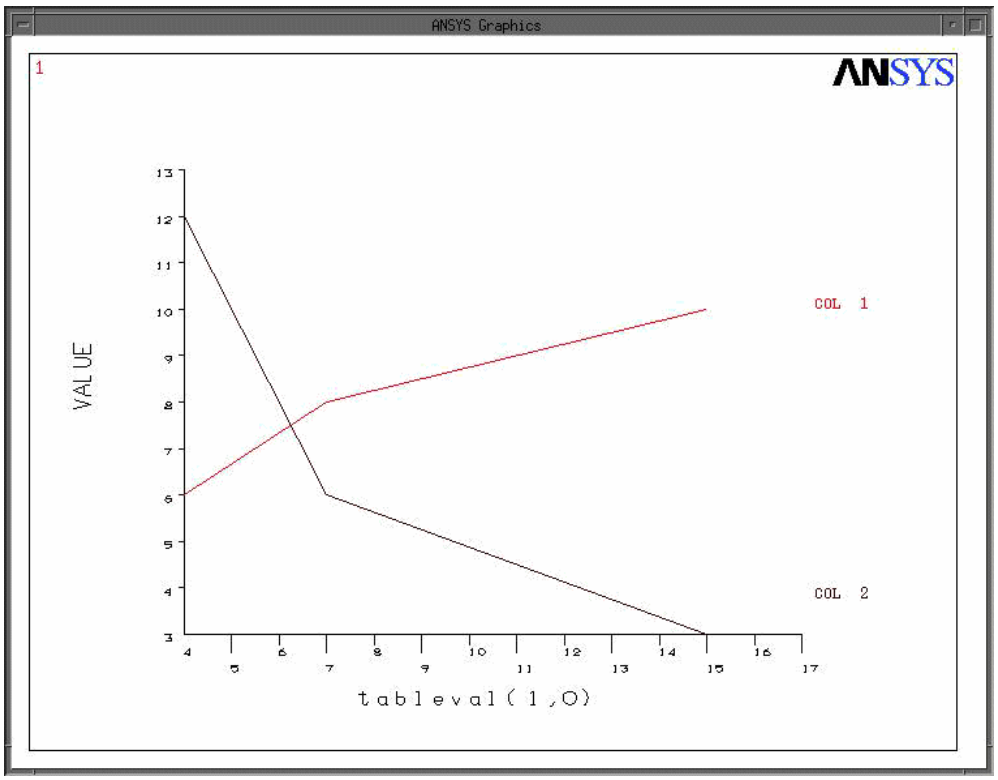


Рис. 3.14. Пример построения графика.

3.10.9. Изменение меток кривой

Когда Вы используете ***VPLOT**, чтобы создать ваши кривые, используются метки по умолчанию. Обычно, метка для кривой 1 – «COL 1», метка для кривой 2 – «COL 2» и так далее; номер столбца это поле, содержащее зависимые переменные для этой конкретной кривой. Вы можете использовать команду **/GCOLUMN**, чтобы применить ваши собственные метки к кривым (любая строка до восьми символов).

Ниже приводится пример использования команды **/GCOLUMN** в начале программы, чтобы применить метки “string01” и “string02” к кривой массива.

```
/gcol,1,string01
/gcol,2,string02

*dim,xxx,array,10
*dim,yyy,array,10,2

xxx( 1,1) =1e6
xxx( 2,1) = 1e6 + 1e5
xxx( 3,1) = 1e6 + 2e5
xxx( 4,1) = 1e6 + 3e5
xxx( 5,1) = 1e6 + 4e5
xxx( 6,1) = 1e6 + 5e5
xxx( 7,1) = 1e6 + 6e5
xxx( 8,1) = 1e6 + 7e5
xxx( 9,1) = 1e6 + 8e5
xxx(10,1) = 1e6 + 9e5

yyy( 1,1) = 1
yyy( 2,1) = 4
yyy( 3,1) = 9
yyy( 4,1) = 16
yyy( 5,1) = 25
yyy( 6,1) = 36
yyy( 7,1) = 49
yyy( 8,1) = 64
yyy( 9,1) = 81
yyy(10,1) = 100

yyy( 1,2) = 1
yyy( 2,2) = 2
yyy( 3,2) = 3
yyy( 4,2) = 4
yyy( 5,2) = 5
yyy( 6,2) = 6
yyy( 7,2) = 7
yyy( 8,2) = 8
yyy( 9,2) = 9
yyy(10,2) = 10

*vplo,xxx(1,1), yyy(1,1) ,2
```

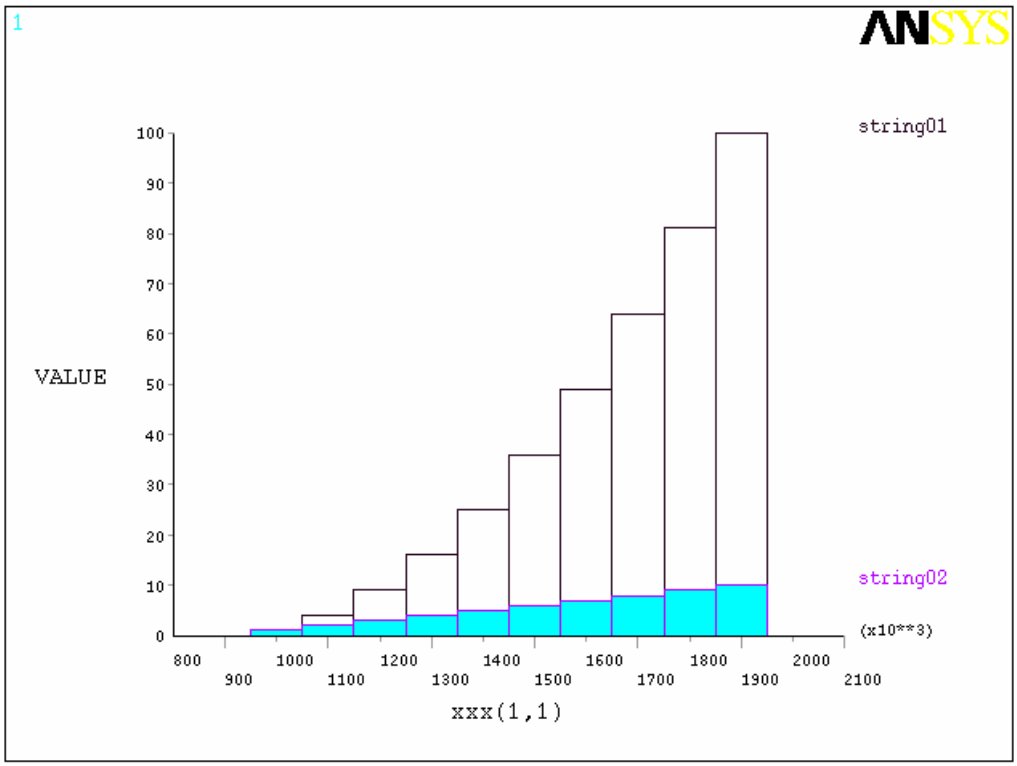


Рис. 3.15. Пример построения графика с пользовательскими метками.

Метки могут быть возвращены к значениям по умолчанию (COL 1 и COL 2), используя команду **/GCOLUMN** без аргумента *STRING*.

```
/gcol,1
/gcol,2
```

4. APDL как макроязык

Вы можете сделать запись часто используемой последовательности команд ANSYS в макрофайле (их иногда называют командными файлами). Создание макроса, в действительности, подобно созданию вашей собственной команды ANSYS. Например, вычисляя потери мощности в магнитном анализе, поток данных должен был бы пройти через серию команд ANSYS в постпроцессоре. Сделав запись этого набора команд в макросе, Вы получите новую отдельную команду, которая выполняет все команды, требуемые для этого вычисления. В дополнение к выполнению серии команд ANSYS, макрос может вызвать функции GUI или передать значения в параметры.

Вы можете также использовать вложенные макросы. Таким образом, один макрос может вызвать второй макрос, второй макрос может вызвать третий макрос, и так далее. Вы можете использовать до 20 уровней вложения, включая любые смены файла, вызванные командой **/INPUT**. После того, как каждый вложенный макрос выполнится, управление потоком данных возвращается программой ANSYS к предыдущему макро-уровню.

Ниже следует очень простой пример макрофайла. В этом примере, макрос создает призму с размерами 4, 3 и 2, и сферу с радиусом 1. Затем выполняется вычитание сферы из одного угла призмы.

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

Если бы этот макрос называли `mymacro.mac`, то Вы могли бы выполнить эту последовательность команд используя следующую единственную команду ANSYS

```
*use,mymacro
```

или просто (потому что расширение `.mac`)

```
mymacro
```

Хотя это и не реальный макрос, однако он иллюстрирует принцип использования макросов.

Эта глава предоставляет информацию относительно различных способов, которыми Вы можете создавать, хранить и выполнять макрос. В ней также обсуждается основная информация о том как использовать APDL в создании макросов.

Далее рассматриваются следующие разделы:

- Создание макроса
- Выполнение макросов и макробиблиотек
- Локальные переменные
- Управление процессом выполнения программы в APDL
- Справочник функций управления
- Использование параметров _STATUS и _RETURN в макросах
- Использование макросов с отдельными компонентами и сборками
- Примеры макросов

4.1. Создание макроса

Вы можете создать макрос непосредственно в ANSYS или с использованием вашего текстового редактора (типа emacs, vi, или wordpad). Если ваш макрос довольно прост и короток, то создать его в ANSYS может быть более удобным. Если Вы будете создавать большой и более сложный макрос или редактируете существующий макрос, то Вам понадобиться текстовый редактор. Кроме того, текстовый редактор позволит Вам использовать похожие макросы или log-файл ANSYS как источник для вашего нового макроса.

Для любого большого и сложного макроса Вы должны всегда рассматривать похожие макросы или выполнение задачи в интерактивном режиме в ANSYS и использование получающегося log-файла как подоснову для нового макроса. Любой из этих методов может значительно сократить время и усилия, требуемые для создания необходимого макроса.

Далее будут рассмотрены следующие темы по созданию макросов:

- Соглашение об именах макросов
- Путь поиска макрофайлов
- Создание макросов в среде ANSYS
- Создание макросов в текстовом редакторе
- Использование библиотек макросов

4.1.1. Соглашение об именах макросов

Макрос - это последовательность команд ANSYS, сохраненных в файле. Макрос не должен иметь того же самого имени как и существующие команды ANSYS, или начинаться с первых четырех символов команд ANSYS, потому что ANSYS выполнит внут-

реннюю команду вместо макроса. Следующие ограничения имен относятся к макрофайлам:

- Имя файла не может превышать 32 символа.
- Имя файла не может начаться с цифры.
- Расширение файла не может содержать больше чем восемь символов (если Вы выполняете макрос как команду ANSYS, то расширение должно быть `.mac`.)
- Имя файла или расширение не могут содержать пробелов.
- Имя файла или расширение не могут содержать никаких символов запрещенных вашей файловой системой и, для совместимости, не должны содержать никаких символов запрещенных UNIX или Windows.

Для гарантии того, что Вы не используете имя команды ANSYS, перед запуском макроса наберите имя файла, которое Вы желаете использовать как команду ANSYS, в командной строке и запустите на выполнение как обычную команду. Если ANSYS выведет сообщение, которое показано ниже, то Вы будете знать точно, что такая команда не используется в текущем процессоре. Вы должны проверить имя макрофайла в каждом процессоре, в котором Вы планируете использовать макрос. (Вы можете также проверить соответствует ли имя макрофайла какой-нибудь команде, перечисленной в справочной документации ANSYS; однако, этот метод не может определить имен недокументированных команд.)

Рис. 4.1. Окно сообщения ANSYS о неизвестной команде.

Использование расширения `.mac` позволяет ANSYS выполнять макрос как любую другую внутреннюю команду. Вы должны избегать использовать расширения `.MAC`, потому что оно используется для внутренних макросов ANSYS.

4.1.2. Путь поиска макрофайлов

По умолчанию ANSYS ищет пользовательский макрофайл (с расширением `.mac`) в следующих местоположениях:

1. `.../ansys_inc/v100/ansys/apdl`.
2. Каталог (или каталоги) определяемый переменной окружения `ANSYS_MACROLIB` (если определена) или регистрационный (домашний) каталог. Эта переменная окружения зарегистрирована в главе «The ANSYS Environment» в «ANSYS Operations Guide».
3. Каталог, определяемый переменной окружения `$HOME`.
4. Текущий рабочий каталог.

Вы можете поместить макрос для вашего личного использования в вашем основном каталоге. Макрос, который должен быть доступным параллельно другим пользовате-

лям, должен быть помещен в каталог .../ansys_inc/v100/ansys/apdl или некоторый другой общедоступный каталог, на который каждый может сослаться через переменную окружающей среды ANSYS_MACROLIB.

Для пользователей Windows: "текущий каталог" - основной каталог (обычно сетевой ресурс) установленный администраторами, и Вы должны спросить у вашего сетевого администратора его местоположение. Вы можете использовать переменные окружающей среды, чтобы создать локальный "домашний каталог". Локальный домашний каталог проверяется после основного каталога, определяемого в вашем профиле домена.

4.1.3. Создание макросов в среде ANSYS

Вы можете создать макрос четырьмя методами в среде ANSYS:

- Выполните команду ***CREATE** в input window. Значения параметров не возвращаются и в файл записываются имена параметров.
- Используйте команды ***CFOPEN**, ***CFWRITE,** и ***CFCLOS**. Параметры определены их текущими значениями и эти значения записываются в макрофайл.
- Выполните команду **/TEE** в input window. Эта команда пишет список команд в файл в то же самое время, когда команды выполняются. Поскольку команды выполняются в текущем сеансе ANSYS, всем именам параметров возвращаются их текущие значения. Однако, в файл, который создан, записываются имена параметров, а не их текущие значения.
- Выберите пункт меню **Utility Menu> Macro> Create Macro**. Этот метод открывает диалоговое окно, которое может использоваться как простой многострочный редактор для того, чтобы создать макрос. Значения параметров не возвращаются и в файл записываются имена параметров.

Следующие разделы детализируют каждый из этих методов.

4.1.3.1. Использование команды *CREATE

Выполнение команды ***CREATE** переадресовывает введенные в input window команды к файлу, определяемому командой. Все команды переадресуются пока Вы не выполните команду ***END**. Если существующий файл имеет то же самое имя как и имя макрофайла, который Вы создаете, то программа ANSYS запишет его поверх существующего файла.

Например, предположим, что Вы хотите создать макрос, с именем matprop.mac, который автоматически определяет ряд свойств материала. Набор команд, водимых в input window для этого макроса, мог бы выглядеть следующим образом:

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

Команда ***CREATE** берет из аргументов имя файла, расширение файла, и путь каталога (в этом случае, каталог макроса определен).

Используя ***CREATE**, все параметры, используемые в командах, записываются файл (не заменяются назначаемыми в данный момент значениями для параметров).

Вы не можете использовать ***CREATE** в пределах цикла типа DO.

4.1.3.2. Использование команды *CFWRITE

Если Вы желаете создать макрофайл, в котором заменяются значения параметров текущими значениями, Вы можете использовать ***CFWRITE**. В отличие от ***CREATE**, команда ***CFWRITE** не может определять имя макроса; Вы должны сначала определить макрофайл командой ***CFOPEN**. Только те команды ANSYS, которые явно предварены командой ***CFWRITE**, записываются в определяемый файл; все другие команды, введенные в input window выполняются. Как и с командой ***CREATE**, ***CFOPEN** может определить имя файла, расширение файла, и путь. Следующий пример пишет команду **BLOCK** в текущий открытый макрофайл.

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

Обратите внимание, что использовались параметры для аргументов команды **BLOCK**. Текущее значение этих параметров (а не имена параметров) записываются в файл. Так, для этого примера, могла бы быть строка, записанная в макрофайл

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

Чтобы закрыть макрофайл, выполните команду ***CFCLOS**.

Примечание

В то время как имеется возможность создавать макрос с помощью этого метода, эти команды (***CFOPEN**, ***CFWRITE,** и ***CFCLOS**) являются наиболее полезными для того, чтобы записывать команды ANSYS в файл во время выполнения макроса.

4.1.3.3. Использование команды /TEE

Выполнение команды **/TEE**,NEW или **/TEE**, APPEND переадресовывает команды ANSYS, введенные в input window к файлу, определяемому этой же командой, одновременно с выполнением этих команд. Все команды выполняются и переадресуются, пока Вы не выполните команду **/TEE**,END. Если существующий файл имеет то же самое имя как и имя макрофайла, который Вы определяете командой, программа ANSYS записывает макрофайл поверх существующего файла. Чтобы этого избежать, используйте вместо **/TEE**,NEW команду **/TEE**, APPEND.

В дополнение к аргументу Label (который может иметь значение NEW, APPEND, или END), команда **/TEE** использует аргументы имени файла, расширения файла и пути каталога.

Поскольку команды выполняются в текущем сеансе ANSYS, всем именам параметров возвращаются их текущие значения. Однако, в файле, который создается, записываются имена параметров (не заменяются назначаемыми в настоящее время значениями параметров). Если ваши текущие значения параметров важны, Вы можете сохранить их в файл, используя команду **PARSAV**.

4.1.3.4. Использование меню Utility Menu> Macro> Create Macro

Выбор этого пункта меню открывает диалоговое окно ANSYS, которое Вы можете использовать как простой редактор для того, чтобы создать макрос. Вы не можете открыть и редактировать существующий макрос этим средством; если Вы будете использовать имя существующего макроса как аргументы для полей команды ***CREATE**, то существующий файл будет перезаписан.

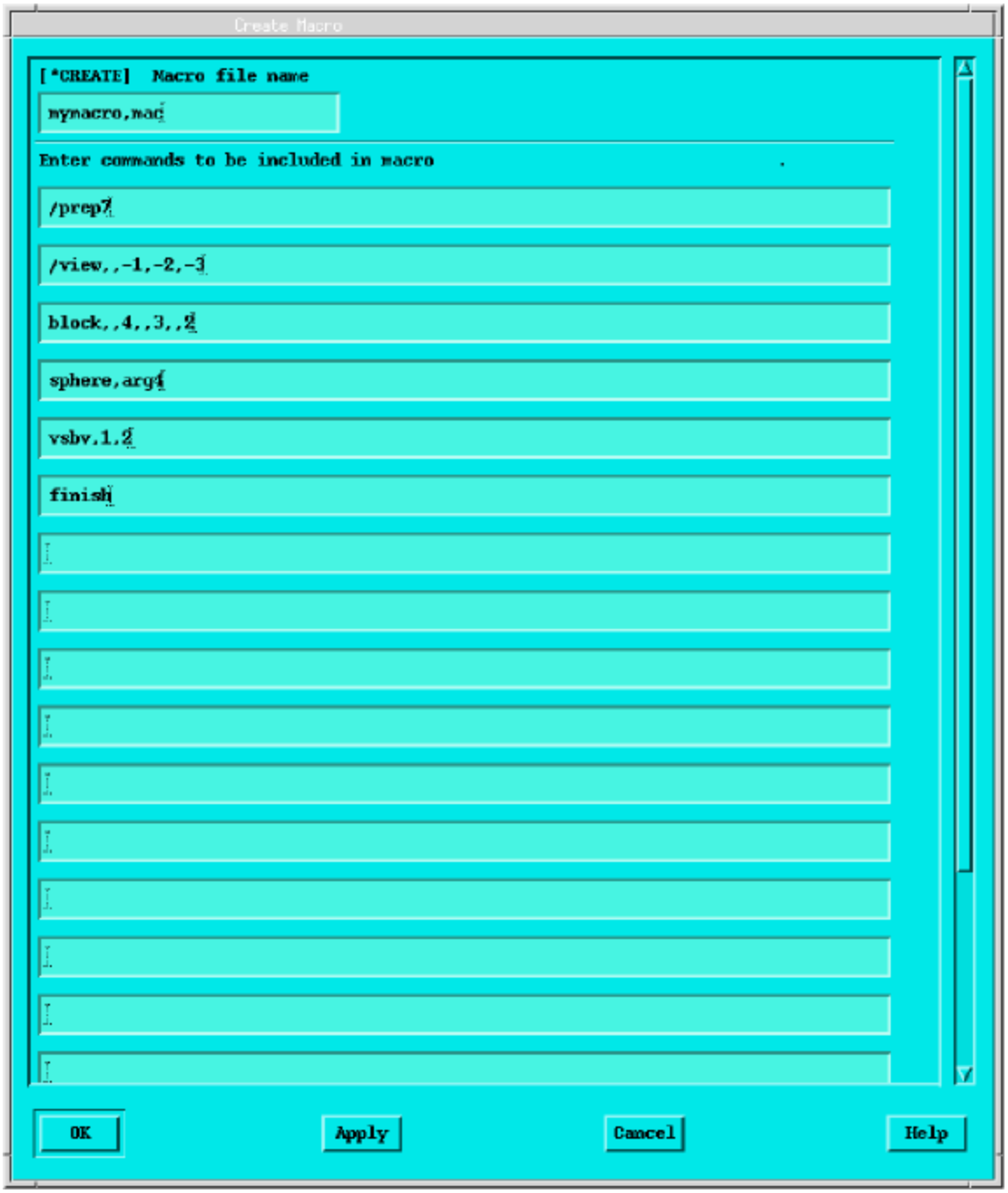


Рис. 4.2. Диалоговое окно создания макроса.

Как и с командой ***CREATE**, параметры не вычисляются, но записываются буквально в макрофайл. Обратите внимание, что Вы не делаете последнюю строку с командой ***END**.

4.1.4. Создание макроса в текстовом редакторе

Вы можете использовать ваш любимый текстовый редактор, чтобы создавать или редактировать макрофайлы. Будет работать любой редактор ASCII. Кроме того, макрос ANSYS можно закончить завершающей строкой с управляющим обозначением UNIX или Windows, (возврат каретки, парного перевода строки или просто перевода строк), таким образом Вы сможете создать макрос на одной платформе и использовать его на других платформах.

Если Вы используете этот метод, чтобы создать макрос, не включайте команды ***CREATE** и ***END**.

Рис. 4.3. Создание макроса в текстовом редакторе.

4.1.5. Использование библиотек макросов

Для большего удобства ANSYS позволяет Вам помещать ряд макросов в отдельный файл, называемый библиотекой макросов. Вы можете создать его через команду ***CREATE** или в текстовом редакторе. Учитывая, что макробиблиотеки имеют тенденцию быть более длинными (многострочными), чем отдельный макрос, использование текстового редактора обычно обеспечивает лучший результат по их созданию.

Библиотеки макросов не имеют никакого явного расширения файла и подчиняются тем же самым соглашениям по именам как и макрофайлы. Макрофайл библиотеки имеет следующую структуру:

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

Например, следующий макрофайл содержит два простых макроса:

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

Обратите внимание, что каждый макрос предварен именем макрофайла (иногда передаваемым именем блока данных), и оканчивается командой **/EOF**.

Макрофайл библиотеки может постоянно находиться где-нибудь на вашей системе, хотя для удобства Вы должны поместить его в пределах пути поиска макрофайлов. В отличие от обычных макрофайлов, макрофайл библиотеки может иметь любое расширение до восьми символов.

4.2. Выполнение макросов и макробиблиотек

Вы можете выполнить любой макрофайл, используя команду ***USE**. Например, чтобы выполнить макрос MYMACRO (без расширения) постоянно находящийся где-нибудь в пределах пути поиска макрофайлов, Вы использовали бы

```
*use,mymacro
```

В этом случае для выполнения макроса не требуется никаких дополнительных аргументов. Если бы вместо этого макрос назывался MYMACRO.MACRO и он постоянно находился в /myaccount/macros, то Вы могли бы вызвать его следующим образом

```
*use,/myaccount/macros/mymacro.macro
```

Обратите внимание, что команда ***USE** позволяет Вам вводить путь и расширение наряду с именем файла и что они не вводятся как отдельные аргументы команды.

Если макрос имеет расширение файла .mac и постоянно находится в пределах пути поиска файлов, Вы можете выполнить его, как будто это команда ANSYS, просто введя его имя в командную строку. Например, чтобы вызвать mymacro.mac Вы можете просто ввести

```
mymacro
```

Вы можете также выполнить макрос с расширением .mac через пункт меню **Utility Menu> Macro> Execute Macro**.

Если тот же самый макрос использует аргументы (см. раздел 4.3.1. для получения дополнительной информации о передаче аргументов к макросу), то они могут быть введены в командную строку следующим образом

```
mymacro,4,3,2,1.5
```

или

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

Диалоговое окно пункта меню **Utility Menu> Macro> Execute Macro** имеет поля для аргументов.

Выполнение макроса, содержавшегося в макробиблиотеках, делается аналогичным образом. Вы должны сначала определить файл библиотеки, используя команду ***ULIB**. Например, чтобы определить, что макрос находится в файле mymacros.mlib, который постоянно находится в каталоге /myaccount/macros, Вы выполнили бы следующую команду:

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

После выбора макробиблиотеки, Вы можете выполнить любой макрос, содержавшийся в библиотеке, определяя его через команду ***USE**. Как и с макросом, содержавшимся в индивидуальном файле, Вы можете определить аргументы как параметры в команде ***USE**.

Примечание

После выполнения команды ***ULIB** Вы не можете использовать команду ***USE**, чтобы обратиться к макросу, не содержавшемуся в указанном макрофайле библиотеки.

4.3. Локальные переменные

APDL обеспечивает два набора специально именованных скалярных параметров, которые являются доступными для использования как локальные переменные. Они состоят из

- Набора скалярных параметров, которые обеспечивают способ передачи аргументов из командной строки к макросу.
- Набора скалярных параметров, которые могут использоваться в пределах макроса. Они обеспечивают набор локальных переменных, которые могут использоваться, чтобы определять значения только в пределах этого макроса.

Следующие разделы рассматривают оба этих набора параметров более подробно.

4.3.1. Передача аргументов в макрос

Есть 19 скалярных параметров, которые Вы можете использовать, чтобы передать аргументы из командной строки в макрос. Эти скалярные параметры могут многократно использоваться с множеством макросов; то есть, их значения являются локальными к каждому макросу. Параметры имеют имена от ARG1 до AR19 и они могут использоваться для любого из следующих элементов:

- Чисел
- Алфавитно-цифровых строк символов (до восьми символов заключенных в одиночные кавычки)
- Числовых или символьных параметров
- Параметрических выражений

Примечание

Вы можете передать только значения аргументов от ARG1 до AR18 к макросу как параметры с командой ***USE**. Если Вы создаете макрос, который может использоваться как команда ANSYS (макрофайлы имеют расширение `.mac`), то Вы можете передать к макросу значения аргументов от ARG1 до AR19.

Например, следующий простой макрос требует четырех аргументов – *ARG1*, *ARG2*, *ARG3*, и *ARG4*:

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

Чтобы выполнить этот макрос, пользователь мог бы ввести

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

4.3.2. Локальные переменные в пределах макроса

Каждый макрос может иметь до 79 скалярных параметров, используемых как локальные переменные (от AR20 до AR99). Эти параметры являются полностью локальными по отношению к отдельному макросу и к множественным макросам, при этом каждый из них может иметь собственные уникальные значения, определенные на эти параметры. Эти параметры не передаются к макросу, вызванному из макроса (вложенный макрос).

Они передаются к любым файлам, обработанным через команду **/INPUT**, или в цикл, выполняющийся в пределах этого макроса.

4.3.3. Локальные переменные вне макроса

ANSYS также имеет подобный набор скалярных параметров от ARG1 до AR99, которые являются локальными к входному файлу, и не передаются ни к какому макросу, вызванному этим входным файлом. Таким образом, после завершения макроса выполнение команд возвращается к входному файлу и значения от ARG1 до ARG99 возвращаются к любым значениям, которые были определены в пределах входного файла.

4.4. Управление процессом выполнения программы в APDL

Выполняя входной файл, ANSYS обычно ограничивается линейным процессом выполнения программы; то есть, каждая команда выполняется последовательно в определенном во входном файле порядке. Однако, APDL обеспечивает богатый набор команд, которые Вы можете использовать для управления процессом выполнения программы.

- Вызов подпрограммы (вложенного макроса).
- Безусловный переход к указанному местоположению с макросом.
- Переход, основанный на условии, к указанному местоположению в пределах макроса.
- Повтор выполнения отдельной команды, с приращением значения одного или более параметров команды.
- Зацикливание (повторение) фрагмента макроса указанное количество раз.

Следующие разделы детализируют каждую из этих возможностей управления процессом выполнения программы.

- Вложенные макросы: выполнение подпрограмм в пределах макроса
- Безусловный переход: Goto
- Условный переход: команда *IF
- Повторение команды
- Циклы: Do-Loops
- Неявные циклы Do Loops
- Дополнительный цикл: Do-While

4.4.1. Вложенные макросы: выполнение подпрограмм в пределах макроса

APDL позволяет Вам создавать вложенные макросы до 20 уровня вложенности. Вы можете передать до 19 аргументов к макросу и, в конце каждого вложенного макроса, выполнение программы возвращается к уровню, который вызвал этот макрос. Например, следующий простой макрофайл библиотеки демонстрирует макрос `MYSTART`, который вызывает макрос `MYSPHERE`, чтобы создать сферу.

```
mystart
/prep7
/view,,-1,-2,-3
mysphere,1.2
finish
/eof
mysphere
sphere,arg1
/eof
```

4.4.2. Безусловный переход: Goto

Самая простая команда, выполняющая переход, это команда ***GO**, инструктирует программу перейти к указанной метке, не выполняя никаких промежуточных команд. Процесс выполнения программы продолжается от указанной метки. Например

```
*GO,:BRANCH1
---    ! этот блок команд пропускается (не выполняется)
---
:BRANCH1
---
---
```

Метка, определенная командой ***GO** должна начаться с двоеточия (:) и не должна содержать больше чем восемь символов, включая двоеточие. Метка может находиться в любом месте файла.

Примечание

Использование команды ***GO** теперь считают устаревшим и ее использование не рекомендуется. Для лучшего управления процессом выполнения программы см. другие команды выполняющие переход.

4.4.3. Условный переход: команда *IF

APDL позволяет Вам выполнять один из ряда альтернативных блоков, основанных на проверке условия. Условия проверяются путем сравнения двух числовых значений (или параметров, которые возвращают числовые значения).

Команда ***IF** имеет следующий синтаксис

***IF**, VAL1, Oper, VAL2, Base

где

- `VAL1` - первое числовое значение (или числовой параметр) для сравнения.
- `Oper` - оператор сравнения.
- `VAL2` - второе числовое значение (или числовой параметр) для сравнения.
- `Base` - действие, которое происходит, если условие выполняется.

APDL предлагает восемь операторов сравнения, которые обсуждаются подробно в справочной информации команды ***IF**. Кратко они следующие:

EQ – равно (для `VAL1` = `VAL2`).
NE – не равно (для `VAL1` ≠ `VAL2`).
LT – меньше чем (для `VAL1` < `VAL2`).
GT – больше чем (для `VAL1` > `VAL2`).
LE – меньше чем или равный (для `VAL1` ≤ `VAL2`).
GE – больше чем или равный (для `VAL1` ≥ `VAL2`).
ABLT – абсолютные значения `VAL1` и `VAL2` перед выполнением операции <.
ABGT – абсолютные значения `VAL1` и `VAL2` перед выполнением операции >.

Давая аргументу `Base` значение THEN, команда ***IF** становится началом конструкции "если-тогда-иначе". Конструкция состоит из

- Команды ***IF**, сопровождаемой
- Одной или более дополнительными командами ***ELSEIF**
- Дополнительной командой ***ELSE**
- Необходимой командой ***ENDIF**, отмечающей конец конструкции.

В ее самой простой форме, команда ***IF** выполняет сравнение и, если это сравнение истинно, то выполняется переход к метке, определенной в аргументе Base. (В комбинации, ряд таких команд ***IF** может функционировать подобно команде CASE в других языках программирования.) Чтобы не выполнить переход к метке используются конструкции «if-then-else» или «do-loop». Если поток данных выходит в конец файла из-за ложного условия ***IF**, то сеанс ANSYS не будет закончен как обычно. Вы будете должны закончить его извне (используйте в UNIX "kill" функцию или менеджера задачи Windows).

Устанавливая значение STOP в аргумент Base, Вы можете выйти из ANSYS основываясь на специальном условии.

Конструкция «if-then-else» просто проверяет условие и выполняет следующий блок или переходит к следующей команде после команды ***ENDIF** (показано комментарием "Continue").

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

Следующий пример демонстрирует более сложную структуру. Обратите внимание, что только один блок может быть выполнен. Если ни одно из условий не имеет значение «истина», то выполняется блок после команды ***ELSE**.

```
*IF,A,EQ,1,THEN
    ! Block1
    .
    .
    *ELSEIF,A,EQ,0
     ! Block2
     .
     .
     *ELSEIF,A,EQ,-1
     ! Block3
     .
     .
     *ELSE
     ! Block4
     .
     .
*ENDIF
! Continue
```

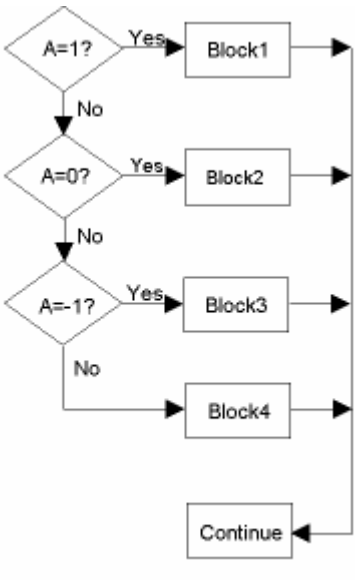


Рис. 4.4. пример конструкции «if-then-else».

Прмечание

Вы можете использовать команду **/CLEAR** в пределах конструкции «if-then-else». Команда **/CLEAR** не очищает ***IF** стек и число ***IF** уровней сохраняется. Командой ***ENDIF** необходимо закрывать любую выполняющую переход логику. Кроме того, имейте в виду, что команда **/CLEAR** удаляет все параметры, включая любые параметры, которые используются в ваших командах, выполняющих переход. Вы можете избежать любых проблем, которые могли бы явиться результатом удаления параметров, использовав команду **PARSAV** перед **/CLEAR**, и затем после **/CLEAR** выполнить команду **PARRES**.

4.4.4. Повторение команды

Самая простая возможность зацикливания это команда ***REPEAT**, которая позволяет Вам выполнять предыдущую команду указанное количество раз, увеличивая любое поле в той команде на постоянное значение. В следующем примере

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

команда создает один элемент между узлами 1 и 2, а следующая команда ***REPEAT** определяет, что команда **E** выполняется в общей сложности пять раз (включая первоначальную команду **E**), увеличивая номер второго узла на единицу при каждом дополнительном выполнении. Результат - пять конечных элементов связанных по узлам 1-2, 1-3, 1-4, 1-5, и 1-6.

Примечание

Большинство команд, которые начинаются со слэша (/) или звездочки (*), так же как и макросы, выполненные как "неизвестные команды," не могут быть повторены. Однако, графические команды, которые начинаются со слэша, могут повторяться. Кроме того, избегайте использовать команду ***REPEAT** с интерактивными командами, типа тех, которые требуют интерактивного выбора объектов или тех, которые требуют ответа пользователя.

4.4.5. Циклы: Do-Loops

Цикл позволяет Вам повторять ряд команд указанное количество раз. Команды ***DO** и ***ENDDO** отмечают начало и конец цикла. Команда ***DO** имеет следующий синтаксис:

Следующий пример цикла редактирует пять файлов шагов нагружения (нумерованные от 1 до 5), и делает одни и те же изменения в каждом файле.

```
*DO,I,1,5        ! для I = 1 до 5:
LSREAD,I         ! считывание файла нагружения I
OUTPR,ALL,NONE   ! внесение изменений по выходным данным
ERESX,NO
LSWRITE,I        ! перезапись файла нагружения I
*ENDDO
```

Вы можете добавить ваши собственные циклы управления при использовании команд ***IF**, ***EXIT** или ***CYCLE**.

Создавая циклы, помните следующие рекомендации.

- Не выполняйте выход из цикла к метке :Label на командах ***IF** или ***GO**.
- Избегайте использовать :Label для выполнения перехода к другой строке в пределах цикла. Используйте вместо этого конструкцию if-then-else-endif.
- Выходные данные от команд в пределах цикла автоматически подавляются после первого обхода. Используйте команды **/GOPR** или **/GO** в пределах цикла, если Вы хотите видеть выходные данные для всех обходов цикла.
- Будьте внимательны, если Вы используете команду **/CLEAR** в пределах цикла. Команда **/CLEAR** не очищает стек цикла, но она очищает все параметры, включая параметр цикла заявленный в команде ***DO**. Вы можете избежать проблемы наличия неопределенного значения параметра обхода цикла, используя команду **PARSAV** перед командой **/CLEAR**, и затем, после команды **/CLEAR** команду **PARRES**.

4.4.6. Неявные циклы Do Loops

Вы можете использовать принятое соглашение для цикла (двоеточие). Использование этого соглашения обычно ускоряет процесс обработки данных, потому что цикл делается в памяти. Правильный синтаксис:

```
(x:y:z)
```

z по умолчанию равен 1, если не определено другое значение. Например:

```
n,(1:6),(2:12:2)
```

выполнит те же самые шаги как и приведенные ниже:

```
n,1,2
n,2,4
n,3,6
.
.
.
n,6,12
```

Когда используете неявные циклы знайте, что управляет выполнением самое короткое выражение. Например,

```
n,(1:7),(2:12:2)
```

вел бы себя тождественно примеру показанному выше.

Дополнительные числовые поля, которые не имеют двоеточия (:), будут считаться как постоянное значение.

Кроме того, нецелые числа будут функционировать как обычно. Однако, если нецелые числа будут применены к команде, которая требует целых чисел, то тогда нецелое число будет округлено в соответствии с правилами математики.

Это соглашение по циклам может использоваться только для полей, требующих числового ввода. Текстовое поле ввода обработает (x:y:z) буквально.

4.4.7. Дополнительный цикл: Do-While

Вы можете выполнять функции цикла, которые повторяются неопределенное количество раз, пока внешний параметр не изменяется. Команда ***DOWHILE** имеет следующий синтаксис:

```
*DOWHILE,Parm
```

Повторение цикла происходит пока параметр Parm имеет значение «истина». Если `Parm` становится ложным (меньше чем или равный 0.0), цикл заканчивается. Команды ***CYCLE** и ***EXIT** могут использоваться в пределах цикла ***DOWHILE**.

4.5. Краткий справочник функций управления

Следующая ниже таблица описывает команды APDL, которые выполняют функции управления в пределах макроса.

Большинство важной информации об этих командах приведено здесь, но Вы можете посмотреть на полные описания команд в справочной системе ANSYS.

Команды APDL	Выполняемое действие	Примечания
***DO**	Определяет начало цикла. Команды после команды ***DO** выполняются (до команды ***ENDDO**) заданное количество раз.	• Вы можете также управлять циклом через команду ***IF**. • ANSYS допускает до 20 уровней вложенности циклов, хотя циклы, которые включают **/INPUT**, ***USE**, или "неизвестную" команду макроса, имеют меньше уровней вложения, потому что они делают внутреннюю смену файла. • Команды ***DO**, ***ENDDO**, ***CYCLE**, и ***EXIT** должны все считываться из файла или вводиться с клавиатуры. • *Не включайте операции интерактивного выбора в циклы.* • Будьте внимательны, если Вы используете команду **/CLEAR** в пределах цикла. Команда **/CLEAR** не очищает стек цикла, но она очищает все параметры, включая параметр цикла заявленный в команде ***DO**. Вы можете избежать проблемы наличия неопределенного значения параметра обхода цикла, используя команду **PARSAV** перед командой **/CLEAR**, и затем, после команды **/CLEAR** команду **PARRES**.
***ENDDO**	Обозначает окончание цикла и запускает его на выполнение.	Вы должны использовать одну команду ***ENDDO** для каждого вложенного цикла. Команды ***ENDDO** и ***DO** для цикла должны быть в одном и том же файле.
***CYCLE**	Выполняя цикл, ANSYS обходит все команды между командами ***CYCLE** и ***ENDDO**, а затем (если применимо) начинает следующий обход.	Вы можете использовать опцию цикла по условию (через команду ***IF**). Команда ***CYCLE** должна быть в том же самом файле где и команда ***DO** и должна находиться перед командой ***ENDDO**.
***EXIT**	Выходы цикла.	Команда выполняется после команды ***ENDDO**. Команды ***EXIT** и ***DO** для цикла должны быть в том же самом файле. Вы можете использовать опцию выхода по условию (через команду ***IF**).

Команды APDL	Выполняемое действие	Примечания
***IF**	Команда для считывания условия.	• Вы можете иметь до 10 вложенных уровней ***IF** блока. • Вы не можете перескочить из или в пределах цикла или конструкции "if-then-else" к строке с меткой *:label*, и переход к строке с меткой *:label* не допускается вводом с клавиатуры. • Вы можете использовать команду **/CLEAR** в пределах конструкции "if-then-else". Команда **/CLEAR** не очищает ***IF** стек и число ***IF** уровней сохраняется. Команда ***ENDIF** необходима для закрытия любой выполняющий переход конструкции. • Команда **/CLEAR** удаляет все параметры, включая любые параметры, которые используются в ваших командах, выполняющих переход. Вы можете избежать любых проблем, которые могли бы явиться результатом удаления параметров, использовав команду **PARSAV** перед **/CLEAR**, и затем после **/CLEAR** выполнить команду **PARRES**.
***ENDIF**	Заканчивает конструкцию "if-then-else".	Команды ***IF** и ***ENDIF** должны находиться в одном и том же файле.
***ELSE**	Создает заключительный дополнительный раздел блока в пределах конструкции "if-then-else".	Команды ***ELSE** и ***IF** должны находиться в одном и том же файле.
***ELSEIF**	Создает дополнительный, промежуточный раздел блока в пределах конструкции "if-then-else".	Если `Oper` = EQ или NE, *VAL1* и *VAL2* могут быть строками символов (заключенные в одиночные кавычки) или параметрами. Команды ***IF** и ***ELSEIF** должны находиться в одном и том же файле.

4.6. Использование параметров _STATUS и _RETURN в макросах

Программа ANSYS генерирует два параметра, _STATUS и _RETURN, которые Вы можете также использовать в вашем макросе. Например, Вы могли бы использовать значение _STATUS или _RETURN в конструкциях «if-then-else», чтобы получить макрос, который предпринимает некоторое действие, основанное на результате выполнения команды или функции ANSYS.

Функции твердотельного моделирования создают параметр _RETURN, который содержит результат выполнения функции. Следующая таблица определяет значения _RETURN для различных функций твердотельного моделирования:

Таблица 4.1. Значения параметра _RETURN

Команда	Функция	Значение параметра _RETURN
Точки		
K	Определяет точку	Номер точки
KL	Точка на линии	Номер точки

Команда	Функция	Значение параметра _RETURN
KNODE	Точка на узле	Номер точки
KBETW	Точка между двумя точками	Номер точки
KCENTER	Точка в центре	Номер точки
Линии		
BSPLIN	Создает сплайн	Номер линии
CIRCLE	Создает дугу окружности	Номер первой линии
L	Линия между двумя точками	Номер линии
L2ANG	Линия под углом к двум линиям	Номер линии
LANG	Касательная к двум линиям	Номер линии
LARC	Определяет дугу окружности	Номер линии
LAREA	Линия между двумя точками	Номер линии
LCOMB	Объединяет две линии в одну	Номер линии
LDIV	Делит линии на две или более линии	Номер первой точки
LDRAG	Линия вытянутая по точкам	Номер первой линии
LFILLT	Линия скругления между двумя линиями	Номер линии скругления
LROTAT	Дуга относительно точки поворота	Номер первой линии
LSTR	Прямая линия	Номер линии
LTAN	Линия из конца и по касательной	Номер линии
SPLINE	Сегмент сплайна	Номер первой линии
Поверхности		
A	Поверхность по точкам	Номер поверхности
ACCAT	Объединяет две и более поверхности	Номер поверхности
ADRAG	Протаскивание линии вдоль пути	Номер первой поверхности
AFILLT	Скругление по пересечению двух поверхностей	Номер поверхности скругления
AL	Поверхность ограниченная линиями	Номер поверхности
ALPFILL	Все циклы	Номер поверхности
AOFFST	Поверхность смещенная относительно данной поверхности	Номер поверхности
AROTAT	Поворот линий вокруг оси	Номер первой поверхности
ASKIN	Натягивает поверхность по направляющим линиям	Номер первой поверхности
ASUB	Поверхность используя форму существующей поверхности	Номер поверхности
Объемы		
V	Объем по точкам	Номер объема
VA	Объем	Номер объема
VDRAG	Протаскивание поверхности вдоль траектории для создания объема	Номер первого объема
VEXT	Объем выдавливанием поверхности	Номер первого объема
VOFFST	Объем смещением относительно данной поверхности	Номер объема
VROTAT	Объем поворотом поверхности	Номер первого объема

Выполнение команды ANSYS в макросе или в другом месте, создает параметр _STATUS. Этот параметр отражает состояние выполнения этой команды:

- 0 нет ошибки

- 1 для примечания
- 2 для предупреждения
- 3 для ошибки

4.7. Использование макросов с отдельными компонентами и блоками

Чтобы сделать большие модели легко управляемыми, Вы можете разделить модель на отдельные компоненты, основанные на различных типах объектов: узлах, элементах, точках, линиях, поверхностях или объемах. Каждый компонент может содержать только один тип объекта. Это даст Вам возможность легко выполнить задачи типа приложения нагрузок или создания графических видов по отдельности на различных частях модели.

Вы можете также создать блоки, которые являются группами скомбинированными из двух или более компонентов или даже мультиблоки. Вы можете вкладывать блоки до пятого уровня вложенности. Например, Вы можете построить блок, названный motor из компонентов по имени STATOR, PERMMAG, ROTOR, и WINDINGS.

Следующая ниже таблица описывает некоторые из команд, которые Вы можете использовать, чтобы строить компоненты и блоки. Для более детального обсуждения этих команд см. справочную систему ANSYS. Для получения дополнительной информации относительно компонентов и сборок см. Selecting and Components в ANSYS Basic Analysis Guide.

CM	Группы элементы геометрии в компонент.
CMDELE	Удаляет компонент или блок.
CMEDIT	Редактирует существующий компонент или блок. ANSYS обновляет блоки автоматически, чтобы отразить удаления низшего уровня или блоков.
CMGRP	Компоненты групп и блоки в один блок. Однажды определенный блок может быть выведен в листинг, удален, селектирован или анселектирован с использованием тех же самых команд что и для компонентов.
CMLIST	Листинг объектов, содержащихся в компоненте или блоке.
CMSEL	Выбирает подмножество компонентов и блоков.

4.8. Примеры макросов

Ниже следуют два примера макросов. Первый пример макроса, названный `offset.mac`, смещает выбранные узлы в препроцессоре PREP7. Этот макрос выполнен только в целях демонстрации, потому что для этого более удобно применить команду **NGEN**.

```
!   Макрос смещает выбранные узлы в PREP7
!   Следующий ниже файл сохранен как:  offset.mac (нижний регистр)
!   Использование:  offset,dx,dy,dz

/nop            ! подавление вывода команд для этого макроса

*get,nnode,node,,num,max   ! получить количество узлов

*dim,x,,nnode       ! определение массивов для координат узлов
*dim,y,,nnode
*dim,z,,nnode

*dim,sel,,nnode     ! определение массива для выбранного вектора

*vget,x(1),node,1,loc,x   ! получение координат
```

```
*vget,y(1),node,1,loc,y
*vget,z(1),node,1,loc,z

*vget,sel(1),node,1,nsel  ! получение выбранного набора данных

*voper,x(1),x(1),add,arg1 ! местоположение смещения
*voper,y(1),y(1),add,arg2
*voper,z(1),z(1),add,arg3

! *do,i,1,nnode           ! сохранение нового положения
!  *if,sel(i),gt,0,then   ! такая форма требует 98 сек. для 100000 узлов
!    n,i,x(i),y(i),z(i)
!  *endif
! *enddo

*vmask,sel(1)          ! требуется 3 сек. для 100000 узлов
n,(1:NNODE),x(1:NNODE),y(1:NNODE),z(1:NNODE)

x(1) =       ! удаление параметров
y(1) =
z(1) =
sel(1) =
i=
nnode=

/go          ! отмена подавления вывода команд
```

Следующий пример макроса, названный `bilinear.mac`, создает два билинейных материала. Это полезный макрос, который может быть выполнен после выполнения статического анализа. Материал 1 - свойства растяжения и Материал 2 - свойства сжатия. ARG1 - число итераций (значение по умолчанию - 2).

```
/nop
_niter = arg1                       ! ввод количества итераций
*if,_niter,lt,2,then
   _Niter = 2
*endif
*do,iter,1,_niter                   ! цикл с заданным числом итераций
/post1
set,1,1
*get,ar11,elem,,num,maxd            ! получение количества элементов
*dim,_s1,,ar11                      ! массив для элемента s1
*dim,_s3,,ar11                      ! массив для элемента s3
etable,sigmax,s,1                   ! s1 is in element table sigmax
etable,sigmin,s,3                   ! s3 is in element table sigmin
*vget,_s1(1),elem,1,etab,sigmax     ! get element maximum stress in s1
*vget,_s3(1),elem,1,etab,sigmin     ! get element minimum stress in s3
*dim,_mask,,ar11                    ! array for mask vector
*voper,_mask(1),_s1(1),lt,0         ! true if max. stress &lt; 0
*vcum,1                             ! accumulate compression elements
*vabs,0,1                           ! absolute value of s3
*voper,_mask(1),_s3(1),gt,_s1(1)    ! true if abs(minstr) > maxstr
finish

/prep7                            ! go to prep7 for element material mods
mat,1                           ! set all materials to tension properties
emod,all

*vput,_mask(1),elem,1,esel      ! select compression elements
mat,2                           ! change selected elements to compression
emod,all
```

```
call                                    ! select all elements
finish

_s1(1)=                                 ! clean up all vectors (set to zero)
_s3(1)=
_mask(1)=

/solve                                  ! rerun the analysis
solve
finish

*enddo                                  ! end of iterations

_niter=                                 ! clean up iteration counters
_iter=
/gop
```

5. Интерфейс с GUI

В пределах макроса ANSYS, Вы имеете несколько способов обратиться к компонентам графического пользовательского интерфейса ANSYS (GUI):

- Вы можете изменить и обновить панель инструментов ANSYS (это обсуждалось выше в разделе 2.1.).
- Вы можете использовать команду ***ASK**, чтобы запросить пользователя ввести отдельное значение параметра.
- Вы можете создать диалоговое окно, чтобы запросить пользователя ввести множественные значения параметра.
- Вы можете использовать команду ***MSG**, чтобы создать макрос выводящий текстовое сообщение.
- Вы можете создать макрос строки состояния обновления или удаления.
- Вы можете позволить пользователю выбирать объекты через графическое меню выбора внутри макроса.
- Вы можете вызвать любое диалоговое окно.

Далее рассматриваются следующие разделы:

- Запрос пользователя на ввод значения одного параметра
- Запрос пользователя с диалоговым окном
- Использование макросов для отображения ваших собственных сообщений
- Создание и поддержка строки состояния из макроса
- Интерактивный выбор в пределах макроса
- Вызов диалоговых окон из макроса

5.1. Запрос пользователя на ввод значения одного параметра

Используя команду ***ASK** в пределах макроса, Вы можете получить запрос Пользователя на ввод значения параметра.

Формат для команды ***ASK**

***ASK**, *Par*,*Query*,*DVAL*

Где

- *Par* - алфавитно-цифровое имя, которое идентифицируется как скалярный параметр и сохранит значение введенное пользователем.

- *Query* - текстовая строка, которую ANSYS отображает, чтобы запросить пользователя. Эта строка может содержать до 54 символов. *Не используйте символы, которые имеют специальные значения, типа "$" или "!".*
- *DVAL* - значение по умолчанию, которое присваивается параметру, если пользователь не вводит значении (оставляет поле пустым). Это значение может содержать строку от одного до восьми символов (заключенных в единичные кавычки) или число. Если Вы не назначите никакого значения по умолчанию, пустой пользовательский ответ удалит параметр.

Команда ***ASK** печатает текст *Query* на экране и ждет ответа. Она считает ответ введенный с клавиатуры кроме тех случаев, когда ANSYS выполняется в пакетном режиме. (В этом случае ответ или ответы должны содержаться в следующей считываемой строке или строках.) ответ может быть числом, строкой от одного до восьми символов заключенной в одиночные кавычки, числовым или символьным параметром, или выражением, которое определяет численное значение. Следующий пример отображает диалоговое окно, которое показано ниже, затем определяет параметр PARM1 значением, которое вводит пользователь.

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

Рис. 5.1. Пример диалогового окна ***ASC**.

Когда Вы используете ***ASK** в пределах макроса, ANSYS записывает ответ пользователя в файл File.LOG на следующей строке после имени макроса.

5.2. Запрос пользователя с диалоговым окном

Команда **MULTIPRO** создает простое диалоговое окно многократного запроса, которое может содержать до 10 запросов. Команда позволяет Вам использовать ряд UIDL команды ***CSET**, чтобы создать запросы, а так же определить значение по умолчанию для каждого запроса. Обратите внимание, что макрос, использующий **MULTIPRO** нельзя вызвать из UIDL. Вы не можете использовать **MULTIPRO** в пределах цикла типа DO Loop.

Команда **MULTIPRO** должна содержать:

- От одного до десяти приглашений команды ***CSET**
- До двух специальных команд ***CSET**, которые обеспечивают две строковые области для пользовательских инструкций.

Команда имеет следующий синтаксис:

MULTIPRO,'start',Prompt_Num

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

Где

'start'

Буквенная строка, которая ставиться как первый аргумент, отмечает начало конструкции **MULTIPRO**. Аргумент должен быть заключен в одиночные кавычки.

Prompt_Num

Требуется только если аргумент Def_Value опущен по крайней мере в одной команде ***CSET,** или если Def_Value присвоено значение 0. Значение Prompt_Num это целое число равное числу последующих запросов ***CSET**.

Strt_Loc,End_Loc

Начальное значение для аргумента Strt_Loc первой команды ***CSET** = 1, и значение для End_Loc = Strt_Loc+2 (= 3 для первой команды ***CSET**). Значение каждого последующего аргумента Strt_Loc = предыдущий End_Loc+1.

Param_Name

Имя параметра, в который будет считано значение определенное пользователем или, если пользователь не будет вводить никакого значения, значение Def_Value.

'Prompt_String'

Строка, которая может содержать до 32 символов, которые могут использоваться для описания параметра. Эта строка должна быть заключена в одиночные кавычки.

Def_Value

Значение параметра используемое по умолчанию, если пользователем не определено другое значение. Значение по умолчанию может быть числовым выражением, включая числовые параметры APDL. Символьные выражения не допускаются.

'end'

Буквенная строка, которая используется как первый аргумент для заключительной команды **MULTIPRO**.

Ниже следует типичный пример использования команды **MULTIPRO**.

```
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'
```

До двух дополнительных команд ***CSET** могут быть добавлены к конструкции, которые могут обеспечить два строки по 64 символа. Вы можете использовать их, чтобы вывести инструкции пользователю. Для них используется специальный синтаксис команды ***CSET**.

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

Где

'Help_String'

Строка, которая может содержать до 32 символов. Если Вы нуждаетесь в большем количестве символов чем 32, Вы можете использовать второй параметр Help_String.

Ниже следует пример конструкции **MULTIPRO** с использованием дополнительных справочных строк. Обратите внимание, что используется два параметра Help_String, чтобы преодолеть предел на 32 символа.

```
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'
```

Вышеупомянутая конструкция создает следующее диалоговое окно многократного запроса.

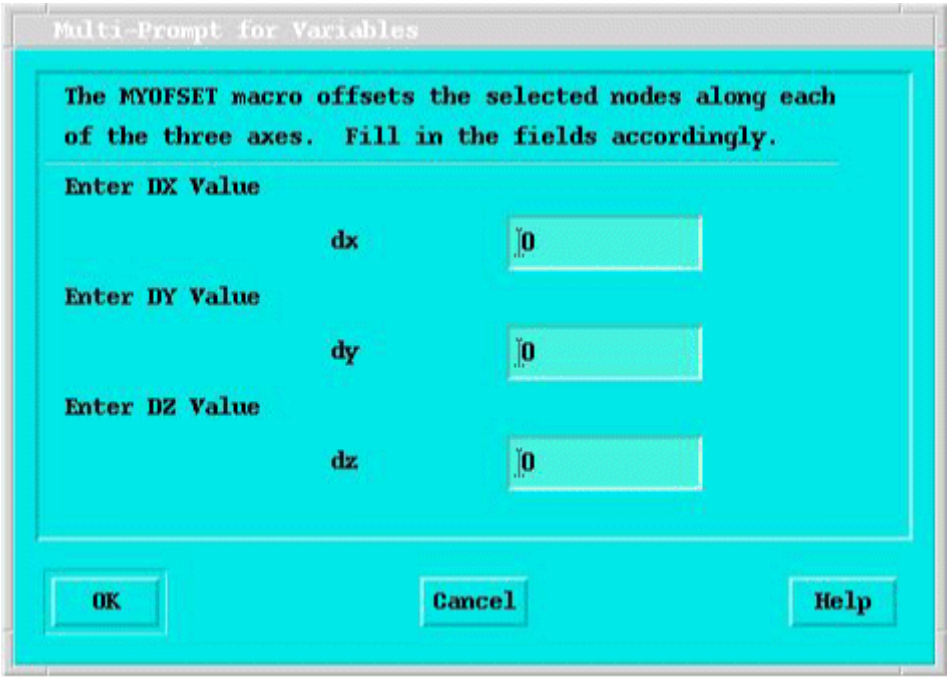


Рис. 5.2. Типичное диалоговое окно для нескольких запросов.

Вы можете проверить состояние кнопок, проверяя значение параметра _BUTTON. Следующие списки значения состояния кнопки:

- _BUTTON = 0 указывает, что была нажата кнопка OK.
- _BUTTON = 1 указывает, что была нажата кнопка Cancel.

В настоящее время, кнопка Помощи не функциональна.

5.3. Использование макросов для отображения ваших собственных сообщений

Используя команду ***MSG** в пределах макроса, Вы можете вывести произвольные сообщения через подпрограмму сообщений ANSYS. Команда имеет следующий формат:

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

Где Lab - одна из следующих меток для управления завершением и выводом:

INFO	Пишет сообщение без заголовка (по умолчанию).
NOTE	Пишет сообщение с заголовком "NOTE".
WARN	Пишет сообщение с заголовком "WARNING", а также записывает его в файл ошибок Jobname.ERR.
ERROR	Пишет сообщение с заголовком "ERROR", а также записывает его в файл ошибок Jobname.ERR. Если оно выполняется в пакетном режиме ANSYS, то эта метка заканчивает выполнение команд на предыдущей правильной выполненной команде.
FATAL	Пишет сообщение с заголовком "FATAL ERROR", а также записывает его в файл ошибок Jobname.ERR. Эта метка также немедленно заканчивает сеанс ANSYS.
UI	Пишет сообщение с возглавляющим "ПРИМЕЧАНИЕМ" и отображает это в диалоговом окне сообщения.

Аргументы с VAL1 по VAL8 это числовые или алфавитно-цифровые символьные значения, которые будут включены в сообщение. Значения могут быть результатами вычислений параметров. Все числовые значения допускают двойную точность (64-битовое представление действительного числа).

Вы должны определить формат сообщения немедленно после команды ***MSG**. Формат сообщения может содержать до 80 символов, состоя из текстовых строк и предопределенных "дескрипторов данных" между строками, где должны быть вставлены числовые или алфавитно-цифровые символьные данные.

Дескрипторы (описатели) данных:

- %i, для целочисленных данных.
- %g, для данных двойной точности
- %c, для алфавитно-цифровых символьных данных
- %/, для прерывания строки

Соответствующие дескрипторы данных ФОРТРАНА для первых трех дескрипторов - I9, 1PG16.9, и A8 соответственно. *Пробел должен предшествовать каждому дескриптору.* Вы также должны проставлять один дескриптор для каждого указанного значения (максимум восемь) в порядке указанных значений.

Не начинайте строку формата после команды ***MSG** с ***IF**, ***ENDIF**, ***ELSE** или ***ELSEIF**. Если последний символ не пустой и заканчивается амперсандом (&), то программа ANSYS считывает вторую строку как продолжение строки формата. Вы можете использовать до 10 строк (включая первую), чтобы определить информацию о формате данных.

Последовательные пробелы сжимаются в один пробел после вывода. Произведенный вывод может быть до 10 строк по 72 символов каждый (использование дескриптора %/).

Пример ниже показывает Вам как использовать ***MSG**, который печатает сообщение с двумя целочисленными значениями и одним действительным значением:

```
*MSG, INFO, 4Inner4 ,25,1.2,148
Radius ( %C) = %I, Thick = %G, Length = %I
```

В результате получается следующее сообщение:

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

Вот пример, иллюстрирующий многострочные тексты с данными в окнах сообщения GUI:

```
*MSG,UI,Vcoilrms,THTAv,Icoilrms,THTAi,Papprnt,Pelec,PF,indctnc
Coil RMS voltage, RMS current, apparent pwr, actual pwr, pwr factor: %/&
Vcoil = %G V (electrical angle = %G DEG) %/&
Icoil = %G A (electrical angle = %G DEG) %/&
APPARENT POWER = %G W %/&
ACTUAL POWER = %G W %/&
Power factor: %G %/&
Inductance = %G %/&
VALUES ARE FOR ENTIRE COIL (NOT JUST THE MODELED SECTOR)
```

Примечание

Команда **/UIS,**MSGPOP контролирует какие сообщения отображаются в диалоговом окне сообщения, когда GUI активен.

5.4. Создание и поддержка строки состояния из макроса

В пределах макроса Вы можете вставить команды, чтобы определить диалоговое окно ANSYS, содержащее строку состояния, отображающую прогресс операции, кнопку STOP, на которую Вы можете нажать, чтобы остановить операцию, или оба этих элемента.

Чтобы определить диалоговое окно состояния используйте следующую команду:

***ABSET**,Title40,Item

- Title40 – текстовая строка, которая появляется в диалоговом окне со строкой состояния. Строка может содержать максимум 40 символов.
- Item – принимает одно из следующих значений:

BAR	Отображает строку состояния без кнопки STOP
KILL	Отображает кнопку STOP без строки состояния
BOTH	Отображает строку состояния и кнопка STOP

Чтобы обновлять строку состояния, используйте команду ***ABCHECK**,Percent, NewTitle.

- Percent - целое число между 0 и 100. Это определяет позицию строки состояния.
- NewTitle – строка в 40 смволов, которая содержит информацию о прогрессе. Если Вы определяете строку для NewTitle, то оно заменяет строку определенную в аргументе Title40 команды ***ABSET**.

Если Вы устанавливаете значение KILL или BOTH, то ваш макрос должен проверять параметр _RETURN после каждого выполнения ***ABCHECK**, чтобы видеть нажал ли пользователь кнопку STOP и предпринять соответствующее действие.

Чтобы удалить строку состояния из ANSYS GUI, используйте команду ***ABFINI**.

Следующий пример макроса иллюстрирует использование строки состояния (состоящей из полосы прогресса и кнопки STOP). Диалоговое окно состояния, которое получается, показано в следующем рисунке. Обратите внимание, что макрос проверяет состояние параметра _RETURN и, если кнопка STOP нажата, отправляет сообщение "We are stopped..... ".

```
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
```

Примечание

Не вызывайте ***ABCHECK** в цикле более чем ≈20 раз.

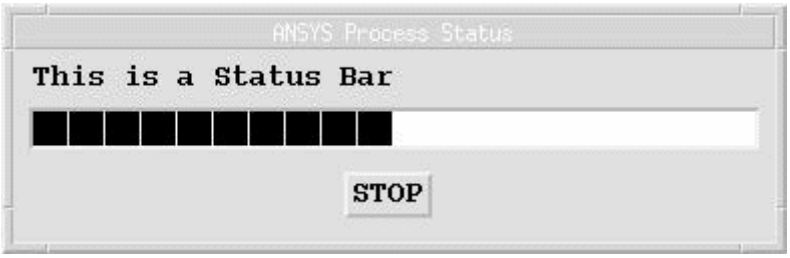


Рис. 5.3. Типичное окно со строкой состояния.

5.5. Интерактивный выбор в пределах макроса

Если Вы используете программу ANSYS в интерактивном режиме, то Вы можете вызвать меню выбора GUI внутри макроса. Чтобы сделать это просто включите команду выбора в макрос. Многие команды ANSYS (типа **K,,**P) принимают ввод значения "P" для того, чтобы использовать интерактивный выбор. Когда ANSYS сталкивается с такой командой он отображает соответствующий диалог выбора и затем, когда пользователь нажимает OK или Cancel, продолжает выполнение макроса.

Имейте в виду, что команды выбора не доступны во всех процессорах ANSYS, и Вы должны сначала переключиться на соответствующий процессор перед запросом команды.

Примечание

Если макрос включает функции GUI, то команда **/PMACRO** должна быть первой командой в этом макросе. Эта команда заставляет команды макроса записываться в журнал сеанса (log-файл). Это важно, потому что, если Вы опустите команду **/PMACRO**, ANSYS не сможет читать журнал сеанса, чтобы воспроизвести сеанс ANSYS.

5.6. Вызов диалоговых окон из макроса

Когда программа ANSYS сталкивается с именем функции UIDL диалогового окна (типа Fnc_UIMP_Iso), то она отображает соответствующее диалоговое окно. Таким образом Вы можете запустить любое диалоговое окно ANSYS, создавая функцию его имени в отдельную строку макроса. Когда Вы закрываете это диалоговое окно, программа продолжает обрабатывать макрос начиная со следующей строки после функционального запроса.

Имейте в виду, что множество диалоговых окон имеют множество зависимостей, включая то, что активен ли соответствующий процессор ANSYS активен и выполнены ли ранее определенные необходимые условия. Например, запуск диалогового окна для выбора узлов сначала предполагает, что узлы существуют, если узлы не будут существовать, то макрос будет потерпит неудачу, когда пользователь нажимает OK или Apply.

Примечание

Если макрос включает функции GUI, то команда **/PMACRO** должна быть первой командой в этом макросе. Эта команда заставляет команды макроса записываться в журнал сеанса (log-файл). Это важно, потому что, если Вы опустите команду **/PMACRO**, ANSYS не сможет читать журнал сеанса, чтобы воспроизвести сеанс ANSYS.

6. Шифрование макросов

ANSYS предоставляет возможность шифровки макрофайлов так, чтобы источник не был "удобочитаемым". Зашифрованный макрос требует, чтобы выполнился ключ кодирования. Вы можете или поместить ключ шифрования явно (в читаемом ASCII) в макросе, или Вы можете установить это в ANSYS как глобальный ключ шифрования.

Далее рассматриваются следующие разделы:

- Подготовка макроса к шифрованию
- Создание зашифрованного макроса
- Выполнение зашифрованного макроса

6.1. Подготовка макроса к шифрованию

Перед шифровкой макроса, Вы сначала создаете и отлаживаете макрос как обычно. *Когда Вы создаете зашифрованный макрос, Вы отвечаете за сохранность оригинального исходного файла. Вы не можете обновить исходный файл из зашифрованного макроса.*

Затем Вы добавляете команду **/ENCRYPT** в первую и последнюю строку макроса. Команда **/ENCRYPT** для первой строки макроса имеет следующий синтаксис:

```
/ENCRYPT,Encryption_key,File_name,File_ext,Directory_Path/
```

Где

- `Encryption_key` - восьмисимвольный пароль.
- `File_name` - имя зашифрованного макрофайла.
- `File_ext` - дополнительное расширение файла для зашифрованного макрофайла. Если Вы хотите, чтобы пользователи выполняли макрос как "неизвестную" команду, Вы должны использовать расширение `.mac`.
- `Directory_Path/`- дополнительный путь к каталогу, который может содержать до 60 символов; Вам понадобится этот аргумент только в том случае, если Вы не хотите записать зашифрованный макрофайл в ваш "домашний" каталог.

В следующем примере обратите внимание на размещение команд **/ENCRYPT** в начале и в конце листинга:

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

Команда **/ENCRYPT** в начале макроса инструктирует ANSYS зашифровать файл и использовать строку "mypasswd" как пароль для расшифровки. В результате ANSYS создаст зашифрованный макрофайл с именем `myenfile.mac` и поместит его в подкаталог `/macros` домашнего каталога. Команда **/ENCRYPT** в конце макроса инструктирует ANSYS остановить шифрование и записать зашифрованный макрос в указанный файл.

Примечание

Зашифрованный макрос использует команду **/NOPR** во второй строке, чтобы выключить повторение команд ANSYS в журнале сеанса. Это важно, если Вы не желаете, чтобы пользователям читали содержание макроса из файла сеанса (log-файла). Чтобы восстановить запись команд в файл сеанса, используйте команду **/GOPR** как последнюю команду в макросе перед замыкающей командой **/ENCRYPT**.

6.2. Создание зашифрованного макроса

После помещения команд **/ENCRYPT** в начале и в конце макроса, Вы можете продолжить создавать зашифрованную версию макроса. Чтобы сделать это, просто выполните макрос через ANSYS. ANSYS создаст зашифрованную версию с именем и местоположением, которое Вы определили через команду **/ENCRYPT** в начале макроса. Результат будет выглядеть примерно как это:

```
/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
```

Обратите внимание, что индивидуальные команды в пределах макроса теперь зашифрованы, и что зашифрованный материал заключен между командами **/DECRYPT**. Ключ расшифровки это аргумент к первой команде **/DECRYPT**.

6.3. Выполнение зашифрованного макроса

Вы можете выполнить зашифрованный макрос так же, как и любой другой макрос; поместите зашифрованный макрос в пределах пути поиска файлов. Если Вы предпочитаете выполнить зашифрованный макрос не имея ключа шифрования в макрофайле, то Вы можете определить ключ как "глобальный ключ шифрования" в пределах ANSYS. Чтобы сделать это, Вы должны сначала заменить аргумент ключа шифрования в команде **/DECRYPT** на параметр PASSWORD. Таким образом, первая строка зашифрованного макроса становится:

```
/DECRYPT,PASSWORD
```

Перед выполнением макроса в пределах ANSYS, используйте следующую команду через командную строку ANSYS:

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

Где Encryption_Key - ключ шифрования, используемый для шифровки файла. Теперь Вы можете выполнять зашифрованный пароль. Чтобы удалить текущий глобальный ключ шифрования, используйте следующую команду ANSYS:

```
/DECRYPT,PASSWORD,OFF
```