

**ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ АВТОМАТИЗАЦИИ  
МОДЕЛИРОВАНИЯ НЕСТАЦИОНАРНЫХ ПРОЦЕССОВ  
В МЕХАНИЧЕСКИХ СИСТЕМАХ И СИСТЕМАХ ИНОЙ  
ФИЗИЧЕСКОЙ ПРИРОДЫ**

**PRADIS**

**версия 2.2**

**Включение программ пользователя в  
библиотеки комплекса**

**Руководство пользователя**

## СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ .....	I-1
2. ОБЩИЕ ПОЛОЖЕНИЯ .....	II-1
2.1. Структура библиотечной программы .....	II-1
2.2. Информационная часть библиотечного модуля .....	II-1
2.3. Связь библиотечной программы с вычислительным ядром .....	II-2
2.3.1. Непоименованная общая область .....	II-3
2.3.2. COMMON - область NOTAT .....	II-7
2.4. Процедура включения программы пользователя в библиотеки комплекса .....	II-8
3. ВКЛЮЧЕНИЕ МОДЕЛЕЙ ЭЛЕМЕНТОВ В БИБЛИОТЕКИ КОМПЛЕКСА .....	III-1
3.1. Краткое описание вычислительного алгоритма. Функции модели элемента в составе комплекса.....	III-1
3.2. Структура оператора SUBROUTINE модели элемента .....	III-3
3.3. Паспорт модели элемента .....	III-6
3.4. Справочная информация по модели элемента .....	III-10
3.5. Особенности вычислительного ядра, на которые нужно обратить внимание при разработке модели элемента .....	III-10
3.5.1. Проверка параметров на допустимость, начальные инициализации рабочего вектора и вектора состояния .....	III-10
3.5.2. Особенности моделей элементов, задающих начальные условия .....	III-12
3.5.3. Учет возможности использования различных схем интегрирования и обеспечение повторной входимости в модель элемента .....	III-14
3.5.4. Некоторые другие важные особенности	

моделей элементов .....	III-15
3.6. Пример разработки модели элемента .....	III-15
3.6.1. Постановка задачи .....	III-15
3.6.2. Основные зависимости для модели элемента и ее параметры .....	III-16
3.6.3. Якобиан элемента .....	III-17
C - 1	
3.6.4. Особые ситуации .....	III-18
3.6.5. Выбор имени модели пружины .....	III-19
3.6.6. Паспорт модели элемента .....	III-20
3.6.7. Текст модели элемента .....	III-21
3.6.8. Автономное тестирование модели элемента .....	III-25
3.6.9. Текст модели технической системы на языке PRADIS с использованием элемента LINKD .....	III-29
3.6. Еще один пример модели элемента .....	III-31
4. ВКЛЮЧЕНИЕ ПРОГРАММ РАСЧЕТА ВЫХОДНЫХ ПЕРЕМЕННЫХ В БИБЛИОТЕКИ КОМПЛЕКСА .....	IV-1
4.1. Функции программ расчета выходных переменных и структура оператора SUBROUTINE .....	IV-1
4.2. Паспорт программы расчета выходных переменных .....	IV-5
4.3. Справочная информация по ПРВП .....	IV-7
4.4. Примеры программ расчета выходных переменных комплекса PRADIS .....	IV-7
4.4.1. ПРОГРАММА X. Расчет перемещения, силы и элемента рабочего вектора .....	IV-7
4.4.2. ПРОГРАММА MAXI. Расчет максимального значения из N внутренних переменных .....	IV-8
5. ВКЛЮЧЕНИЕ ПРОГРАММ РЕАЛИЗАЦИИ ГРАФИЧЕСКИХ ОБРАЗОВ В БИБЛИОТЕКИ КОМПЛЕКСА .....	V-1
5.1. Функции программ реализации графических образов и структура оператора SUBROUTINE .....	V-1

- 5.2. Паспорт программы реализации графического образа и правила формирования имен ПГО, связанных с моделями элементов по умолчанию .. V-4
- 5.3. Основные принципы, положенные в основу разработки программ реализации графических образов ..... V-6
- 5.4. Справочная информация по ПГО ..... V-7
- 5.5. Примеры программ реализации графических образов ..... V-7

ПРИЛОЖЕНИЕ 1. Описание нижнего графического уровня комплекса ..... П1-1

ПРИЛОЖЕНИЕ 2. Список зарезервированных имен ..... П2-1

## 1. ВВЕДЕНИЕ

В состав вычислительного ядра программного комплекса PRADIS кроме постоянной части входят несколько библиотек программ, которые обеспечивают возможности комплекса в части решения задач для той или иной предметной области и отображения полученных результатов.

Это - расширяемые библиотеки комплекса. Для каждого типа библиотек комплекса (моделей элементов, программ расчета выходных переменных (ПРВП), программ графических образов (ПГО) и программ отображения) существует набор базовых модулей, достаточно универсальных, чтобы во многих задачах ограничиваться только их использованием. Однако, большое разнообразие технических систем и критериев оценки протекающих в них процессов требуют соответствующей гибкости от программного обеспечения. Такая гибкость в ПК PRADIS поддерживается возможностью включения новых модулей как в базовые, так и в пользовательские библиотеки. Поэтому состав библиотек комплекса динамичен и для текущей версии комплекса зависит от конфигурации и предметной ориентации конкретной поставки.

Включение какого-либо модуля в состав комплекса PRADIS может быть осуществлено квалифицированным пользователем по формальным правилам, описанным в настоящем документе.

Для того, чтобы начинающий разработчик библиотечных модулей комплекса PRADIS представил себе место этих программ в вычислительном алгоритме, воспользуемся рис. 1.1. Исходная система дифференциальных уравнений решается численно. На каждом шаге интегрирования применяются формулы дискретизации (для различных методов интегрирования они РАЗНЫЕ), что позволяет перейти от исходных уравнений к системе нелинейных уравнений. Полученная система уравнений решается итерационно и на каждой итерации производятся обращения к МОДЕЛЯМ элементов. По завершении каждого шага интегрирования для расчета требуемых выходных переменных происходит обращение к необходимым ПРОГРАММАМ РАСЧЕТА ВЫХОДНЫХ ПЕРЕМЕННЫХ. Если пользователь в своем задании предусмотрел отображение объекта по ходу расчета, то здесь же происходит обращение к программам реализации ГРАФИЧЕСКИХ ОБРАЗОВ.

На рис. 1.1. не указана одна важная особенность вычислительного алгоритма PRADIS. Некоторая часть моделей и программ расчета выходных переменных может потребовать предрасчетной инициализации данных. Поэтому существует понятие НУЛЕВОГО ШАГА ИНТЕГРИРОВАНИЯ. На нулевом шаге интегрирования обращения производятся только к моделям элементов и программам расчета выходных переменных. На этом шаге интегрирования номер итерации равен 0, фактическое интегрирование не производится. Таким образом, все инициализации, проверку параметров на допустимость и однократные вычисления

- в МОДЕЛЯХ элементов и ПРОГРАММАХ РАСЧЕТА ВЫХОДНЫХ

ПЕРЕМЕННЫХ следует производить на нулевом шаге интегрирования и при каждом новом вызове программы

I - 1

интегрирования, осуществляемом с использованием заголовка \$ RESTORE;

- в ПРОГРАММАХ РЕАЛИЗАЦИИ ГРАФИЧЕСКИХ ОБРАЗОВ - на первом шаге интегрирования.

Структура библиотечных модулей комплекса во многом похожа. Однако, поскольку модули различного типа выполняют в составе комплекса PRADIS различные функции, естественно, что содержательная часть этих модулей имеет значительные отличия. Информация, общая для библиотечных модулей всех типов, приводится в разделе 2 настоящего документа. Далее приводится более подробная информация, необходимая для включения в состав комплекса PRADIS модели элемента (глава 3), программы расчета выходных переменных (глава 4) и программы реализации графического образа (глава 5). Там же приводятся примеры конкретных библиотечных программ различных типов.

Каждая из включаемых в состав комплекса библиотечных программ должна корректно взаимодействовать с другими компонентами вычислительного ядра комплекса. Поэтому при разработке своих программ пользователь должен давать им уникальные имена. Список зарезервированных имен PRADIS приводится в Приложении 2 к настоящему документу. Кроме того, имя вновь разрабатываемого модуля не должно пересекаться со списком уже имеющихся в составе вашего комплекса моделей элементов, программ расчета выходных переменных и программ реализации графических образов (справка по этим модулям выдается по запросу ARM ?).

Если программа пользователя осуществляет операции ввода-вывода в создаваемые ей временные или постоянные рабочие файлы, открываемые самой программой пользователя, необходимо иметь в виду, что номера файлов с 1 по 20 могут использоваться вычислительным ядром комплекса для своих нужд.

И последнее. Исходный код комплекса PRADIS в основном удовлетворяет стандарту языка FORTRAN-77. Отступления от стандарта :

- явные описатели типов (REAL \* 8, INTEGER \* 4);
- наличие символьных и числовых переменных в одном и том же COMMON-блоке (символьная переменная NAME в непоименованной общей области вычислительного ядра PRADIS).

Поскольку вы пользуетесь этим документом для включения своих программ в состав комплекса PRADIS, используемый вами компилятор с FORTRANa позволяет это сделать (иначе, комплекс PRADIS у вас не работал бы). Если вы хотите, чтобы разрабатываемые вами модули, как и PRADIS, не встречали проблем при переносе с одной вычислительной установки на другую, то иных отступлений от стандарта лучше не допускать.

## 2. ОБЩИЕ ПОЛОЖЕНИЯ

### 2.1. Структура библиотечной программы.

Библиотечная программа является подпрограммой общего вида (SUBROUTINE), написанной на языке FORTRAN. Она состоит из следующих основных частей :

1) Информационная. Находится в начале программы. Состоит из одной или нескольких строк, имеющих в первой позиции символ "C". С точки зрения языка FORTRAN эта часть подпрограммы содержит комментарии. Информационная часть программы заканчивается первой строкой, не имеющей в первой позиции символа "C". При включении модуля в библиотеки комплекса она обрабатывается утилитами PRADIS для получения дополнительной информации о назначении и интерфейсе включаемого модуля.

2) Описание точки входа в программу (оператор SUBROUTINE).

3) Содержательная часть программы.

4) Оператор возврата управления в основную программу.

### 2.2. Информационная часть библиотечного модуля.

И н ф о р м а ц и о н н а я часть программы содержит два важных элемента - паспорт программы и справочную информацию по данному модулю, которая будет доступна пользователю в режиме "ON LINE" (непосредственно при работе с комплексом на экране дисплея или в виде распечатки).

ПАСПОРТ ПРОГРАММЫ необходим для того, чтобы исполняющая система комплекса PRADIS могла правильно построить вызов этого модуля. Фактически, это формальное описание вида оператора SUBROUTINE, который начинает содержательную часть программы. Примеры паспортов :

C MODEL TEST1:EXT=3,PAR=2,WRK=1 - паспорт модели элемента TEST1

C OUTPUT TEST2:SYS=3,OUT=2,PAR=3 - паспорт программы расчета выходных переменных TEST2

C IMAGE TEST3: EXT=2 - паспорт программы  
графического образа TEST3.



Содержание паспорта, состав и назначение его элементов зависят от функционального назначения включаемого модуля. Поэтому более подробная информация о паспортах будет дана ниже, при обсуждении конкретных типов включаемых в библиотеки комплекса модулей.

СПРАВОЧНАЯ ИНФОРМАЦИЯ предназначена для включения во встроенный HELP системы. Утилита включения модуля в библиотеку,

## II - 1

просматривая информационную часть программы, помещает справочную информацию в системный каталог и связывает ее с именем включаемого модуля. С этого момента информация становится доступной по команде ARM (например, ARM ? и ARM ? TEST1). Строки, содержащие справочную информацию, как и все строки информационной части программы, начинаются символом "С" и, кроме того, содержат ключевое слово HELP. Пример справочной информации по модулю TEST :

```
С
С HELP Идеально упругий одномерный элемент TEST.
С HELP НАЗВАНИЕ: Идеально-упругий одномерный безинерционный
С HELP элемент, служащий для описания связи двух
С HELP тел.
С
С HELP ОБЛАСТЬ ПРИМЕНЕНИЯ : Механика.
С
С HELP СТЕПЕНИ СВОБОДЫ:
С HELP 1 - поступательная точки А элемента.
С HELP 2 - поступательная точки В элемента.
С
С HELP ПАРАМЕТРЫ :
С HELP 1 - коэффициент жесткости.
С
```

В приведенном примере в системный каталог системы попадает 10 строк справочной информации. При этом первая строка попадает в сокращенный справочник, выдаваемый по команде ARM ?. Целесообразно первую строку справочной информации ВСЕГДА использовать для краткого описания назначения включаемой в библиотеку программы. Остальные строки справочной информации попадают в расширенный справочник, информация из которого выдается по запросу ARM ? <имя модуля>. Рекомендуется при создании справочной информации по конкретному модулю придерживаться порядка и содержания подразделов, которые приняты для базовых модулей соответствующего типа.

Более подробно о содержании информационной части библиотечного модуля, структуре его паспорта, описании точки входа в модуль и структуре содержательной

части приводятся в соответствующих разделах этого документа при описании модулей различных типов.

### 2.3. Связь библиотечной программы с вычислительным ядром.

Программа, включаемая в библиотеки комплекса PRADIS, осуществляет связь с вычислительным ядром комплекса с помощью механизма формальных/фактических параметров и через общие области данных.

Имеется две общедоступные для включаемых в библиотеки комплекса COMMON - области. Это непоименованная COMMON - область и поименованная COMMON - область NOTAT. Кроме того,

#### II - 2

COMMON - область GRCONF может оказаться полезной для программ реализации графических образов (она описана в Приложении 1).

При разработке библиотечной программы рекомендуется ВСЕГДА включать в текст программы описание непоименованной общей области и COMMON-области NOTAT, используя для обозначения переменных те же имена, что и в этом документе.

#### 2.3.1. Непоименованная общая область.

Имена включенных в непоименованную COMMON - область переменных, их типы и назначение описаны в таблице 1.

Таблица 1. СОДЕРЖАНИЕ НЕПОИМЕНОВАННОЙ ОБЩЕЙ ОБЛАСТИ  
КОМПЛЕКСА PRADIS

Имя переменной	Тип	Примечание
TIME	R*8	Момент времени интегрирования, для которого получено решение. Текущее время определяется как TIME+STEP
STEP		Текущая величина шага интегрирования

STEP01		Значение предыдущего шага интегрирования
STEP02	R*8	Значение предпредыдущего шага интегрирования
SMIN		Минимально возможный шаг интегрирования
DABSI	R*8	Абсолютная составляющая допустимой невязки правой части в процессе решения СнЛУ для данной программы интегрирования
DRLTI	R*8	Относительная составляющая допустимой невязки правой части в процессе решения СнЛУ для данной программы интегрирования
STEPMD	R*8	Максимальное значение следующего шага интегрирования, рекомендуемое моделью элемента
TIMEND		Время окончания интегрирования
NAME	C*8	В эту переменную при выходе из программы пользователя со значением CODE, отличным от нуля, помещается имя этой программы
NSTEP		Номер текущего шага интегрирования

## II - 3

Таблица 1 (продолжение)

Имя переменной	Тип	Примечание
SYSPRN	I*4	Номер открытого файла системной печати (SYSPRINT.TXT)
NITER		Номер текущей итерации
ITR	I*4	Максимально допустимое количество итераций на шаге интегрирования
CODE		Код возврата из модуля
NUMINT	I*2	Порядковый номер работающей программы интегрирования

NUMPRV	I*2	Порядковый номер вызванной модели элемента или программы расчета выходных переменных
CODSTP	I*2	Если этот признак имеет значение, равное 1, то модель элемента может не вычислять якобианы при текущем вызове модели
CODGRF	I*2	Признак инициализации графического режима
NEWINT	I*2	Действителен ТОЛЬКО ДЛЯ МОДЕЛЕЙ ЭЛЕМЕНТОВ. Признак того, что модель элемента вызвана данной программой интегрирования первый раз
MINSTP	I*2	Признак того, что текущий шаг интегрирования минимален

Программа пользователя может изменять следующие элементы COMMON-блока :

STEPMD - максимальный рекомендуемый моделью элемента шаг интегрирования (эта COMMON - переменная используется только моделями элементов)

CODE - код возврата из программы пользователя

NAME - имя программы пользователя, установившей код возврата.

STEPMD используются моделями элементов, "внутри" которых могут произойти те или иные события, достаточно сильно влияющие на анализируемый процесс. Типичные модели такого типа

- различные упоры, зависимости силы от времени или перемещения и т.д. В этом случае, если вычислительное ядро комплекса не имеет информации о том, что должно произойти какое-либо

## II - 4

событие, то может быть выбран слишком большой шаг интегрирования. Это, в свою очередь, может привести к тому, что событие вообще останется "незамеченным" (скажем, шаг интегрирования может превышать продолжительность действия импульса силы). Другая неприятность - большие потери шагов интегрирования из-за несоответствия точностным параметрам или несходимости решения системы нелинейных уравнений (например, внутри шага интегрирования упор был подвергнут

значительной деформации, в то время как на предыдущем шаге интегрирования он не работал).

В этих случаях модель элемента должна "сообщить" вычислительному ядру комплекса о предполагаемом промежутке времени до наступления того или иного события. Если это событие наступает по времени, то промежуток времени до его наступления может быть определен точно (разница между моментом времени наступления события и текущим временем). Если событие наступает не по времени (как в упорах), но модель элемента с той или иной долей уверенности может предсказать время наступления события, лучше это все равно сделать. Например, модели упоров, как правило, предсказывают момент времени, когда упор начинает срабатывать, исходя из текущих перемещения, скорости и ускорения.

Итак, в этих случаях модель элемента может рекомендовать вычислительному ядру сделать шаг такой величины, чтобы попасть точно в начало события. Следующий шаг интегрирования будет выбран, исходя из этой рекомендации и других соображений (например, точность интегрирования), но не будет превышать величины шага, рекомендованного моделью. При установке переменной STEPMD модель элемента должна иметь ввиду, что в анализируемом объекте могут быть другие модели, которые также могут делать свои рекомендации. Чтобы не потерять эту информацию, используется такой прием. Допустим, модель элемента предполагает, что соответствующее событие наступит через промежуток времени PROGTM. Тогда установка переменной STEPMD осуществляется оператором

$$\text{STEPMD} = \text{MIN} (\text{STEPMD}, \text{PROGTM})$$

С помощью переменной CODE библиотечный модуль может передать вычислительному ядру сигнал о возникновении той или иной ситуации. Перечень возможных ситуаций, обрабатываемых вычислительным ядром, и соответствующие им коды приводятся ниже. Если программа пользователя устанавливает переменную CODE, то в переменную NAME она должна переслать свое имя (поскольку в этом случае имя соответствующей модели элемента используется в сообщениях вычислительного ядра комплекса).

Переменная CODE, вообще говоря, контролируется после выполнения любой программы пользователя. Однако для программ расчета выходных переменных и программ - графических образов рекомендуется не злоупотреблять использованием этой переменной (лучше, если в некорректной или не очень корректной ситуации программа расчета выходной переменной или графический образ выберут "разумную линию поведения" и не будут прерывать процесса интегрирования или вмешиваться в ход расчета).

Следует подчеркнуть, что это не требование, накладываемое на программы комплексом, а рекомендация пользователю. Просто, если всякая там ПРВП берет на себя смелость прервать расчет, длящийся иногда часами, по пустяку - это раздражает.

Ниже описаны возможные ситуации, возникающие в ходе расчета внутри программы пользователя и значения переменной CODE, которые требуется установить в этом случае :

- Требуется продолжить ньютоновские итерации на данном шаге интегрирования, даже если сходимость будет достигнута. Эта необходимость может возникнуть только в моделях элементов, поскольку в ПРВП и ПГО обращение происходит только по завершении шага интегрирования. Если значение COMMON-переменной  $CODE < 5$ , модель элемента устанавливает  $CODE = 5$  ( $CODE = \text{MAX}(CODE, 5)$ ); в COMMON переменную NAME пересылается имя модели элемента.

- Требуется уменьшить шаг интегрирования.

Если значение COMMON-переменной  $CODE < 10$ , программа пользователя устанавливает  $CODE = 10$ ; в COMMON переменную NAME пересылается имя программы пользователя. Эта необходимость также может возникнуть только в моделях элементов, поскольку в ПРВП и ПГО обращение происходит только по завершении шага интегрирования.

- Требуется завершить работу текущей программы интегрирования с сохранением данных о состоянии расчета

Если значение COMMON-переменной  $CODE < 50$ , программа пользователя устанавливает  $CODE = 50$ ; в COMMON переменную NAME пересылается имя программы пользователя.

- Возникла аварийная ситуация, делающая корректное продолжение расчетов невозможным.

Если значение COMMON-переменной  $CODE < 75$ , программа пользователя устанавливает  $CODE = 75$ ; в COMMON переменную NAME пересылается имя программы пользователя.

- Модель элемента пытается переустановить значение потенциальной переменной узла, для которого это значение уже установлено. Для ПРВП и ПГО эта операция является запрещенной.

Если значение COMMON-переменной  $CODE < 90$ , модель элемента устанавливает  $CODE = 90$ ; в COMMON переменную NAME пересылается имя этой модели элемента.

- Параметры, переданные в программу пользователя, лежат вне допустимого диапазона значений.

## II - 6

Если значение COMMON-переменной CODE < 100, программа пользователя устанавливает CODE = 100; в COMMON переменную NAME пересылается имя программы пользователя.

Как сказано выше, значения переменной CODE 5 или 10 используются только в моделях элементов. Если эти коды установлены, программа интегрирования продолжает работу. Значение CODE = 90 также устанавливается моделью элемента. Этот код приводит к аварийному (немедленному) завершению расчета без окончания шага интегрирования.

Значения CODE 50, 75, 100 используются любыми программами пользователя. Код 50 приводит к нормальному завершению расчета по окончании текущего шага интегрирования, 75 и 100 - к немедленному прекращению расчета.

Изменение других переменных непоименованной общей области программой пользователя, как правило, приводит к аварийному завершению расчета.

### 2.3.2. COMMON - область NOTAT.

Часто в процессе расчетов в программе требуется использовать машинно-зависимые и общеупотребительные константы. При использовании зависящих от вычислительной техники констант (например, машинный ноль) желательно не привязывать программу пользователя к конкретной технике. Более целесообразно воспользоваться тем фактом, что PRADIS определяет некоторые машинные константы и помещает их в поименованную COMMON-область NOTAT. Многие программы при вычислениях используют число "пи", при этом точность его задания может зависеть от вычислительного алгоритма. В то же время часто бывает желательным согласовать это значение для различных расчетных программ. Поэтому также рекомендуется для числа PI использовать значение, хранящееся в COMMON-блоке NOTAT (см. таблицу 2).

Таблица 2. СОДЕРЖАНИЕ ПОИМЕНОВАННОЙ ОБЩЕЙ ОБЛАСТИ NOTAT

N	Имя переменной	Тип	Примечание
---	----------------	-----	------------

1	RLMAX	R*8	Верхняя граница представимых действительных положительных чисел двойной точности
2	RLMIN	R*8	Нижняя граница представимых действительных положительных чисел двойной точности
3	INTMAX	R*8	Верхняя граница представимых целых положительных чисел, описанных как INTEGER*4

## II - 7

N	Имя переменной	Тип	Примечание	
4	MSHEPS	R*8	Машинное эpsilon для действительных чисел двойной точности (разница между 1 и ближайшим большим действительным числом двойной точности)	
5	PI	8	Число	двойной точности
	REZERV (3)	8	Зарезервировано	

### 2.4. Процедура включения программы пользователя в библиотеки комплекса.

В комплексе PRADIS имеется две объектные библиотеки, содержащие модели элементов, программы расчета выходных переменных и программы графических образов. Это библиотеки BASIS.LIB и CURRENT.LIB.

BASIS.LIB содержит основную часть объектных модулей, которая поставляется с данной конфигурацией комплекса ("базовые библиотеки"). Библиотека CURRENT.LIB служит для включения программ пользователя. Поэтому желательно, чтобы эта библиотека имела небольшой размер, иначе процедура включения объектного модуля в эту библиотеку каждый раз будет занимать значительное время. Если пользователь большую часть времени работы с PRADIS посвящает разработке и



включению своих программ в библиотеки комплекса, то рекомендуется время от времени контролировать размер библиотеки CURRENT.LIB. В случае, если она будет иметь существенный размер, модули, которые не предполагается больше изменять, можно перенести в BASIS.LIB. Перенос объектных модулей из библиотеки в библиотеку осуществляется средствами обычной программы LIB. Пользователь намечает список устоявшихся модулей, исключает их из библиотеки CURRENT.LIB (например, с помощью операции \*-, если речь идет о пакете MS-FORTRAN) и включает их в библиотеку BASIS.LIB.

**ВНИМАНИЕ !** Пользователь должен сам заботиться о том, чтобы библиотека CURRENT.LIB в любой момент времени была доступна для программы LIB, т.е., нельзя допускать ситуации, чтобы она вся была перенесена в BASIS.LIB. Иначе команда на включение в CURRENT.LIB нового объектного модуля, выдаваемая процедурой ARM, может быть некорректной.

Поиск библиотек осуществляется процедурами комплекса по правилам, принятым для используемого пакета FORTRAN.

Включение программы пользователя в библиотеки комплекса осуществляется процедурой ARM. Для получения объектного

## II - 8

модуля, его включения в библиотеку CURRENT.LIB и включения справочной информации в системный каталог комплекса используется команда "ARM +". Например :

> ARM + MODEL

Необходимо обратить внимание, что для того, чтобы эта команда была корректной, в текущем каталоге должен присутствовать файл MODEL.FOR, содержащий программу пользователя, оформленную в соответствии с правилами, приведенными в настоящем документе.

По этой команде процедура ARM инициирует компилятор с FORTRANa. В случае, если синтаксических ошибок не обнаружено, полученный объектный код включается в библиотеку CURRENT.LIB программой-библиотекарем из используемого пакета FORTRAN.

После этого включается в работу утилита PRADIS, анализирующая информационную часть программы и включающая ее в системный каталог. В файл SYSPRINT.TXT попадает листинг информационной части программы с соответствующими сообщениями. Если в системном каталоге информация по данному модулю отсутствовала, то выдается сообщение, что паспорт программы добавлен в системный каталог. В противном случае сообщение указывает, что паспорт программы замещается в системном каталоге.

Библиотечная программа, как правило, содержит справочную информацию, предназначенную для включения в системный каталог. Если включение этой информации проходит успешно, утилита выдает сообщение :

М (I 001) Обработана и внесена в системный каталог справочная информация по включаемому модулю.

В случае отсутствия в обрабатываемом модуле справочной информации по включаемому модулю, в SYSPRINT.TXT попадает сообщение:

М (I 002) Включаемый в системный каталог модуль не содержит справочной информации.

Нормальное завершение утилиты включения модуля в системный каталог сопровождается сообщением об успешном завершении программы.

Если не требуется получать объектный модуль программы и включать его в CURRENT.LIB, а нужно только включить или заменить информационную часть программы в системном каталоге (например, возникла ситуация, когда неверно был задан паспорт программы), используется команда "ARM !":

> ARM ! MODEL

В этом случае процедура не будет вызывать компилятор с языка FORTRAN, а сразу иницирует утилиту включения модуля в системный каталог.

## II - 9

Наконец, если не требуется включать информационную часть программы в системный каталог, а необходимо получить и включить объектный модуль в объектную библиотеку, используется команда "ARM #":

> ARM # MODEL

В этом случае после вызова компилятора и библиотекаря не будет вызвана утилита включения модуля в системный каталог.

Исключение ссылки на библиотечную программу из системного каталога осуществляется командой "ARM -" :

> ARM - MODEL

По этой команде процедура обслуживания системного каталога вызовет утилиту удаления ссылки на соответствующий модуль из системного каталога и программу-библиотекарь для удаления этого объектного модуля из объектной библиотеки.

Каждая из описанных выше команд может быть использована для выполнения одноименной операции одновременно для нескольких модулей, например:

> ARM + MODEL PRVP TEST

После завершения операций включения/исключения модуля рекомендуется проверить правильность сделанной работы, последовательно выполнив команды получения справочной информации. По команде

> ARM \*

выдается отсортированный по алфавиту список всех тем, по которым в системном каталоге имеется справочная информация. Имя включенной библиотечной программы должно присутствовать в этом списке.

По команде

> ARM ?

распечатывается содержимое каталогов расширяемых компонент комплекса. Если все было сделано верно, то ссылка на включенную библиотечную программу должна присутствовать в одном из этих каталогов. Напротив имени этой модели должна распечатываться первая строка справочной информации.

По команде

> ARM ? MODEL

выдается расширенная справочная информация о библиотечной программе (вся справочная информация, кроме первой строки).

## II - 10

Для оперативной справки о возможностях команды ARM (например, пользователь забыл, чему соответствуют ключи !, # и +), можно выдавать команду ARM без параметров. В этом случае на экран дисплея будет выдана краткая справка о возможностях процедуры обслуживания системного каталога.



### 3. ВКЛЮЧЕНИЕ МОДЕЛЕЙ ЭЛЕМЕНТОВ В БИБЛИОТЕКИ КОМПЛЕКСА

#### 3.1. Краткое описание вычислительного алгоритма. Функции модели элемента в составе комплекса.

Здесь приводятся лишь краткие сведения по вычислительному алгоритму комплекса. Для более полного понимания содержания этого документа рекомендуется знакомство с подробным описанием используемых в PRADIS математических методов.

Математическая модель произвольного технического объекта формируется вычислительным ядром комплекса PRADIS с использованием узлового метода (рис. 3.1).

В основе этого алгоритма лежит формирование и решение системы уравнений, отражающих законы сохранения в той или иной предметной области. При описании технического объекта пользователем осуществляется дискретизация пространства (т.е., декомпозиция объекта на элементы и выделение из бесконечного количества степеней свободы реального объекта какого-то конечного их числа, которые будут представлены в модели). Уравнения равновесия формируются для каждой степени свободы модели. Они записываются относительно переменных, которые условно можно назвать потоковыми (сила, момент в механике, расход в гидравлике, ток в электронике). Полная система уравнений равновесия для рассматриваемой модели технической системы называется системой топологических уравнений, поскольку она фактически отражает структуру связей между элементами, входящими в эту модель.

Состояние каждой степени свободы модели в каждый момент времени характеризуется некоторым значением переменной, которую условно можно назвать потенциальной (перемещение, угол поворота, давление, потенциал). Каждый из потоков, входящих в уравнение равновесия, в общем случае зависит от времени и значений потенциалов для нескольких степеней свободы модели. Уравнения, позволяющие связать величину потока для заданного момента времени с соответствующими значениями потенциалов, называются компонентными.

Подстановка компонентных уравнений в систему топологических уравнений приводит в общем случае к системе нелинейных дифференциальных уравнений. Использование формул численного интегрирования позволяет на каждом шаге интегрирования преобразовать дифференциальные уравнения в нелинейные (алгебраические и трансцендентные). Таким образом, для выполнения каждого шага интегрирования вычислительным ядром должна быть решена система нелинейных уравнений. Она решается итерационно методом Ньютона. В алгоритм метода Ньютона входит формирование на каждой итерации системы линейных алгебраических уравнений и ее решение. Матрица коэффициентов этой системы линейных уравнений называется якобианом системы. Это частные производные от соответствующих потоковых переменных по соответствующим потенциальным переменным (т.е.,  $i$ -я строка якобиана соответствует частным производным от потоков, входящих

#### III - 1

в  $i$ -е уравнение, а  $j$ -й столбец - частным производным по  $j$ -й потенциальной переменной). Частные производные вычисляются при значениях потенциальных переменных, соответствующих текущему приближению к решению.

Для заданных значений потенциалов модель элемента может определить текущие значения потоков и величины частных производных от потоков по потенциалам. Суммирование потоков, вычисленных каждой из моделей элементов, приводит в общем случае к некоторой невязке (она возникает из-за того, что по сравнению с предыдущим шагом интегрирования состояние анализируемой системы несколько изменилось, а значения потоковых переменных вычислены для значений потенциальных переменных на предыдущем шаге интегрирования). Смысл решения системы нелинейных уравнений на шаге интегрирования состоит в подборе таких значений потенциальных переменных, которые будут с заданной точностью удовлетворять уравнениям равновесия.

Кроме суммирования потоков на каждой итерации метода Ньютона для получения матрицы коэффициентов системы линейных уравнений необходимо просуммировать также и якобианы всех моделей элементов. Решением полученной системы уравнений являются приращения текущих значений потенциальных переменных. Текущие значения потенциальных переменных уточняются, и, если это необходимо и возможно, осуществляется новая итерация.

Из сказанного вытекают функции модели элемента в составе комплекса PRADIS. Она должна по заданным значениям потенциальных переменных определить значения потоковых переменных и величины частных производных от потоков по потенциалам (якобиан элемента). Кроме того, всегда нужно иметь в виду, что в общем случае в пределах одного шага интегрирования обращение к каждой модели происходит НЕСКОЛЬКО раз (количество таких обращений соответствует количеству итераций метода Ньютона).

Модель элемента характеризуется некоторым количеством степеней свободы (или ветвей), которыми она связана с внешней средой - другими элементами, включенными в описание объекта. Такие степени свободы называются ВНЕШНИМИ. Кроме того, некоторые степени свободы могут использоваться для внутренних нужд модели - например, описание перемещения внутренних узлов, потенциалов внутренних точек, никак не связанных с другими элементами системы, степени свободы, использующиеся для интегрирования потоковой переменной и др. Такие степени свободы модели элемента называются ВНУТРЕННИМИ.

Процесс интегрирования по времени начинается с первого шага (NSTEP=1). Для инициализации (установки начальных значений) некоторых используемых моделью переменных, которую необходимо провести перед началом интегрирования, существует понятие нулевого шага (NSTEP=0).

На нулевом шаге интегрирования модель элемента может установить начальные значения:

### III - 2

- потенциальных переменных для тех степеней свободы, с которыми связана модель;

- элементов вектора состояния (NEW, OLD) и рабочего вектора модели (WRK), которые НЕ ЗАВИСЯТ ОТ ПАРАМЕТРОВ МОДЕЛИ.

#### 3.2. Структура оператора SUBROUTINE модели элемента.

Ниже рассмотрена модель механического элемента, как наиболее представительный из возможных случаев. Где это необходимо, сделаны оговорки относительно моделей из других предметных областей.

В наиболее общем случае оператор SUBROUTINE модели элемента выглядит следующим образом :

```
SUBROUTINE MODEL (I,Y,X1,...,XN,PAR,NEW,OLD,WRK)
```

```
REAL * 8 I(1),          Y(1), X1(1), ... XN(1)
```

```
REAL * 8 NEW(1), OLD(1),WRK(1)
```

Здесь MODEL - имя модели элемента. Задается по правилам языка FORTRAN (1-6 прописных букв латинского алфавита или цифр). Для моделей элементов, которые будут иметь свой графический образ по умолчанию, нужно учитывать правила формирования имени графического образа.

- I - вектор сил (моментов) для элемента. Размер вектора соответствует общему количеству степеней свободы модели элемента (количество внешних степеней свободы + количество внутренних степеней свободы). В моделях другой физической природы это вектор электрических токов, тепловых потоков, расходов. Модель элемента должна определить величину сил (моментов), действующих СО СТОРОНЫ СИСТЕМЫ НА ЭЛЕМЕНТ (в электронике - токи, втекающие в элемент). Силы в этом массиве должны располагаться в том порядке, в котором соответствующие степени свободы элемента будут перечисляться при описании топологии на входном языке комплекса.
- Y - якобиан модели элемента. Размерность якобиана -  $Y(N*N,3)$ , где N - количество степеней свободы с учетом внутренних узлов. Структура якобиана изображена на рис.3.2. Необходимо отметить, что в общем случае модель элемента при каждом обращении к ней должна определить ВСЕ элементы якобиана (так как этот массив используется всеми остальными моделями, принимающими участие в расчете). В некоторых

### III - 3

случаях (ключевые параметры паспорта ADR и IGN; информация о ключевых параметрах паспорта модели - см. ниже) можно избавить модель от ответственности за заполнение отдельных участков якобиана.

X1-XN - массивы потенциальных переменных. В общем случае количество таких массивов соответствует количеству степеней свободы элемента. Предполагается, что в начале списка потенциальных переменных располагаются внешние степени свободы, в конце - внутренние. Если не

заданы параметры паспорта ADR и/или IGN (см. далее), то элементы этих массивов соответствуют

X1 (1) - перемещению по первой степени свободы (в электронике - интегралу по времени от потенциала, в гидравлике - интегралу от давления и т.д.);

X1 (2) - скорости по первой степени свободы (потенциалу, давлению);

X1 (3) - ускорению по первой степени свободы (первой производной по времени от потенциала, давления);

XN (1...3) - перемещению, скорости и ускорению по N-й степени свободы.

**ВНИМАНИЕ !** Пользователь ни в каком случае не должен изменять значения потенциальных переменных внутри модели после нулевого шага интегрирования! НЕ НАДО.

PAR (M) - массив параметров модели (параметры подпрограмме передаются из текста описания структуры объекта). Количество параметров модели элемента определяется ее паспортом. Модель не должна изменять элементы этого вектора. Если этот массив в программе, реализующей модель элемента, описан обычным образом (например, PAR(1)), то ячейка памяти PAR (0) содержит количество параметров для этой модели. Это может понадобиться в случае реализации модели элемента с переменным количеством параметров. В таком случае ячейка памяти PAR (0) содержит фактическое количество параметров, передаваемых в модель элемента при данном вызове.

NEW (L), - элементы вектора "состояния" модели

OLD (L) элемента. Часто в ходе расчетов требуется накапливать какую-либо величину или устанавливать признак текущего состояния

### III - 4

объекта (например, разрушен - неразрушен) по результатам текущего шага интегрирования.

При этом никогда не известно, сколько итераций при решении системы нелинейных уравнений будет сделано (т.е., является ли данное обращение к модели элемента на данном шаге интегрирования последним, или нет). Для таких переменных удобно использовать вектор "состояния". При каждом очередном обращении в модель для I-й переменной состояния подсчитывается элемент массива NEW (I). В случае, если шаг будет успешно завершен, рабочая программа сама позаботится о пересылке вектора NEW в вектор OLD. Например, интегрирование какой-либо величины, прямо не связанной с определением вектора потока или якобиана, может происходить так :



IF (NSTEP .EQ. 0) OLD (1) = 0.D0

C

Текущее значение работы :  
 $NEW(1) = OLD(1) + STEP * X1(2)$

Во многом вектор состояния модели элемента ... для тех,--->  
перекликается с одноименным объектом в ПА6,

кто  
работал  
с ПА6 ...

за исключением того, что он НИКАК не влияет  
прямо на процесс интегрирования и программой  
интегрирования не используется и не  
модифицируется (кроме автоматической  
пересылки из NEW в OLD после успешного  
завершения шага интегрирования). Этот вектор  
- частная собственность модели элемента.  
Длина вектора состояния определяется сочетанием ключевых  
параметров паспорта модели элемента STR и STP. Если эти ключевые  
параметры определяют нулевую длину вектора состояния, то этот  
вектор отсутствует в списке формальных параметров модели элемента.

WRK (K) - рабочий массив для модели элемента. Разработчик модели  
элемента должен иметь  
ввиду, что в анализируемом техническом  
объекте может присутствовать большое  
количество одноименных элементов. Поэтому  
необходимо обеспечить ПОВТОРНУЮ входимость в  
элемент на шаге интегрирования. Нельзя в  
теле элемента хранить какие-либо величины,  
если они могут быть другими для других  
используемых одноименных элементов. Для  
хранения этих переменных используется массив  
WRK. Для каждого конкретного вызова модели элемента используется  
свой рабочий массив (этих массивов столько, сколько одноименных  
моделей элемента присутствует в описании объекта). По ходу решения  
элемент рабочего

### III - 5

вектора может быть передан в программу  
расчета выходных переменных с помощью  
указателя W:. Например, W:Тело(2) -  
передать в программу расчета выходной переменной второй элемент  
рабочего вектора модели элемента с идентификатором Тело.  
Рабочий вектор модели элемента имеет постоянную часть и  
переменную часть,  
зависящую от количества элементов в векторе параметров модели.  
Общая длина доступного в  
модели рабочего вектора определяется сочетанием ключевых  
параметров паспорта WRK и WRP. Если эти ключевые параметры  
паспорта определяют нулевую длину рабочего вектора, то этот вектор  
отсутствует в списке формальных параметров модели элемента.

### 3.3. Паспорт модели элемента.

Паспорт модели элемента располагается в первых строках программы, реализующей модель элемента в колонках со 2 по 72 включительно. Если он не умещается на одной строке, то для переноса на следующую строку последним символом текущей строки должна быть запятая.

Паспорт модели элемента начинается ключевым словом MODEL. После этого идет имя подпрограммы, реализующей модель элемента, и, после символа ":", - список имен ключевых параметров паспорта и их значений - положительных целых чисел. Имя ключевого параметра и его значение разделяются символом присваивания (=). Различные ключевые параметры в этом списке отделяются друг от друга запятыми.

В паспорте модели элемента могут быть такие ключевые параметры:

- EXT - количество внешних степеней свободы модели элемента ( $>0$ ). Этот ключевой параметр в паспорте элемента обязателен (нельзя сделать элемент с нулевым количеством внешних степеней свободы).
- ENT - количество внутренних степеней свободы модели элемента. Если этот ключевой параметр паспорта не задан, то предполагается  $ENT = 0$ ;
- PAR - определяет количество параметров для элементов с постоянным количеством параметров. Для элементов с переменным количеством параметров - минимально возможное количество параметров для этого элемента. Если ключевой параметр PAR не задан, количество параметров элемента принимается равным 1;

### III - 6

- VPR - признак того, что элемент имеет переменное количество параметров. Если VPR не задан или равен 0, считается, что элемент имеет постоянное количество параметров. Если  $VPR=1$ , то элемент имеет переменное количество параметров. При этом на этапе синтаксического анализа контролируется, чтобы заданное количество параметров элемента было не меньше ключевого параметра PAR. Разработчик может потребовать от транслятора дополнительного контроля количества параметров на четность и нечетность. Если количество параметров модели элемента должно быть нечетным, то ключевой параметр VPR задается равным 11, если четным - то 21;
- STR - количество элементов в независимой от количества параметров части вектора состояния модели элемента (по умолчанию = 0);
- STP - определяет, сколько элементов вектора состояния должно соответствовать каждому параметру в переменной части вектора параметров. Этот ключевой параметр полезен в моделях элементов с переменным количеством параметров, когда требуется для каждого параметра или группы параметров сохранять какую либо характеристику состояния. Например, контактный элемент, моделирующий контакт

эллипса с ломаной, использует переменную часть вектора состояния для сохранения текущего состояния элемента в каждой из контактных точек (учитывается тот факт, что эллипс может контактировать с ломанной в нескольких точках).

ПРИМЕЧАНИЕ.

Общая длина вектора состояния для каждого элемента вычисляется по зависимости

STR +

(<фактическое количество параметров>-PAR)\*STP

- WRK - количество элементов в постоянной части рабочего вектора модели (по умолчанию = 0).
- WRP - определяет, сколько элементов рабочего вектора должно соответствовать каждому параметру в переменной части вектора параметров. Этот ключевой параметр полезен в моделях элементов с переменным количеством параметров, когда требуется для каждого параметра или группы параметров сохранять результаты каких-либо однократных вычислений. Типичные модели, в паспорте которых используется этот ключевой параметр - контактный участок, модель кулачка, табличная зависимость усилия от деформации и т.д.

### III - 7

ПРИМЕЧАНИЕ.

Общая длина рабочего вектора, передаваемого модели элемента, вычисляется по формуле

WRK +

(<фактическое количество параметров>-PAR)\*WRP

- ADR - этот ключевой параметр дает возможность изменить правила вызова модели элемента. Если задано ADR=2, то считается, что якобиан модели "написан относительно скоростей". В этом случае якобиан имеет размерность  $Y(N \times N, 2)$ , при этом  $Y(I,1)$  соответствует производным по скоростям,  $Y(I,2)$  - производным по ускорениям. Соответственно, размерность каждого из векторов потенциальных переменных уменьшается на единицу.  $XJ(1)$  будет соответствовать скорости по J-й степени свободы,  $XJ(2)$  - ускорению. ADR=3 - следующий шаг в этом направлении. Размерность якобиана -  $Y(N \times N, 1)$ , все производные - по ускорениям, размерность вектора потенциальных переменных - тоже 1.  $XJ(1)$  (можно также записывать просто XJ, без указания индекса) - ускорение по J-й степени свободы. По умолчанию ADR=1.
- IGN - указание вычислительному ядру комплекса игнорировать соответствующие фрагменты якобиана. IGN=2 говорит о том, что элементы якобиана  $Y(I,2)$  являются нулевыми (свойства модели элемента не зависят от скорости), и этот участок якобиана моделью не заполняется. IGN=3 говорит о том, что нулевыми являются элементы якобиана  $Y(I,3)$ . IGN=23 говорит о том, что свойства модели элемента зависят только от перемещения. Важно иметь в виду, что этот ключевой параметр не влияет на фактическую размерность якобиана и его структуру. Так, для модели с IGN=2 в якобиане по-прежнему

присутствуют элементы, отражающие зависимость усилий от скорости, просто они считаются нулевыми и вычислительным ядром комплекса не анализируются

Возможные сочетания ADR и IGN :

ADR=1,IGN=0 - соответствует "полному" элементу, описанному в разделе 3.1;

ADR=1,IGN=2 - элемент обладает жесткостными и инерционными свойствами. При написании модели такого элемента не нужно заботиться об определении элементов якобиана, связанных со скоростью;

### III - 8

ADR=1,IGN=3 - элемент с жесткостными и вязкими свойствами. Не нужно определять элементы якобиана, связанные с ускорением;

ADR=1,IGN=23 - чисто жесткостной элемент. Не нужно определять элементы якобиана, связанные со скоростью и ускорением ;

ADR=2,IGN=0 - вязко-инерционная элемент. В якобиане отсутствуют элементы, отражающие зависимость сил от перемещений;

ADR=2,IGN=3 - элемент с вязкими свойствами.  
Элементы якобиана, отражающие зависимость сил от перемещений в якобиане отсутствуют, а элементы, отражающие зависимость от ускорений, определять не нужно;

ADR=3,IGN=0 - чисто инерционный элемент. В якобиане отсутствуют члены, отражающие зависимость сил от перемещений и скоростей.

По умолчанию IGN=0.

Важно отметить, что одна и та же программа при различных сочетаниях ADR и IGN может представлять различные элементы.

Пример :

```
SUBROUTINE  PROG (I, Y, X1, X2, PAR, ...)
REAL * 8    I(1), Y(1), X1, X2, PAR (1), K
K  =        PAR (1)
I(1) =       K * (X1 - X2)
I(2) =       K * (X2 - X1)
Y (1) =      K
Y (2) = -    K
Y (3) = -    K
Y (4) =      K
RETURN
```

END

Если в паспорте этой модели указано  $ADR=1$ ,  $IGN=23$ , то это - модель линейно упругого одномерного элемента, соединяющего два тела. Если  $ADR=2$ ,  $IGN=3$ , то это - модель линейно вязкого одномерного элемента, соединяющего два тела, если  $ADR=3$ , то это инерционный элемент. В механике для моделирования точечного инерционного элемента этот элемент должен стоять одной "ногой" на земле. В наибольшей степени соответствует электрической емкости.

### III - 9

**ВНИМАНИЕ !** Как следует из приведенной выше информации, эту программу нельзя считать моделью элемента при таких сочетаниях ключевых параметров  $ADR$  и  $IGN$  :

$ADR=1$ ,  $IGN=0$ ,  
 $ADR=1$ ,  $IGN=2$ ,  
 $ADR=1$ ,  $IGN=3$ ,  
 $ADR=2$ ,  $IGN=0$ ,

так как она не заполняет соответствующие участки якобиана.  
Т.е., при таких сочетаниях паспортных параметров  $IGN$  и  $ADR$  модель является некорректной.

#### 3.4. Справочная информация по модели элемента.

Рекомендуются следующие разделы справочной информации по модели элемента:

- 1) первая строка справочной информации, содержащая краткое название модели элемента (попадает в сокращенный HELP по модели);
- 2) полное название модели элемента;
- 3) область применения;
- 4) описание степеней свободы модели;
- 5) описание вектора параметров;
- 6) описание тех элементов рабочего вектора, которые могут быть полезны пользователю.

#### 3.5. Особенности вычислительного ядра, на которые нужно обратить внимание при разработке модели элемента.

Кроме этого подраздела об особенностях разработки модели элемента говорится также в следующем подразделе, на примере разработки конкретной модели элемента.

##### 3.5.1. Проверка параметров на допустимость, начальные инициализации рабочего вектора и вектора состояния.

Свойства модели элемента определяются ее параметрами. Поэтому разработчик модели должен уделить особое внимание:

- 1) проверке параметров модели на допустимость;
- 2) подсчету тех или иных величин, использующих значения параметров модели. Наиболее типичный случай - это

## III - 10

однократный подсчет элементов рабочего вектора модели, зависящих от параметров.

В ходе разработки модели элемента необходимо предусматривать возможность внезапной смены параметров по ходу интегрирования. Здесь мы отвлекаемся от последствий такой смены параметров для текущего расчета (например, удар страшной силы по механической системе можно вызвать, хорошенько сжав стержень небольшой жесткости и затем увеличив его жесткость на несколько порядков). Что касается входного языка PRADIS, то это обеспечивается использованием заголовка \$REPLACE в сочетании с заголовком \$RESTORE.

Поэтому разработчик модели элемента должен предусмотреть все необходимые проверки параметров на допустимость и необходимые однократные вычисления ПРИ КАЖДОМ НОВОМ ВЫЗОВЕ ПРОГРАММЫ ИНТЕГРИРОВАНИЯ.

Нужно подчеркнуть здесь, что условия

IF (NSTEP .EQ. 0)                      И IF (NEWINT .EQ. 1)

не идентичны. Первое из них выполняется только в самом начале расчета, на "нулевом шаге интегрирования". Второепри каждом новом вызове программы интегрирования. Например, выполняется такое задание на расчет:

\$ RUN:  
Первый этап интегрирования' SHTERM (END=1)  
Второй этап интегрирования' SHTERM (END=1.5)  
Третий этап интегрирования' SHTERM (END=2)

В этом случае условие IF (NEWINT .EQ. 1) будет по ходу расчета выполняться три раза. Кроме того, оно выполняется всякий раз, когда пользователь запускает задание для уже сформированной модели, в котором присутствует заголовок \$RESTORE.

Таким образом, использование в модели элемента для входа в блок однократных вычислений проверки

IF (NEWINT .EQ. 1)

страхует от неприятностей, связанных с совместным использованием заголовков \$REPLACE и \$RESTORE.

Диагностика неверного значения того или иного параметра осуществляется самой моделью и должна сопровождаться установкой соответствующего кода возврата (см. выше описание COMMON переменной CODE. При неверном значении параметра модели элемента

задают CODE=100). Получение такого сигнала вычислительным ядром комплекса PRADIS приводит к немедленному прекращению анализа с выдачей соответствующего диагностического сообщения в файл SYSPRINT.TXT.

### III - 11

Пример фрагмента модели элемента, где осуществляется проверка допустимости значения одного из параметров.

```

IF (NEWINT .EQ. 1) , THEN
    IF (PAR (1) .LT. 0.D0) , THEN
C
C                Параметр PAR(1) при новом вызове
C                программы интегрирования имеет
C                недопустимое значение.
C                IF ( CODE .LT. 100) ,                THEN
C                CODE      = 100
C                NAME      = NAMEMD
C                END IF C
C                Сообщение об ошибке и аварийный выход
C                из модели элемента.
C
C                . . .
C                RETURN
C            ELSE
C
C                Однократный подсчет элементов
C                рабочего вектора, зависящих
C                от PAR(1) и другие однократные
C                вычисления :
C                WRK (1) = PAR (1) ** 2
C
C                . . .
C            END IF END IF

```

#### 3.5.2. Особенности моделей элементов, задающих начальные условия.

В комплексе PRADIS имеется группа моделей элементов, предназначенных для задания начальных значений потенциальных переменных (начальных скоростей и начальных перемещений). Эту операцию можно осуществить ТОЛЬКО НА НУЛЕВОМ ШАГЕ ИНТЕГРИРОВАНИЯ. На последующих шагах интегрирования модель НЕ МОЖЕТ ИЗМЕНЯТЬ ЗНАЧЕНИЯ ПОТЕНЦИАЛЬНЫХ ПЕРЕМЕННЫХ, передаваемых в нее в качестве параметров. Такие попытки приведут к непредсказуемым последствиям. Кроме того, моделью элемента должна учитываться внезапно пришедшая пользователю мысль в ходе расчета изменить начальную скорость (или начальное перемещение).

При инициализации начальных значений потенциальных переменных модель элемента должна сама заботиться о том, чтобы не произошло противоречивой установки значений потенциальной переменной. Т.е., нельзя переустанавливать значения потенциальной переменной, уже установленного другой моделью. Диагностика такой ситуации осуществляется самой моделью и должна сопровождаться установкой соответствующего значения кода

### III - 12

возврата (см. выше описание COMMON переменной CODE. При противоречивой установке потенциала задают CODE=90). Получение такого сигнала вычислительным ядром комплекса PRADIS приводит к немедленному прекращению анализа с выдачей соответствующего диагностического сообщения в файл SYSPRINT.TXT.

Пример фрагмента модели элемента, у которой в качестве параметра задается начальное значение скорости первой степени свободы.

```

      IF (NEWINT .EQ. 1) , THEN
        V0 = PAR (1)
        IF (NSTEP .EQ. 0) , THEN
C          Вычисления на НУЛЕВОМ ШАГЕ
C          ИНТЕГРИРОВАНИЯ
          IF ( ABS (X1(2)          .GT. RLMIN .AND.
            , ABS (V0-X1(2)) .GT. RLMIN          )
            , THEN
              IF ( CODE .LT. 90 ) ,
THEN
                  CODE = 90
                  NAME = NAMEMD
                  END IF
C          Сообщение об ошибке и
C          аварийный выход из модели
C          элемента.
          . . .
          RETURN
        ELSE
          X1 (2) = V0
          WRK (1) = V0 END IF
        ELSE
C          Вычисления при новом входе в программу
C          интегрирования, но НЕ НА НУЛЕВОМ ШАГЕ
C          ИНТЕГРИРОВАНИЯ.
          IF (WRK(1)          .NE. V0)
            , THEN
C          Попытка переустановить
C          начальную скорость по ходу
C          расчета.
              IF ( CODE .LT. 100) ,
THEN
                  CODE = 100
                  NAME = NAMEMD
                  END IF
C          Сообщение об ошибке и выход из
C          программы.
          . . .
          RETURN END IF
        END IF END IF

```



### 3.5.3. Учет возможности использования различных схем интегрирования и обеспечение повторной входимости в модель элемента.

Вследствие того, что PRADIS для анализа процессов во временной области может использовать различные схемы интегрирования, использование шага интегрирования (COMMON переменная STEP) для определения потоков и вычисления якобиана внутри модели не желательно (если говорить откровенно - то запрещено. Единственное исключение - использование текущего шага интегрирования для аппроксимации производных по времени выше второго порядка).

Еще раз нужно обратить внимание пользователя на такие факторы, как МНОГОКРАТНОЕ обращение к моделям элементов на каждом шаге интегрирования и возможное многократное вхождение одной и той же модели элемента в структуру модели объекта (а значит, и многократное обращение к моделям элементов в пределах одной итерации метода Ньютона). Это приводит к двум важным особенностям программирования такого рода модулей :

- 1) нужно быть осторожными с операторами типа

$$ALFA = ALFA + DALFA,$$

поскольку в пределах одного шага интегрирования он может выполняться неопределенное количество раз. В таких или похожих целях используется вектор состояния модели элемента (см. пример, приведенный выше при описании вектора состояния модели).

- 2) хранение каких-либо величин в сохраняемых (SAVE) локальных переменных также может привести к ряду проблем из-за того, что одна и та же модель может использоваться в структуре объекта многократно. Например, в описании какой-либо структуры модель стержня может присутствовать несколько раз, у каждой из этих моделей может быть различная начальная длина. Тогда, если разработчик модели элемента хочет сохранить начальную длину стержня, он не может просто вычислить ее и запомнить в SAVE переменной из-за того, что при следующем вхождении в модель она будет перевычислена для другого сочетания параметров - в модели объекта может присутствовать большое количество стержней. В этом случае используется так называемый рабочий вектор модели элемента. Для каждого вхождения модели элемента в модель объекта заводится своя уникальная копия рабочего вектора.

Для выполнения вычислений и хранения промежуточных величин модель элемента может использовать рабочий вектор и вектор состояния.

### 3.5.4. Некоторые другие важные особенности моделей элементов.

1) Поскольку нулевой шаг не является полноценным шагом интегрирования, то вычисление якобиана элемента при  $NSTEP=0$  можно не проводить. Однако, потоковые переменные рекомендуется вычислять (они могут попасть в файл результатов, в который на нулевом шаге проводится запись начального состояния выходных переменных).

2) Разработчик модели элемента должен обратить внимание на то, что величины усилий, генерируемых моделью элемента при отсутствии внешнего воздействия, должны быть равны нулю.

3) В моделях элементов делается большое количество промежуточных вычислений. Кроме того, часто без особых затрат можно подсчитать те или иные величины, которые могут оказаться полезными как выходные переменные (силы и моменты в локальной системе координат, напряжения и т.д.). Разработчик должен обратить внимание на этот факт и обеспечить дополнительные удобства пользователю его модели. Как правило, элементы рабочего вектора, предназначенные к использованию в качестве выходных переменных, должны находиться в начале рабочего вектора и должны быть описаны в HELP'e по модели.

### 3.6. Пример разработки модели элемента.

#### 3.6.1. Постановка задачи.

Рассмотрим здесь процесс разработки и включения в состав комплекса PRADIS новой модели элемента.

Предположим, что перед пользователем стоит задача анализа системы из двух тел, совершающих плоское движение. Каждое из тел характеризуется своей массой и моментом инерции относительно центра тяжести. Центры тяжести двух тел связаны идеально упругой связью, работающей на растяжение - сжатие. Ко второму телу вертикально вниз приложена сила. Схема этой системы изображена на рис. 3.3.

Допустим, что мы не обнаружили в библиотеках комплекса никакого элемента, позволяющего реализовать идеальную упругую связь между телами для плоского движения, и уверены, что этого элемента там действительно нет (в противном случае лучше воспользоваться консультацией знающего человека). Результаты декомпозиции системы на элементы иллюстрируются рис.3.4. Выделены следующие элементы, необходимые для формирования модели :

- точечный инерционный элемент, совершающий плоское движение (MD,- Тело 1, Тело 2);
- источник постоянной силы (F, Сила);

### III - 15

- идеально упругий элемент, движущийся в плоскости и работающий на растяжение - сжатие (??? , Идеальная упругая связь).

В качестве необходимого комментария добавим, что здесь рассматривается только гипотетический пример для более подробного знакомства с включением в состав комплекса модели элемента. Если же в вашем комплексе действительно нет подобного элемента (скажем, элементы группы стержней), то можете быть уверены, что он находится в адамовом (т.е., совершенно голом) состоянии.

Итак, перед нами во весь рост встала задача разработки модели элемента, реализующего идеальную упругую осевую связь (т.е., силовые факторы действуют вдоль оси элемента) для плоского движения - т.е., модель пружины растяжения-сжатия.

### 3.6.2. Основные зависимости для модели элемента и ее параметры.

Как следует из информации, приведенной в этой главе, модель разрабатываемого нами элемента по заданным значениям перемещений центров масс двух соединяемых ею тел должна определить величины сил, с которыми СИСТЕМА ДЕЙСТВУЕТ НА ЭЛЕМЕНТ и частные производные от сил по перемещениям.

Мы собираемся довести дело до создания подпрограммы, реализующей модель этого элемента. Поэтому, для того, чтобы можно было легко разобраться с текстом программы, ниже во всех выкладках будем использовать обозначения переменных и параметров, которые в дальнейшем войдут в текст программы.

Величину действующей на элемент осевой силы можно определить из закона Гука:

$$F = U * K \quad (3.1)$$

, где  $U$  - деформация растяжения элемента,  
 $K$  - коэффициент упругости,  
 $F$  - осевая растягивающая сила.

На каждый из концов пружины будут действовать силы, равные по абсолютной величине  $F$  и взаимно противоположно направленные. Данные, необходимые для их определения :

- начальные координаты концов пружины (задаются в качестве параметров модели).  
Начальные координаты точки  $A$  обозначим  $XA0, YA0$ , точки  $B$  -  $XB0, YB0$ ;
- перемещения пружины по каждой из степеней свободы :  
 $X1$  - перемещение точки  $A$  от начального положения в направлении оси  $X$ ,  $X2$  - перемещение точки  $A$  от начального положения в направлении оси  $Y$ ,  $X3$  - перемещение точки  $B$  от начального положения в

### III - 16

направлении оси  $X$ ,  $X4$  - перемещение точки  $B$  от начального положения в направлении оси  $Y$ . Таким образом, модель элемента будет иметь четыре степени свободы;

- коэффициент жесткости (задается в качестве параметра модели).

Расчетная схема пружины и зависимости для определения величин усилий, действующих на пружину со стороны системы по каждой из степеней свободы, приводятся на рис. 3.5.

### 3.6.3. Якобиан элемента.

В основе зависимостей для силовых факторов, полученных в предыдущем пункте, лежит закон Гука. Разрабатываемый элемент не обладает инерционными и вязкостными свойствами. Поэтому необходимо и достаточно вычислить только ту часть якобиана элемента, которая отражает зависимость сил от перемещений. Это массив из 16 элементов (производные каждой из четырех сил по четырем перемещениям). Тот факт, что производные от сил по скоростям и ускорениям равны нулю, отразим соответствующим определением элементов паспорта ADR и IGN.

Вспомним, что компонентами якобиана являются ЧАСТНЫЕ производные от сил по соответствующим потенциальным переменным (это означает, в частности, что  $dX_i/dX_1=0$ ). Для хранения якобиана используем одномерный массив. Тогда первые четыре элемента этого массива - производные от первой силы по соответствующим перемещениям, следующие четыре элемента - производные от второй силы и т.д.

В окончательных выражениях для якобианов использованы результаты нескольких промежуточных выкладок :

$$\begin{aligned} 1) \quad dL/dX_1 &= (\text{SQRT}((XB-XA)**2+(YB-YA)**2))' = \\ &= 2*(XB-XA)*(XB-XA)' / 2L \end{aligned}$$

Учитывая, что  $dXA/dX_1 = 1$ ;  $dXB/dX_1 = 0$ , получим:

$$dL/dX_1 = -(XB-XA)/L = -\cos(\alpha)$$

Аналогично

$$\begin{aligned} dL/dX_2 &= -\sin(\alpha) \\ dL/dX_3 &= -dL/dX_1 \\ dL/dX_4 &= -dL/dX_2 \end{aligned}$$

$$\begin{aligned} 2) \quad d\cos(\alpha)/dX_1 &= ((XB-XA)/L)' = \\ &= (-L - dL/dX_1*(XB-XA)) / L**2 = \\ &= (-1 + \cos(\alpha)*\cos(\alpha)) / L = \\ &= -\sin(\alpha)**2 / L \\ d\cos(\alpha)/dX_2 &= (0 - dL/dX_2*(XB-XA)) / L**2 = \\ &= \sin(\alpha)*\cos(\alpha) / L \end{aligned}$$

### III - 17

$$\begin{aligned} d\sin(\alpha)/dX_1 &= \cos(\alpha)*\sin(\alpha) / L \\ d\sin(\alpha)/dX_2 &= -\cos(\alpha)**2 \quad / L \end{aligned}$$

$$\begin{aligned} d\cos(\alpha)/dX_3 &= -d\cos(\alpha)/dX_1 \\ d\cos(\alpha)/dX_4 &= -d\cos(\alpha)/dX_2 \\ d\sin(\alpha)/dX_3 &= -d\sin(\alpha)/dX_1 \\ d\sin(\alpha)/dX_4 &= -d\sin(\alpha)/dX_2 \end{aligned}$$

Таким образом, якобиан разрабатываемого элемента выглядит так:

$$\begin{aligned} Y(1) = dF_1/dX_1 &= -K * (dL/dX_1 * \cos(\alpha) + \\ &\quad + U * d\cos(\alpha)/dX_1) = \\ &= -K * (-\cos(\alpha)**2 - \\ &\quad - U/L * \sin(\alpha)**2) = \\ &= -K * (-1 + L_0/L * \sin(\alpha)**2) \end{aligned}$$

$$\begin{aligned}
Y(2) &= dF1/dX2 = -K * (dL/dX2 * \cos(\alpha) + \\
&\quad + U * d\cos(\alpha)/dX2) = \\
&= -K * (-\sin(\alpha) * \cos(\alpha) + \\
&\quad + U/L * \sin(\alpha) * \cos(\alpha)) = \\
&= K * (L0/L * \sin(\alpha) * \cos(\alpha)) \\
Y(3) &= -Y(1) \\
Y(4) &= -Y(2) \\
Y(5) &= dF2/dX1 = -K * (dL/dX1 * \sin(\alpha) + \\
&\quad + U * d\sin(\alpha)/dX1) = \\
&= K * (L0/L * \sin(\alpha) * \cos(\alpha)) \\
Y(6) &= dF2/dX2 = -K * (-1 + L0/L * \cos(\alpha)^2) \\
Y(7) &= -Y(5) \\
Y(8) &= -Y(6)
\end{aligned}$$

Исходя из факта, что  $F3 = -F1$ , а  $F4 = -F2$ , производные по перемещениям от третьей силы будут равны соответственным производным от первой силы с противоположным знаком, а производные от четвертой силы - соответственным производным от второй силы. Кроме того, учитывая зависимость для определения текущей длины пружины, имеем  $dF_n/dX1 = -dF_n/dX3$ ,  $dF_n/dX2 = -dF_n/dX4$ .

Самая сложная и ответственная часть работы над моделью элемента - получение его якобиана - завершена.

### 3.6.4. Особые ситуации.

Важным вопросом при обеспечении вычислительной надежности модели элемента является вопрос идентификации особых ситуаций. Обычно он распадается на два случая - диагностика ошибок в параметрах элемента и диагностика ошибок по ходу вычислений (мы полагаем, что пользователь, как и авторы, стремится не получать прерываниями "по морде", т.е., избегает делить на 0, извлекать корни из отрицательных чисел и совершать другие подобные вызывающие поступки).

## III - 18

1) Параметры модели разрабатываемого нами элемента могут содержать ошибку в двух случаях - когда они определяют элемент нулевой длины или задают отрицательную жесткость.

Строго говоря, нулевая начальная длина элемента как таковая не является ошибкой, поскольку  $L0$  в наших выкладках нигде не присутствует в знаменателе. Более серьезным является тот факт, что если такой элемент не будет деформироваться на первой же итерации, то его длина по-прежнему останется нулевой. В приведенных зависимостях текущая длина присутствует в знаменателях полученных выражений. Т.е., этот случай должен обрабатываться отдельно, с использованием специально полученных зависимостей. Учитывая, что в подавляющем большинстве случаев эти возможности элемента использоваться не будут, принято решение запретить нулевую начальную длину стержня.

Весьма опасным с точки зрения вычислений является случай отрицательной жесткости. Такой элемент, получив любую, сколь угодно малую деформацию, будет генерировать силу, направленную на ее увеличение. И чем больше будет деформация, тем больше будет эта сила. Механических устройств с такими свойствами в реальной жизни не встречается.

2) В ходе вычислений может быть единственная особая ситуация - нулевая текущая длина элемента.

### 3.6.5. Выбор имени модели пружины.

Важным является вопрос выбора имени модели элемента. При включении новой модели элемента в состав комплекса PRADIS необходимо иметь ввиду следующие соображения:

1) Имя модели элемента должно быть мнемоничным, т.е., по возможности отражать ее назначение. При этом не следует забывать, что стандарт FORTRANa допускает в имени подпрограммы использовать не более 6 символов.

2) Имя модели элемента должно позволять получить из него имя графического образа по умолчанию (правила образования имен графических образов, соответствующих модели элемента по умолчанию, см. в главе 5 настоящего руководства). Если иметь ввиду эти правила, то в модели элемента запрещено :

- если имя модели элемента содержит 4 или 5 символов, использовать в качестве последнего символа цифру (из-за того, что при образовании имени графического образа по-умолчанию имя такой модели элемента реверсируется);
- если имя модели элемента содержит 6 символов, то нельзя использовать цифру в качестве 5 символа имени (так как перед реверсированием такого имени последний символ отбрасывается).

## III - 19

Исходя из правил формирования имени графического образа, всем моделям элементов, пять первых символов имени которых совпадают, соответствует по умолчанию один и тот же графический образ. Это сделано для обеспечения возможности реализации графических образов, соответствующих группам похожих элементов. Поэтому если вы даете модели элемента имя, настолько похожее на имя модели уже присутствующего или будущего элемента, нужно обязательно принимать этот факт во внимание.

3) Имя модели элемента не должно пересекаться с именами других модулей, включенных в расширяемые библиотеки комплекса (их список можно получить по команде ARM ?). Полезно получить также из предполагаемого имени модели имя графического образа, который будет соответствовать этой модели элемента по-умолчанию (см. главу 5), и проследить, чтобы оно также не пересекалось с именами модулей, включенных в расширяемые библиотеки комплекса.

4) Имя модели элемента не должно пересекаться с именами подпрограмм вычислительного ядра комплекса. Список зарезервированных имен приводится в Приложении 2 к настоящему документу.

5) Имя модели элемента не должно пересекаться с именами COMMON-блоков, используемых вычислительным ядром комплекса : NOTAT, GRCONF, FLDSR.

6) В качестве имени модели элемента нельзя использовать имена V00 - V99.

Для нашего примера выберем имя модели элемента LINKD - ЛИН(=LIN)ейный упругий(K) элемент, совершающий плоское(D) движение. В дальнейшем всем моделям

элементов, имена которых будут начинаться символами LINKD, будет по умолчанию соответствовать один и тот же графический образ - DKNIL. Его имя также не пересекается с зарезервированными именами вычислительного ядра и именами, включенными в настоящее время в библиотеки комплекса.

### 3.6.6. Паспорт модели элемента.

Разрабатываемый элемент имеет четыре степени свободы (EXT=4). Параметры элемента - начальные координаты концов и коэффициент жесткости (PAR=5). Чтобы не вычислять начальную длину элемента на каждой итерации, при первом обращении к модели запомним начальную длину в первом элементе рабочего вектора. Кроме того, в ходе расчетов для тех или иных задач может быть полезно вычислять текущую длину элемента и осевую силу. Запомним их, соответственно, во втором и третьем элементах рабочего вектора (т.е., WRK=3). Якобиан модели элемента записан относительно перемещений (ADR=1) и не зависит от скоростей и ускорений (IGN=23). Ключевые параметры VPR, STP, WRP для этой модели равны 0, поэтому в паспорте их можно не задавать.

## III - 20

### 3.6.7. Текст модели элемента.

Проделав подготовительную работу, приступаем непосредственно к кодированию текста программы. Необходимые комментарии к нему :

1) Имя модели элемента лучше запоминать в качестве символьного параметра, значение которому присваивать в начале программы (в непосредственной близости от оператора SUBROUTINE). Тогда, при разработке элементов с похожими функциями, когда текст модели LINKD берется в качестве прототипа, снижается вероятность того, что разработчик забудет изменить операторы, пересылающие имя модели в переменную NAME в особых ситуациях. Ошибка такого рода является достаточно распространенной.

2) Начальная длина элемента связи (расстояние между его концами) подсчитывается при первом обращении к модели элемента и запоминается в WRK(1).

3) Имеется только 4 элемента якобиана, которые необходимо подсчитывать. Остальные получаются, исходя из соображений, что  $F(3)=-F(1)$ ,  $F(4)=-F(2)$ ,  $dFn/dX1=-dFn/dX3$ ,  $dFn/dX2=-dFn/dX4$ .

4) В некоторых случаях удобно вызывать модель элемента в качестве "подмодели" из других моделей элементов - "суперэлементов". Тогда бывает нежелательным печатать из подмодели сообщения о неверных параметрах или аварийных ситуациях. Блокирования печати удобно добиваться обнулением номера устройства для вывода диагностических сообщений при вызове подмодели из суперэлемента. Например :

...

SYSOLD = SYSPRN

SYSPRN = 0

CALL LINKD (F, Y, X1, X2, X3, X4, PAR, WRK) C

C Не забыть восстановить значение переменной SYSPRN !

SYSPRN = SYSOLD

...

Для того, чтобы обеспечить эту возможность, перед печатью любого диагностического сообщения в файл сообщений осуществляется проверка переменной SYSPRN на неравенство нулю.

5) Чтобы ваша модель не выглядела белой вороной среди других моделей, и сообщение, выдаваемое моделью, хорошо ложилось в SYSPRINT.TXT, необходимо при выдаче сообщения всегда соблюдать такой его формат, как в приведенных примерах. И не забывать, что на первом месте в сообщении всегда идет имя модели, которое выдает это сообщение.

### III - 21

```
C MODEL LINKD: EXT=4, PAR=5, WRK=3, ADR=1, IGN=23
C
C                               Дата создания:           03/14/92 01:34pm
C                               Дата последней корректировки: 07/21/94 01:43am
C
C HELP Идеальная упругая осевая связь для плоского движения
C HELP НАЗВАНИЕ:             Идеальный упругий элемент для случая
C HELP                       плоского движения, работающий на
C HELP                       растяжение-сжатие.
C
C HELP ОБЛАСТЬ ПРИМЕНЕНИЯ : Механика.
C
C HELP СТЕПЕНИ СВОБОДЫ:
C HELP 1 - поступательная в направлении оси OX т. А элемента;
C HELP 2 - поступательная в направлении оси OY т. А элемента;
C HELP 3 - поступательная в направлении оси OX т. В элемента;
C HELP 4 - поступательная в направлении оси OY т. В элемента.
C
C HELP ПАРАМЕТРЫ:
C HELP 1 - начальная абсцисса точки А элемента;
C HELP 2 - начальная ордината точки А элемента;
C HELP 3 - начальная абсцисса точки В элемента;
C HELP 4 - начальная ордината точки В элемента.
C HELP      П Р И М Е Ч А Н И Е. Точки А и В в начальном
C HELP      положении не должны совпадать;
C HELP 5 - коэффициент жесткости (>=0).
C
C HELP ЭЛЕМЕНТЫ РАБОЧЕГО ВЕКТОРА :
C HELP 1 - начальная длина элемента;
C HELP 2 - текущая длина элемента;
C HELP 3 - текущая величина осевого усилия.
C HELP
C HELP ОСОБЫЕ СИТУАЦИИ :
C HELP В случае, если в ходе вычислений текущая длина элемента
C HELP становится близкой к 0, осуществляется аварийный выход
C HELP из модели.
C
C                               SUBROUTINE LINKD (F, Y, X1, X2, X3, X4, PAR, WRK) C
C                               Имя модели
```





```

C          L0      = DSQRT ((XA0-XB0)**2 + (YA0-YB0)**2)
C
C          Проверка значений параметров на допустимость:
C
C          IF(L0      .LE. RLMIN .OR.
.          K      .LT. 0.D0 )
C          ,
C          THEN
C          =====> EXIT      Аварийный выход из модели :
C          IF(CODE .LT. 100 ) ,      THEN
C          CODE      = 100
C          NAME      = NAMEMD
C          END IF
C          Печать в SYSPRINT сообщения об
C          ошибке в параметрах :
C          IF(SYSPRN.GT.0) ,      THEN

III - 23

C          IF(L0 .LE. RLMIN) ,      THEN
C          WRITE(SYSPRN,900) NAMEMD, , XA0, YA0, ,
XB0, YB0
C          END IF
C          IF(K .LT. 0.D0 ) , THEN
C          WRITE(SYSPRN,901) NAMEMD, K END IF
C          END IF
C          RETURN
C          END IF
C          Конец проверки допустимости параметров
C          WRK (1)= L0 C
C          END IF
C          Конец однократных вычислений в программе.
C
C          Вычисления при каждом обращении к программе
C          L0      = WRK (1)
C
C          Текущие координаты концов элемента:
C          XA      = X1(1) + XA0
C          YA      = X2(1) + YA0
C          XB      = X3(1) + XB0
C          YB      = X4(1) + YB0
C
C          Текущая длина :
C          L      = DSQRT ( (XA-XB)**2 + (YA-YB)**2)
C          Если длина близка к 0, то аварийный выход из модели
C          IF (L .LE. RLMIN ) , THEN
C
C          =====> EXIT      Аварийный выход из модели :
C          IF ( CODE .LT. 75 ) ,
C          THEN
C          CODE      = 75
C          NAME      = NAMEMD
C          END IF

```

```

                IF (SYSPRN.GT.0), THEN
                    WRITE(SYSPRN,902) NAMEMD END IF
                RETURN
            END IF
C
C      Направляющие косинусы оси элемента
COSAL = (XB-XA) / L
SINAL = (YB-YA) / L C
C      Деформация стержня :
U      = L - L0
C
C      Усилия по осям координат :
F (1) = - K * U * COSAL
F (2) = - K * U * SINAL

```

### III - 24

```

F (3) = - F (1)
F (4) = - F (2) C
C      Вычисление элементов якобиана :
Y (1) = - K * (- 1.D0 + L0/L * SINAL*SINAL)
Y (2) = K * (L0/L * SINAL*COSAL)
Y (3) = - Y (1)
Y (4) = - Y (2)
C
Y (5) = K * (L0/L * SINAL*COSAL)
Y (6) = - K * (-1.D0 + L0/L * COSAL*COSAL)
Y (7) = - Y (5)
Y (8) = - Y (6)
C
Y (9) = - Y (1)
Y (10) = - Y (2)
Y (11) = - Y (3)
Y (12) = - Y (4)
C
Y (13) = - Y (5)
Y (14) = - Y (6)
Y (15) = - Y (7)
Y (16) = - Y (8)
C
C      Оставшиеся элементы рабочего вектора (текущая длина
C      и осевая сила со своим знаком)
WRK (2) = L
WRK (3) = U * K
RETURN
900  FORMAT (/T4,A8,',',
      ,      T15,'Ошибка в координатах элемента,'
      ,      /T15,'приводящая к нулевой длине : '
      ,      /T15,'координаты точки A =',2(G11.5,2X),
      ,      /T15,'координаты точки B =',2(G11.5,2X))
901  FORMAT (/T4,A8,',',
      ,      T15,'Ошибочное задание жесткости.'
      ,      /T15,'жесткость =',G11.5)
902  FORMAT (/T4,A8,',',
      ,      T15,'Длина элемента в ходе вычислений близка к 0.')
```

END

### 3.6.8. Автономное тестирование модели элемента.

Очень важным этапом в разработке модели элемента является автономное тестирование разработанной модели. Из приведенной выше информации становится ясно, что наиболее сложная и ответственная часть программы, реализующей модель элемента - это ее якобиан. Даже для такой простой модели, как в разобранный примере, его получение представляет определенные трудности. Прямым следствием ошибки при вычислении якобиана является несходимость при решении системы нелинейных уравнений. В лучшем случае, если ошибка "незначительная", программа интегрирования будет сильно дробить величину шага и терять большое количество шагов. В худшем - процесс вычислений просто оборвется.

## III - 25

Поэтому автономное тестирование модели элемента является необходимым шагом в ее разработке. Как правило, для этого пишется несложная головная программа на FORTRANe, задача которой - инициализация соответствующих элементов COMMON-блока, ввод параметров модели элемента и вызов модели элемента при определенных значениях перемещений (скоростей, ускорений). После этого по каждой степени свободы делаются небольшие приращения перемещений (скоростей, ускорений), и вычисляются численные значения элементов якобиана. Вычисленные численно значения якобиана сравниваются с рассчитанными моделью в исходной точке. Естественно, что они будут несколько различаться. Однако, это отличие для правильно вычисленных якобианов 1) должно быть незначительным и 2) должно уменьшаться при уменьшении приращения перемещения (скорости, ускорения).

Приведем пример такой программы, использованной для тестирования модели элемента LINKD :

```
      REAL      * 8 I(4),          I1(4), Y(16,3), YTMP(16,3)
      REAL      * 8 X1(3),          X2(3), X3(3),          X4(3)
      REAL      * 8 PAR(0:5), WRK(3)

C
      REAL      * 8 DDX,          Y1,          Y2
      INTEGER * 4 NUM,          LLL

C
C      Неименованный COMMON - блок :
C
      REAL      * 8 TIME, STEP, STEP01, STEP02, SMIN,
      VAL001, VAL002, STEPMD, TIMEND
      CHARACTER * 8 NAME
      INTEGER * 4 NSTEP, SYSPRN, NITER, ITR
      INTEGER * 2 CODE, NUMINT, NUMPRV, CODSTP, CODGRF, , NEWINT, MINSTP

C
C      непоименованная COMMON-область
COMMON      TIME, STEP, STEP01, STEP02, SMIN,
      VAL001, VAL002, STEPMD, TIMEND, NAME,
      NSTEP, SYSPRN, NITER, ITR,          CODE,
      NUMINT, NUMPRV, CODSTP, CODGRF,
      NEWINT, MINSTP

C
C      Переменные поименованной COMMON-области /NOTAT/
      REAL      * 8 RLMAX, RLMIN, INTMAX, MSHEPS,
```

```

C      ,          PI,          REZA, REZB,          REZC
C      Поименованная COMMON-область /NOTAT/
COMMON /NOTAT/
C      ,          RLMAX,          RLMIN,          INTMAX, MSHEPS,
C      ,          PI,          REZA,          REZB,          REZC
C
C      Начальные инициализации элементов COMMON-блоков
C      Непоименованный COMMON - блок
TIME      = 0.      D0
STEP      = 0.      D0
STEP01    = 0.      D0
STEP02    = 0.      D0
SMIN      = 1.      D-10

```

### III - 26

```

VAL001    = 0.      D0
VAL002    = 0.      D0
STEPMD    = 0.01 D0
TIMEND    = 0.1     D0
NAME      = '      '
NSTEP     = 0
SYSPRN    =10
NITER     = 0
ITR       = 7
CODE      = 0
NUMINT    = 1
NUMPRV    = 1
CODSTP    = 1
CODGRF    = 0
NEWINT    = 1
MINSTP    = 1
C      COMMON - область NOTAT
PI        = DACOS (-1.D0)
RLMIN     = 1.D-100
MSHEPS    = 1.D-14
C
OPEN (FILE="SYSPRINT.TXT",UNIT=SYSPRN)
C
20      CONTINUE
C      Ввод параметров модели элемента
PAR (0) = 5.
PRINT *, ' XA0, YA0, XB0, YB0 ? '
READ (5, *) PAR(1), PAR(2), PAR(3), PAR(4)
PRINT *, ' K ? '
READ (5, *) PAR(5)
C      Шаг численного дифференцирования
PRINT *, ' DDX ? '
READ (5, *) DDX
C
C      Вызов модели элемента при NSTEP = 0, NEWINT=1
CALL LINKD ( I, Y, X1, X2, X3, X4, PAR(1), WRK)
IF (CODE .GT. 0) , THEN
PRINT *, ' Ошибка в параметрах, CODE=',CODE CODE = 0

```

```

C          GO TO 20 END IF
C
C          Устанавливаем "первый шаг интегрирования". NEWINT = 0.
NSTEP = 1
NITER = 1
NEWINT = 0 C
50      CONTINUE
C          Цикл проверки якобиана при разных значениях
C          перемещений
PRINT *, 'Еще точка ? (1 - да)'
READ (5,*) NUM
IF (NUM .NE. 1) STOP
C          Ввод значений перемещений
PRINT *, 'Значения перемещений X1 ... X4 ?'
READ (5,*) X1(1),X2(1),X3(1),X4(1)

```

### III - 27

```

C          Вызов модели при заданных значениях перемещений
CALL LINKD (I, Y, X1, X2, X3, X4, PAR(1), WRK) C
C          Распечатка вектора усилий.
WRITE (6,799) (I(LLI),LLI=1,4)
WRITE (SYSPRN,799) (I(LLI),LLI=1,4)
C
C          CONTINUE
C          Численная проверка якобиана
PRINT *, 'введите номер узла'
READ (5,*) NUM
C          Если NUM > 4, перейдем к следующей точке
IF (NUM.GT.4) GO TO 50 C
C          Для аппроксимации частных производных используем
C          центральные разности.
IF (NUM.EQ.1) X1 (1) = X1 (1) - DDX
IF (NUM.EQ.2) X2 (1) = X2 (1) - DDX
IF (NUM.EQ.3) X3 (1) = X3 (1) - DDX
IF (NUM.EQ.4) X4 (1) = X4 (1) - DDX
CALL LINKD (I,YTMP,X1,X2,X3,X4,PAR(1),WRK) C
IF (NUM.EQ.1) X1 (1) = X1 (1) + DDX*2
IF (NUM.EQ.2) X2 (1) = X2 (1) + DDX*2
IF (NUM.EQ.3) X3 (1) = X3 (1) + DDX*2
IF (NUM.EQ.4) X4 (1) = X4 (1) + DDX*2
CALL LINKD (I1,YTMP,X1,X2,X3,X4,PAR(1),WRK) C
WRITE (6,800) NUM
C
C          Численное определение элементов якобиана,
C          распечатка значений элементов якобиана,
C          полученных моделью, и численных значений
C          якобиана.
DO 200 LLI=1,4
Y1      = (I1(LLI)-I(LLI))/DDX/2
Y2      = Y (4*(LLI-1)+NUM,1)
C
C          Если Y1 достаточно велико, печатаем
C          относительную разность, в противном случае -
C          абсолютную.

```

```

                IF (ABS(Y1) .GT. SQRT (RLMIN)) ,           THEN
                    WRITE (6,900) LLL, Y1, Y2,
                        (Y1-Y2)/Y1 * 100.
                ,
                ELSE
                    WRITE (6,900) LLL, Y1, Y2, ,
                (Y1-Y2)
                END IF 200 CONTINUE
C
                IF (NUM.EQ.1) X1 (1) = X1 (1) - DDX IF (NUM.EQ.2) X2 (1) =
                X2 (1) - DDX IF (NUM.EQ.3) X3 (1) = X3 (1) - DDX IF
                (NUM.EQ.4) X4 (1) = X4 (1) - DDX
C
                GO TO 100

```

### III - 28

```

C
799      FORMAT (2X, 'Усилия в заданной точке:',
                ,      /,2X, ' FAX=',D12.5,
                ,      /,2X, ' FAY=',D12.5,
                ,      /,2X, ' FBX=',D12.5,
                ,      /,2X, ' FBY=',D12.5)
800      FORMAT (/1X, 'Производные сил по ',I2, ' переменной:',
                ,      //10X, 'численные          аналитические          разница  '/')
900      FORMAT (1X,I3, 3D15.5)
C
      END

```

Автономное тестирование, проводимое с использованием такого рода программ, должно быть возможно более полным. Если не обязательно, то очень желательно осуществлять :

1) проверку поведения модели в особых ситуациях (при соответствующем задании параметров модели или определении перемещений (скоростей, ускорений);

2) а н а л и т и ч е с к у ю проверку рассчитываемых усилий при заданном сочетании исходных параметров и значениях перемещений (скоростей, ускорений);

3) если алгоритм модели содержит ветвления, то весь комплекс проверок таким образом, чтобы осуществить проверку всех частей программы хотя бы один раз.

Следует признать весьма маловероятным исход, при котором в ходе автономного тестирования не будет найдено ни одной ошибки (Такое бывает только для самых простых моделей). ИЩИТЕ, ДА ОБРЯЩЕТЕ.

И еще. Ошибки, найденные при автономном тестировании модели элемента, обходятся значительно дешевле, чем ошибки, которые проявились в модели элемента, когда она вошла в состав модели сложной технической системы. Тогда найти и обезвредить их будет значительно сложнее.

#### 3.6.9. Текст модели технической системы на языке PRADIS с использованием элемента LINKD.

Итак, автономное тестирование модели элемента завершено успешно. Теперь включим ее в состав комплекса командой

> ARM + LINKD

Если все сделано правильно, то по этой команде будет получен объектный модуль модели элемента. Он будет включен в библиотеку CURRENT.LIB. Справочная информация по модели элемента будет помещена в системный каталог. С этого момента

### III - 29

она становится доступной для справки по модели элемента в режиме ON LINE (это нужно ОБЯЗАТЕЛЬНО ПРОВЕРИТЬ !):

- |               |   |
|---------------|---|
| > ARM ?       | - в общую справку по библиотекам комплекса должна попасть информация о назначении модуля LINKD; |
| > ARM ? LINKD | - полная информация о модели элемента LINKD.  |

Теперь модель элемента можно тестировать в составе комплекса. Нам кажется, что неплохим подходом будет ВСЕГДА воспринимать первые запуски разработанного модуля как ТЕСТОВЫЕ и относиться к ним так же придирчиво, как и к автономному тестированию модуля. Еще лучше - первые тесты для разработанного элемента делать на специально разработанных моделях технических систем. Например, для нашего случая - это:

- статическая деформация пружины силой, направленной вдоль оси;
- тестирование в режиме пружинного маятника при различных начальных угловых положениях пружины;
- тестирование в режиме математического маятника.

Каждый тест, по возможности, должен сопровождаться АНАЛИТИЧЕСКОЙ проверкой результатов. Вообще, качество и действенность теста можно оценить по возможности его аналитической проверки.

Окончательной стадией разработки модели элемента будет стадия тестирования в составе модели требуемой технической системы (в данном случае - рис. 3.3). Ниже приводится текст этой модели на языке PRADIS:

```
$ DATA :  
Центр тяжести тела 1 = 1, 1  
Центр тяжести тела 2 = 2, 2  
Масса и момент инерции тела 1 = 1, 0.1  
Масса и момент инерции тела 1 = 2, 0.1  
Жесткость связи = 5 E3  
Сила = 1.E3  
$ FRAGMENT :  
# BASE : 7  
# STRUCTURE :
```



На рис. 3.6.-3.8. приводятся схемы расположения элементов якобиана для всех трех возможных способов его описания.

Текст модели точечного инерционного элемента со смещенным центром тяжести, совершающего плоское движение, приведен ниже.

### III - 31

```

C
C MODEL MEKS: EXT=3, PAR=3, ADR=1, IGN=0
C
C          Дата создания:                01/21/92 08:34am
C          Дата последней корректировки: 10/08/93 10:18am
C
C HELP Точечный инерционный          элемент с эксцентриситетом
C HELP НАЗВАНИЕ:                      Точечный инерционный элемент с
C HELP                               эксцентриситетом.
C
C HELP ОБЛАСТЬ ПРИМЕНЕНИЯ : Механика.
C HELP СТЕПЕНИ СВОБОДЫ:
C HELP 1 - поступательная по оси X;
C HELP 2 - поступательная по оси Y;
C HELP 3 - вращательная.
C HELP ПАРАМЕТРЫ:
C HELP 1 - величина массы                (M >= 0);
C HELP 2 - величина эксцентриситета массы (R >= 0);
C HELP 3 - начальный угол между осью OX и радиус-вектором,
C HELP          соединяющим центр вращения с центром массы(град).
C
      SUBROUTINE MEKS ( I, Y, X1, X2, X3, PAR) C константы этой
программы:
      CHARACTER*8 NAMEMD
      PARAMETER ( NAMEMD = 'MEKS' )
C
C      Передаваемые параметры:
      REAL      * 8 I(1),          Y(9,1)
      REAL      * 8 X1(1), X2(1), X3(1),
      ,          PAR(1)
C      ,      Переменные и параметры программы
      REAL      * 8 M,          R,          ALFA, J,
      ,          FI,          EPS,          W,          AX,          AY,
      ,          COS FI, SIN FI
C
C      переменные непоименованной COMMON-области
      REAL      * 8 TIME, STEP, STEP01, STEP02, SMIN,
      ,          VAL001, VAL002, STEPMD, TIMEND
      CHARACTER * 8 NAME
      INTEGER * 4 NSTEP, SYSPRN, NITER, ITR
      INTEGER * 2 CODE, NUMINT, NUMPRV, CODSTP, CODGRF, , NEWINT, MINSTP
C      ,      переменные поименованной COMMON-области /NOTAT/
      REAL      * 8 RLMAX, RLMIN, INTMAX, MSHEPS,
      ,          PI,          REZA, REZB, REZC
C
C      непоименованная COMMON-область

```

```

COMMON      TIME, STEP, STEP01, STEP02, SMIN,
,            VAL001, VAL002, STEPMD, TIMEND, NAME,
,            NSTEP, SYSPRN, NITER, ITR,          CODE,
,            NUMINT, NUMPRV, CODSTP, CODGRF,
,            NEWINT, MINSTP
C      поименованная COMMON-область
COMMON /NOTAT/
,            RLMAX, RLMIN, INTMAX, MSHEPS,
,            PI,      REZA, REZB, REZC
C

```

### III - 32

```

C      Инициализация параметров программы :
M      = PAR (1)
R      = PAR (2)
ALFA = PAR (3) * PI/180
C      Однократные вычисления в программе :
      IF (NEWINT .EQ. 1), THEN
C      Проверка значений параметров на допустимость:
      IF ( M      .LT. 0.0      .AND.
,      R      .LT. 0.0      )
,      THEN
C
      IF ( CODE .LT. 100 ) ,      THEN
      CODE      = 100
      NAME      = NAMEMD
      END IF
C
C      Печать в SYSPRINT неверного вектора
C      параметров :
      IF (SYSPRN.GT.0) ,      THEN
      WRITE(SYSPRN,900)NAMEMD,M,R END IF
C ==> EXIT      Аварийный выход из модели :
      RETURN
      END IF
C      Конец действий по проверке параметров на
C      допустимость.
      END IF
C      Конец однократных вычислений в программе.
C
C      Некоторые вспомогательные величины
AX      = X1 (3)
AY      = X2 (3)
FI      = X3 (1) + ALFA
W      = X3 (2)
EPS      = X3 (3)
J      = M * R * R
COS FI = DCOS (FI)
SIN FI = DSIN (FI) C
C      Величины сил и моментов :
I (1)      = (- W*W*R*COS FI - EPS*R*SIN FI + AX) * M
I (2)      = (- W*W*R*SIN FI + EPS*R*COS FI + AY) * M
I (3)      = EPS * J
C

```

```

C      Матрица проводимостей элемента
      Y (1, 1) = 0.
      Y (2, 1) = 0.
      Y (3, 1) = M * ( W*W*R * SIN FI - EPS*R * COS FI)
      Y (4, 1) = 0.
      Y (5, 1) = 0.
      Y (6, 1) = M * (-W*W*R * COS FI - EPS*R * SIN FI)
      Y (7, 1) = 0.
      Y (8, 1) = 0.
      Y (9, 1) = 0.

```

### III - 33

```

C
      Y (1, 2) = 0.
      Y (2, 2) = 0.
      Y (3, 2) = - 2.D0 * W * R * M * COS FI
      Y (4, 2) = 0.
      Y (5, 2) = 0.
      Y (6, 2) = - 2.D0 * W * R * M * SIN FI
      Y (7, 2) = 0.
      Y (8, 2) = 0.
      Y (9, 2) = 0.

```

```

C
      Y (1, 3) = M
      Y (2, 3) = 0.
      Y (3, 3) = - R * M * SIN FI
      Y (4, 3) = 0.
      Y (5, 3) = M
      Y (6, 3) = R * M * COS FI
      Y (7, 3) = 0.
      Y (8, 3) = 0.
      Y (9, 3) = J

```

```

C
      RETURN
900   FORMAT (/T4,A8,',',
,      T15,'Ошибка в задании параметров:'
,      /T15,'масса          =' ,G11.5,
,      /T15,'эксцентриситет =' ,G11.5)
      END

```



## 4. ВКЛЮЧЕНИЕ ПРОГРАММ РАСЧЕТА ВЫХОДНЫХ ПЕРЕМЕННЫХ В БИБЛИОТЕКИ КОМПЛЕКСА

### 4.1. Функции программ расчета выходных переменных и структура оператора SUBROUTINE.

Существуют две причины, по которым переменные, непосредственно получаемые в результате решения исходной системы уравнений в комплексе PRADIS, недоступны для отображения:

1) Обычно количество переменных, требующихся для анализа результатов расчета, значительно меньше общего количества результатов, получаемых на каждом шаге интегрирования.

2) Часто для получения требующихся результатов внутренние переменные нужно подвергнуть некоторым преобразованиям.

Рис. 4.1. показывает место программ расчета выходных переменных (ПРВП) в этом процессе. По завершении каждого шага интегрирования происходит обращение к требуемым ПРВП, которые и вычисляют текущее состояние вектора выходных переменных. Исходными данными для этого являются :

- значения внутренних переменных на данном шаге интегрирования. Ко внутренним переменным в комплексе PRADIS относятся значения всех потенциальных переменных данного шага, величины всех потоковых переменных для каждой модели элемента, а также переменные рабочего вектора всех моделей элементов;
- значения постоянных параметров, заданных пользователем в описании вызова программы расчета выходной переменной (подраздел #OUTPUT соответствующего раздела \$FRAGMENT).

Каждая программа расчета выходной переменной может рассчитывать одну или несколько выходных переменных. При этом предполагается, что несколько выходных переменных, рассчитанных одной ПРВП, - это одна многокомпонентная выходная переменная. Доступ к каждой компоненте этой переменной осуществляется по ее порядковому номеру.

Например, такая запись вызова программы отображения DISP:

Вывод результатов 'DISP (FROM=1;  
Переменная (2), Переменная (1))

означает, что требуется построить график зависимости первой компоненты многокомпонентной выходной переменной "Переменная" от второй компоненты этой же переменной.

Значения всех выходных переменных сохраняются в файле результатов интегрирования с заданным для данной программы интегрирования шагом OUT.

#### IV - 1

Оператор SUBROUTINE для программы расчета выходной переменной имеет две формы записи в зависимости от значения ключевого параметра VPS в паспорте ПРВП.

Если ключевой параметр VPS говорит о том, что данная ПРВП ожидает на входе фиксированное количество внутренних переменных (VPS=0), то оператор SUBROUTINE в этом случае выглядит так :

```
SUBROUTINE OUTPUT (OUT, SYS1, ... SYSN, PAR, WRK)
REAL * 8 OUT(1), SYS1, ... SYSN, PAR(1), WRK(1)
```

В этом вызове :

OUTPUT - имя программы расчета выходной переменной, сформированное по правилам языка FORTRAN.

OUT - рассчитываемая выходная переменная или вектор рассчитываемых выходных переменных. Количество рассчитываемых данной ПРВП выходных переменных определяется паспортом ПРВП. Нужно отметить, что в PRADIS не гарантируется сохранение значений элементов массива OUT при переходе на следующий шаг интегрирования. Поэтому для сохранения требуемых величин этого массива (например, если нужно накапливать какую-то величину), используется рабочий вектор ПРВП. Если параметр OUT равен нулю, этот вектор отсутствует в списке формальных параметров программы;

SYS1...SYSN - значения внутренних переменных на данном шаге интегрирования. Необходимо отметить, что если J-ая внутренняя переменная в этом списке имеет быть потенциальной, и в тексте на языке PRADIS она описана как указатель на номер узла, то внутри ПРВП она ВСЕГДА описывается как массив из трех переменных. При этом первый элемент массива - перемещение по заданной степени свободы, второй - скорость, третий - ускорение. Например :

```
SUBROUTINE V (OUT, X, PAR)
```

```

REAL * 8 OUT, X(1), PAR (1) C PAR(1)-
заданный масштаб вывода C Расчет текущей скорости
OUT = PAR(1) * X (2)
RETURN
END

```

Если используется указатель на номер узла с символом ' ', то в ПРВП передается массив из двух переменных, первым из которых является скорость, вторым - ускорение. Если используется указатель на

#### IV - 2

номер узла с символом ", то в ПРВП передается ускорение соответствующего узла. Количество передаваемых в ПРВП внутренних переменных определяется паспортом ПРВП. Как следует из описания языка, для передачи в ПРВП значения силы, действующей на определенную ветвь элемента, используется указатель I; для передачи потенциальной переменной используется номер узла, которому эта переменная соответствует, для передачи элемента рабочего вектора - указатель W;. Если значение ключевого параметра SYS равно 0, внутренние переменные отсутствуют в списке формальных параметров программы.

**ВНИМАНИЕ ! ПРВП ни в коем случае не должна менять значения внутренних переменных.**

#### PAR

- вектор параметров для программы расчета выходной переменной. Количество параметров ПРВП определяется паспортом программы. ПРВП не может менять значения элементов этого вектора. Если этот массив в программе, реализующей ПРВП, описан обычным образом (например, PAR(1)), то ячейка памяти PAR (0) содержит количество параметров для этой ПРВП. Это может понадобиться в случае реализации ПРВП с переменным количеством параметров. В таком случае ячейка памяти PAR (0) содержит фактическое количество параметров, передаваемых в ПРВП при данном вызове. Сочетание нулевых значений ключевых параметров VPR и PAR приводит к тому, что этот вектор отсутствует в списке формальных параметров программы.

#### WRK

- рабочий вектор ПРВП. Используется в тех же целях, что и вектор WRK модели элемента. Правда есть одна тонкость. Обращение в ПРВП происходит только один



раз на шаге интегрирования, в случае его успешного завершения. Поэтому ПРВП не нуждается в векторе состояния. Скажем, накопление какой-либо величины можно делать с использованием следующей группы операторов :

```

.....
IF (NSTEP .EQ. 0) , THEN
    WRK (1) = 0.D0
END IF .....
WRK (1) = WRK (1) + STEP*X1(1)
.....

```

#### IV - 3

Рабочий вектор ПРВП имеет постоянную часть и переменную часть, зависящую от количества параметров в векторе параметров ПРВП. Общая длина доступного в ПРВП рабочего вектора определяется сочетанием ключевых параметров паспорта WRK и WRP. Если длина рабочего вектора, определяемая этими параметрами, равна 0, вектор отсутствует в списке формальных параметров программы.

Вторая разновидность оператора SUBROUTINE для ПРВП выглядит следующим образом:

```

SUBROUTINE OUTPUT (OUT, SYS, NVAR, PAR, WRK)
REAL * 8 OUT (1), SYS (1), PAR (1), WRK (1) INTEGER * 4 NVAR

```

Эта запись заголовка ПРВП соответствует программе расчета выходных переменных с переменным количеством передаваемых ей указателей на внутренние переменные. При этом SYS- это вектор передаваемых в ПРВП внутренних переменных, NVAR - количество элементов в этом векторе. Предположим, что в базисной библиотеке комплекса имеется программа расчета суммы нескольких внутренних переменных :

```

SUBROUTINE SUM (OUT, SYS, NVAR, PAR) C
REAL * 8 OUT, SYS (1), PAR (1) INTEGER * 4 NVAR, I
C
C      PAR(1) - масштаб
OUT = 0.D0
DO 100 I= 1, NVAR
    OUT = OUT + SYS (I) 100
CONTINUE
OUT = OUT * PAR(1)
C
RETURN
END

```

Предположим также, что в тексте описания объекта присутствует следующий подраздел описания вывода :

```
# OUTPUT
Сумма сил          'SUM (I:Элемент(1),
                   I:Элемент(2),I:Привод(3);1)
Сумма перемещений 'SUM (1,2,33,56,78;1)
```

Тогда выходная переменная "Сумма сил" будет равна алгебраической сумме сил, действующих по первой и второй ветвям элемента "Элемент" и по третьей ветви элемента "Привод". Выходная переменная "Сумма перемещений" будет равна алгебраической сумме перемещений по степеням свободы 1, 2, 33, 56 и 78. Необходимо отметить, что при второй форме записи оператора SUBROUTINE для ПРВП (и при соответствующем этому случаю определении ее паспорта) для манипулирования скоростями

#### IV - 4

(ускорениями) необходимо использовать указатели на номер узла с соответствующими символами ( ' или "). Например, алгебраическое суммирование скоростей может быть выполнено таким вызовом программы SUM:

```
Сумма скоростей 'SUM (1',2',33',56',78';1)
```

#### 4.2. Паспорт программы расчета выходных переменных.

Паспорт ПРВП начинается ключевым словом OUTPUT. После этого идет имя подпрограммы, реализующей ПРВП и, после символа ":" - имена ключевых параметров паспорта и их значения - положительные целые числа.

В паспорте ПРВП пользователя, включаемой в состав библиотек комплекса, могут быть заданы следующие ключевые параметры :

- |       |  |
|-------|--|
| OUT - | определяет количество компонент в рассчитываемой данной программой выходной переменной (длину вектора выходных переменных, рассчитываемого данной ПРВП). По умолчанию OUT = 1; |
| SYS - | определяет количество внутренних переменных, передаваемых ПРВП. В ПРВП, предназначенных для обработки переменного количества внутренних переменных, этот                       |

ключевой параметр определяет минимально возможное количество внутренних переменных, передаваемых данной ПРВП. По умолчанию  $SYS = 1$ . Если  $SYS=0$  и  $VPS=0$ , внутренние переменные отсутствуют в списке формальных параметров ПРВП;

VPS - признак переменного количества внутренних переменных. Если VPS не задан или равен 0, считается, что ПРВП предназначена для обработки постоянного количества внутренних переменных (это количество определено ключевым параметром SYS). Если  $VPS=1$ , то ПРВП обрабатывает переменное количество внутренних переменных, но не меньше, чем это определено ключевым параметром SYS. Этот случай соответствует второму типу оператора SUBROUTINE для ПРВП. Разработчик может потребовать от транслятора дополнительного контроля количества внутренних переменных на четность и нечетность. Если количество внутренних переменных должно быть нечетным, то ключевой параметр VPS задается равным 11, если четным - то 21;

#### IV - 5

PAR - определяет количество параметров для ПРВП с постоянным количеством параметров. Для ПРВП с переменным количеством параметров задает минимально возможное количество параметров для этой ПРВП. Если ключевой параметр PAR не задан, количество параметров ПРВП принимается равным 1. Если  $PAR=0$  и  $VPR=0$ , вектор параметров отсутствует в списке формальных параметров ПРВП;

VPR - признак того, что ПРВП имеет переменное количество параметров. Если VPR не задан или равен 0, считается, что ПРВП имеет постоянное количество параметров. Если  $VPR=1$ , то ПРВП имеет переменное количество параметров. При этом на этапе синтаксического анализа контролируется, чтобы заданное количество параметров ПРВП было не меньше ключевого параметра PAR. Разработчик может потребовать от транслятора дополнительного контроля количества параметров на четность и нечетность. Если количество параметров ПРВП должно быть нечетным, то ключевой параметр VPR задается равным 11, если четным - то 21;

WRK	- определяет длину постоянной части рабочего вектора ПРВП;
WRS	- определяет, сколько элементов рабочего вектора должно соответствовать каждой внутренней переменной в переменной части вектора внутренних переменных, передаваемых в ПРВП. Этот ключевой параметр полезен в ПРВП с переменным количеством передаваемых внутренних переменных, когда требуется для каждой переменной или группы переменных сохранять результаты каких-либо однократных вычислений;
WRP	- определяет, сколько элементов рабочего вектора должно соответствовать каждому параметру в переменной части вектора параметров. Этот ключевой параметр полезен в ПРВП с переменным количеством параметров, когда требуется для каждого параметра или группы параметров сохранять результаты каких-либо однократных вычислений. ПРИМЕЧАНИЕ. Общая длина рабочего вектора, передаваемого модели элемента, вычисляется по формуле $WRK + (<\text{фактическое количество вн. переменных} - \text{SYS}) * WRS + (<\text{фактическое количество параметров} - \text{PAR}) * WRP$

#### IV - 6

#### 4.3. Справочная информация по ПРВП.

Рекомендуется следующее содержимое справочной информации по ПРВП:

- 1) первая строка справочной информации, содержащая краткое наименование программы расчета выходной переменной;
- 2) полное наименование ПРВП;
- 3) тип передаваемых в ПРВП указателей на внутренние переменные;
- 4) описание вектора параметров ПРВП;
- 5) описание выходной переменной или описание каждой из компонент многокомпонентной выходной переменной.

#### 4.4. Примеры программ расчета выходных переменных комплекса PRADIS.

##### 4.4.1. ПРОГРАММА X. Расчет перемещения, силы и элемента рабочего вектора.

Примеры вызова программы X в подразделе #OUTPUT описания объекта:

#OUTPUT :

Перемещение по оси X точки A'	X (12; 1)
Скорость по оси Y точки A '	X (13'; 1)
Ускорение по оси Y точки A '	X (13"; 1)
Сила сжатия пружины '	X (I: Пружина(1); 1)
Упругая энергия пружины '	X (W: Пружина(12); 1)

Текст программы :

```
C OUTPUT X: OUT=1, SYS=1, PAR=1
C          Дата создания: 08/22/90 10:21am
C          Дата последней корректировки: 01/24/92 05:24pm
C
C HELP Расчет заданной выходной переменной.
C HELP НАЗВАНИЕ: Программа расчета выходной переменной
C HELP (перемещения, силы или компоненты
C HELP рабочего вектора модели элемента).
C HELP ТИП ПЕРЕДАВАЕМЫХ В ПРОГРАММУ ВНУТРЕННИХ ПЕРЕМЕННЫХ:
C HELP 1 - перемещение, сила, или компонента рабочего
C HELP вектора модели элемента.
C HELP ПОСТОЯННЫЕ ПАРАМЕТРЫ:
C HELP 1 - масштаб.
```

#### IV - 7

```
C HELP ВЫХОДНЫЕ ПЕРЕМЕННЫЕ:
C HELP 1 - переданная внутренняя переменная, умноженная на
C HELP масштаб.
SUBROUTINE X ( XOUT, XSYS, PAR )
REAL * 8 XOUT, XSYS, PAR
XOUT = PAR*XSYS
RETURN
END
```

#### 4.4.2. ПРОГРАММА MAXI. Расчет максимального значения из N внутренних переменных.

Программа MAXI приводится здесь как пример ПРВП с переменным количеством передаваемых в программу внутренних переменных, рассчитывающей многокомпонентную выходную переменную.

Текст программы :

```
C OUTPUT MAXI:SYS=0,VPS=1,OUT=2,PAR=1
C
C          Дата создания:                07/21/92 11:47am
C          Дата последней корректировки: 05/31/93 12:09pm
C HELP Расчет максимального значения из N внутренних переменных C HELP
НАЗВАНИЕ: Программа расчета максимального значения C HELP из N внутренних
переменных.
C
C HELP ТИП ПЕРЕДАВАЕМЫХ В ПРОГРАММУ      ВНУТРЕННИХ ПЕРЕМЕННЫХ:
C HELP 1 ... N - перемещение, сила,      компонента рабочего вектора
C HELP или компонента вектора            состояния модели элемента;
C
C HELP ПОСТОЯННЫЕ ПАРАМЕТРЫ:
C HELP 1 - масштаб.
C
C HELP ВЫХОДНЫЕ ПЕРЕМЕННЫЕ:
C HELP 1 - текущее максимальное значение переданных
C HELP внутренних переменных, умноженное на масштаб;
C HELP 2 - порядковый номер внутренней переменной, текущее
C HELP значение которой максимально.
C
SUBROUTINE MAXI ( XOUT, XSYS, NVAR, PAR ) C
  INTEGER * 4  NVAR
  REAL  * 8  XOUT(1), PAR
  REAL * 8 XSYS(1) C
  INTEGER * 4 I C
  XOUT(1) = XSYS(1)
  XOUT(2) = 1
  DO 10 I = 2,NVAR
    IF ( XSYS(I) .GT. XOUT(1) ) ,      THEN
      IV - 8
      XOUT(1) = XSYS(I)
      XOUT(2) = I
    END IF 10  CONTINUE
```

```
XOUT(1) = PAR * XOUT(1)  
RETURN  
END
```





## ВКЛЮЧЕНИЕ ПРОГРАММ РЕАЛИЗАЦИИ ГРАФИЧЕСКИХ ОБРАЗОВ В БИБЛИОТЕКИ КОМПЛЕКСА

### 5.1. Функции программ реализации графических образов и структура оператора SUBROUTINE.

После завершения очередного шага интегрирования, в случае, если в тексте программы на языке PRADIS присутствовало описание изображения объекта, происходит обращение к программам реализации графических образов для воспроизведения изображения текущего состояния объекта на экране.

В комплексе PRADIS при описании изображения объекта существует несколько правил построения изображения, которые предопределяют существование нескольких разновидностей программ реализации графических образов. ПЕРВЫЙ СПОСОБ описания изображения объекта - это использование графических образов, связанных с моделями элементов "по умолчанию". Некоторые из включенных в состав комплекса моделей элементов (а именно, модели механических элементов, совершающих плоское и пространственное движение) связаны со своими образами "по умолчанию". В этом случае при описании изображения объекта можно не указывать, какой графический образ соответствует тому или иному элементу. ВТОРОЙ СПОСОБ описания изображения объекта

- это явное указание, какой графический образ (возможно, нестандартный) соответствует тому или иному элементу.

Примеры соответствующих описаний изображений объекта :

\$ SHOW :

изображение всех элементов объекта, имеющих графические образы "по умолчанию"

Все изображение ' LAYER (;Параметры слоя 1)

изображение элементов с идентификаторами "Шатун" и "Кривошип" с использованием графических образов по умолчанию

Изображение механизма' LAYER (Шатун, Кривошип;  
Параметры слоя 2)

изображение элементов с идентификаторами "Шатун" и "Кривошип" с использованием нестандартных графических образов

Другой вариант изображения' LAYER (  
Шатун (SHTN;Параметры изображения шатуна),  
Кривошип (KRVSH; Параметры изображения кривошипа);  
Параметры слоя 3  
)

Еще одна возможность построения изображения, - это изображение графических образов, не связанных с элементами изображения объекта, а связанных с неподвижной системой координат (это графические образы изображения окружения объекта). Способ включения таких программ в изображение объекта

V - 1

похож на второй способ изображения элементов объекта, только в этом случае не указывается идентификатор элемента, которому соответствует данный графический образ :

Изображение окружения механизма 'LAYER (  
(ROOM;Параметры изображения);  
Параметры слоя 2)

В соответствии с этим различают следующие разновидности программ реализации графических образов :

- 1) графический образ, связанный с моделью элемента "по умолчанию";
- 2) нестандартный графический образ;
- 3) графический образ, связанный с неподвижной системой координат (неподвижный графический образ).

Оператор SUBROUTINE для графических образов первого и второго типа выглядит следующим образом :

```
SUBROUTINE IMAGE (I,  
, X1, .... X3,  
, PAR, NEW, OLD, WRK,  
, PARIMD, WRKIMD,  
, PARLR2)  
REAL * 8 I(1), X1(1), ... X3(1)  
REAL * 8 PAR(1), NEW(1), OLD(1), WRK(1)  
REAL * 8 PARIMD(1), WRKIMD(1), PARLR2(1)
```

В этом вызове :

I - вектор сил модели элемента, с которой связана программа реализации графического образа. Графический образ не должен изменять элементы этого вектора;

X1 ... XN - массивы потенциальных переменных для узлов модели элемента, с которой связан графический образ. Необходимо отметить, что порядок следования этих переменных соответствует порядку перечисления узлов при описании топологии элемента. В отличие от модели элемента и независимо от параметра ADR, в графический образ всегда передается все три потенциальных переменных, соответствующих данному узлу (X1(1) - перемещение, X1(2) - скорость, X1(3) - ускорение). Программа реализации графического образа не должна изменять элементы вектора потенциальных переменных;

PAR, NEW, OLD, WRK - вектора параметров, состояния и рабочий вектор модели элемента, с которой связан графический образ.

V - 2

ВНИМАНИЕ ! 1) Графический образ не должен изменять элементы этих массивов. 2) Эти массивы присутствуют в вызове программы реализации

графического образа, даже если в операторе SUBROUTINE модели элемента соответствующие элементы отсутствуют.  
Это сделано для того, чтобы некоторые графические образы могли быть использованы для изображения различных элементов, похожих по существу, но отличающихся оператором SUBROUTINE.

- PARIMD - вектор параметров графического образа.  
Количество элементов в векторе параметров графического образа задается паспортом графического образа. Если задано PAR=0, PARIMD все равно присутствует в операторе SUBROUTINE программы реализации графического образа.
- WRKIMD - рабочий вектор графического образа.  
Аналогичен соответствующему массиву ПРВП. Длина рабочего вектора определяется паспортом графического образа. Если задано WRK=0, этот массив все равно присутствует в операторе SUBROUTINE программы реализации графического образа.
- PARLR2 - параметры текущего слоя изображения.

В массиве PARLR2 содержится 11 элементов :

- 1-3 - координаты центра экрана для текущего слоя изображения;
- 4-6 - направляющие косинусы оси OX системы координат текущего слоя в глобальной системе координат;
- 7-9 - направляющие косинусы оси OY системы координат текущего слоя в глобальной системе координат;
- 10 - пиксельный масштаб изображения;
- 11 - текущий цвет изображения.

ПРИМЕЧАНИЕ. Большинство элементов массива PARLR2 не нужны пользователю для непосредственного использования (они применяются в программе получения пиксельных координат изображения GLASS).

Важным для пользователя может оказаться элемент PARLR2 (10) - пиксельный масштаб изображения. Пиксельная длина изображения любого элемента длиной L на экране вычисляется как произведение  $L * PARLR2 (10)$  (по оси Y),  $L * PARLR2 (10) *$

#### V - 3

RELYX (по оси X; по поводу переменной RELYX см. содержимое COMMON блока GRCONF в Приложении 1). В случае, если нужно вычислить пиксельную длину какого-либо изображения вручную, нужно иметь в виду, что она зависит от текущих параметров слоя и в ходе расчета может изменяться (если применяется REPLACE для параметров программы реализации слоя и RESTORE - восстановление расчета с последнего места сохранения). Поэтому эта величина должна вычисляться при каждом входе в программу реализации графического образа.

Еще одним важным элементом - PARLR2 (11). Если пользователю понадобилось по каким-либо причинам использовать

цвет изображения, отличный от цвета изображения для данного слоя, то для возврата к первоначальному цвету изображения нужно использовать PARLR2(11) :

```
.....  
INTEGER * 4  NUMCOL  
.....  
C      Возврат к изображению цветом слоя по умолчанию  
      NUMCOL = INT (PARLR2(11) + 0.5)  
      CALL COLOR (NUMCOL)  
.....
```

Оператор SUBROUTINE для графических образов третьего типа отличается от этого оператора для образов первого и второго типов :

```
SUBROUTINE IMAGE ( PARIMD, WRKIMD, PARLR2)
```

Т.е., формальные параметры, относящиеся к модели элемента, в этом операторе отсутствуют. Для PARIMD, WRKIMD и PARLR2 действуют те же правила, что для графических образов первого и второго типов.

Из вышесказанного следует, например, что графический образ, предназначенный для изображения элемента, имеющего 4 степени свободы, не может быть использован для изображения элемента с другим количеством степеней свободы или быть связанным с программой изображения слоя. Если это не очевидно, лучше перечитать приведенную выше информацию еще раз.

## 5.2. Паспорт программы реализации графического образа и правила формирования имен ПГО, связанных с моделями элементов по умолчанию.

Паспорт ПГО (программы реализации графического образа) начинается ключевым словом IMAGE. После этого идет имя подпрограммы, реализующей ПГО и, после символа ":" - имена ключевых параметров паспорта и их значения - положительные целые числа.

### V - 4

В паспорте ПГО пользователя, включаемой в состав библиотек комплекса, могут быть заданы следующие ключевые параметры :

EXT	- количество степеней свободы модели элемента, с которой связан графический образ. Для ПГО третьего типа задают EXT=0;
PAR	- количество элементов в векторе параметров графического образа (PARIMD). Если задано PAR=0, считается, что этот графический образ будет связан с моделью элемента. В этом случае имя программы реализации графического образа задается по правилам, сформулированным в этом подразделе ниже;
WRK	- количество элементов в рабочем векторе ПГО (WRKIMD).

Правила для имен графических образов, связанных с моделями элементов по умолчанию :

1) Если имя модели элемента содержит 4 или 5 символов, то имя программы реализации графического образа для этого элемента является реверсированным именем модели элемента. Например:

Имя модели элемента	Имя графического образа по умолчанию
PLSTU	- UTSLP
STRGN	- NGRTS
KNTO	- OTNK

2) Если имя модели элемента содержит 6 символов, то перед реверсированием имени последний символ отбрасывается. Например:

Имя модели элемента	Имя графического образа по умолчанию
BALKA	- AKLAB
BALKAD	- AKLAB
BALKAN	- AKLAB

3) Если имя модели элемента содержит 3 символа и меньше, то перед реверсированием оно дополняется символами "G" до 6 символов. Например:

Имя модели элемента	Имя графического образа по умолчанию
K	- GGGGGK
MU	- GGGGUM

## V - 5

### 5.3. Основные принципы, положенные в основу разработки программ реализации графических образов.

1) Программы реализации графических образов, как и все программы комплекса PRADIS, должны удовлетворять основополагающему принципу МОБИЛЬНОСТИ. Это значит, что ПГО не должна использовать в своей работе операций или действий, зависящих от типа используемой ЭВМ или типа используемых периферийных устройств. ВСЕ ИНСТРУКЦИИ ПГО должны быть написаны на стандарте языка FORTRAN. Единственным допускаемым отклонением от стандарта является явное описание типов параметров (INTEGER \* 4, REAL \* 8 и т.д.), и использование текстовой переменной NAME в непоименованном общем блоке.

Предполагается, что на каждой вычислительной установке, где функционирует графическое обеспечение PRADIS, доступными средствами будет реализован так называемый "нижний уровень графики PRADIS", который может быть непосредственно использован программами реализации графических образов.

В случае, если ПГО будет пренебрегать этими рекомендациями, ее использование на вычислительной технике другого типа может быть в значительной степени затруднено.

Описание нижнего графического уровня PRADIS приводится в Приложении 1 настоящего руководства.

2) Как правило, для получения пиксельных координат изображения должна использоваться "программа проецирования" GLASS. Эта программа при всех преобразованиях учитывает масштаб изображения и все изменения положения точки зрения наблюдателя, задаваемые средствами входного языка комплекса. Например :

```
REAL * 8 XC, YC
INTEGER * 4 PXA, PYA
REAL * 8 PARLR2(1)
.....
C      определим центр окружности в глобальной
C      системе координат :
XC = 25.6
YC = 35.7
C      получили координаты центра в метрах
C      получим пиксельные координаты центра
C      окружности
CALL GLASS (XC, YC, Z,
, PXA, PYA,
, PARLR2)
C      PXA, PYA - пиксельные координаты центра
C      окружности.
```

3) Пользователь не заботится о возможности выхода части изображения объекта за пределы экрана (изображение обрезается средствами комплекса PRADIS).

#### V - 6

ЗАМЕЧАНИЕ. При очень большом увеличении изображения можно получить программное прерывание по некорректности целочисленной операции, делению на ноль и т.д., связанное с ограниченностью представления целых чисел в базовом пакете FORTRAN, выбранном для реализации нижнего уровня графики PRADIS на IBM PC. В случае возникновения такой ошибки выполнение задания на расчет с ключевым параметром программы интегрирования MODE=0 приведет к исчезновению как изображения объекта на экране дисплея, так и ошибки как таковой (в это случае PRADIS не обращается к программам реализации графических образов).

Нужно помнить также, что графическое изображение объекта в ходе расчета является дорогой операцией. Счет идет значительно быстрее в режиме отображения графиков и еще быстрее вообще без графики (MODE=0).

#### 5.4. Справочная информация по ПГО.

В настоящее время содержание справочной информации по ПГО не еще не устоялось.

Рекомендуется :

- 1) Не давать справочной информации для ПГО, связанных с моделями элементов по умолчанию;
- 2) Справочную информацию для других ПГО по возможности согласовывать со справочной информацией для ПГО, уже включенных в комплекс.

## 5.5. Примеры программ реализации графических образов.

В этом подразделе разделе в качестве примера приводится программа реализации нестандартного графического образа.

```

C IMAGE  ARROW:EXT=3, PAR=4, WRK=0
C
C
C          Дата создания              09/03/93 03:09am
C          Дата последней корректировки 09/03/93 05:40am
C
C HELP Графический образ стрелки.
C HELP НАЗВАНИЕ: Графический образ стрелки, движущейся
C HELP          в плоскости.
C
C HELP СТЕПЕНИ СВОБОДЫ:
C HELP 1 - поступательная в направлении оси OX основания
C HELP          стрелки;
C HELP 2 - поступательная в направлении оси OY основания
C HELP          стрелки;
C HELP 3 - вращательная (не влияет на изображение).
C

```

V - 7

```

C HELP ПАРАМЕТРЫ:
C HELP      1 - начальная абсцисса основания стрелки;
C HELP      2 - начальная ордината основания стрелки;
C HELP      3 - длина стрелки;
C HELP      4 - угол между направлением стрелки и осью OX
C HELP           $(-0.25 \text{ PI} < \text{ALFA} < 1.75 \text{ PI}, \text{дискретность } 0.5 \text{ PI})$ 
C
C      SUBROUTINE  ARROW (I,
C          ,          X1,      X2,      X3,
C          ,          PAR, NEW, OLD, WRK,
C          ,          PARIMD, WRKIMD,
C          ,          PARLR2)
C
C      Формальные параметры
C      REAL * 8      I (1)
C      REAL * 8      X1,          X2,          X3
C      REAL * 8      PAR(1),      NEW(1),      OLD(1),
C          ,          WRK(1),
C          ,          PARIMD(1),      WRKIMD(1),
C          ,          PARLR2(1)
C
C      Параметры образа и локальные переменные :
C      REAL * 8      X,          Y,
C          ,          L,          ALFA
C      REAL * 8      COSAL,      SINAL
C      REAL * 8      XA,          YA,          Z,
C          ,          XB,          YB
C      REAL * 8      KONTR1 (4,1), KONTR2 (4,2)
C      INTEGER * 4    PXA,          PYA

```

```

      INTEGER * 4      PXB,          PYB
      LOGICAL * 4      FLOOD

C
C      COMMON-область NOTAT
      REAL      * 8  RLMAX, RLMIN, INTMAX, MSHEPS,
      ,          PI,          REZB,          REZC,          REZD
      COMMON /NOTAT/
      ,          RLMAX, RLMIN, INTMAX, MSHEPS,
      ,          PI,          REZB,          REZC,          REZD
C
      DATA KONTR1 / 0.0 D0, -0.1 D0, 0.6 D0, 0.1 D0/, , KONTR2 / 0.6 D0, -0.2 D0, 1.0 D0, 0.0 D0,
      ,          1.00 D0, 0.0 D0, 0.6 D0, 0.2 D0/
C
      DATA      Z          / 0.D0/
      DATA      FLOOD  /.TRUE./
C
C      Параметры образа :
      X          = PARIMD (1)
      Y          = PARIMD (2)
      L          = PARIMD (3)
      ALFA       = PARIMD (4)
C
C      Направляющие косинусы
      IF (ALFA          .GT. -0.25 D0*PI .AND.
      ,          ALFA          .LT. 0.25D0*PI)
      ,          THEN

      V - 8

      COSAL  = L
      SINL = 0 ELSE
      IF (ALFA          .GT. 0.25D0*PI .AND.
      ,          ALFA          .LT. 0.75D0*PI)
      ,          THEN
      COSAL  = 0
      SINL = L ELSE
      IF (ALFA          .GT. 0.75D0*PI .AND.
      ,          ALFA          .LT. 1.25D0*PI)
      ,          THEN
      COSAL  = -L
      SINL = 0 ELSE
      COSAL = 0 SINL = -L
      END IF END IF
      END IF C
C
C      Рисуем прямоугольник :

      XA          = X1 + X + KONTR1(1,1) * COSAL -
      -          KONTR1(2,1) * SINL
      YA          = X2 + Y + KONTR1(1,1) * SINL +
      +          KONTR1(2,1) * COSAL
      CALL GLASS (XA,          YA,          Z,
      ,          PXA, PYA,
      ,          PARLR2)
      XB          = X1 + X + KONTR1(3,1) * COSAL -
      -          KONTR1(4,1) * SINL

```



```

      YB      = X2 + Y + KONTR1(3,1) * SIN AL +
+      KONTR1(4,1) * COS AL
      CALL GLASS (XB,      YB,      Z,
,      PXB, PYB,
,      PARLR2)
C
      CALL RECTAB (PXA, PYA, PXB, PYB, FLOOD) C
C
      Рисуем треугольники :
      XA      = X1 + X + KONTR2(1,1) * COS AL -
-      KONTR2(2,1) * SIN AL
      YA      = X2 + Y + KONTR2(1,1) * SIN AL +
+      KONTR2(2,1) * COS AL
      CALL GLASS (XA,      YA,      Z,
,      PXA, PYA,
,      PARLR2)
      XB      = X1 + X + KONTR2(3,1) * COS AL -
-      KONTR2(4,1) * SIN AL
      YB      = X2 + Y + KONTR2(3,1) * SIN AL +
+      KONTR2(4,1) * COS AL
      CALL GLASS (XB,      YB,      Z,
,      PXB, PYB,
,      PARLR2)
C

```

V - 9

```

      CALL TRGLAB (PXA, PYA, PXB, PYB, FLOOD) C
      XA      = X1 + X + KONTR2(1,2) * COS AL -
-      KONTR2(2,2) * SIN AL
      YA      = X2 + Y + KONTR2(1,2) * SIN AL +
+      KONTR2(2,2) * COS AL
      CALL GLASS (XA,      YA,      Z,
,      PXA, PYA,
,      PARLR2)
      XB      = X1 + X + KONTR2(3,2) * COS AL -
-      KONTR2(4,2) * SIN AL
      YB      = X2 + Y + KONTR2(3,2) * SIN AL +
+      KONTR2(4,2) * COS AL
      CALL GLASS (XB,      YB,      Z,
,      PXB, PYB,
,      PARLR2)
C
      CALL TRGLAB (PXA, PYA, PXB, PYB, FLOOD) C
      RETURN
      END

```

V - 10

## ПРИЛОЖЕНИЕ 1.

Описание нижнего графического уровня комплекса

П.1.1.1. Описание интерфейсов графических примитивов,  
используемых для написания графических образов.

Все графические программы нижнего уровня PRADIS являются подпрограммами общего вида (SUBROUTINE) языка FORTRAN, за исключением подпрограммы-функции INDCOL

Для графических примитивов верхняя левая точка экрана имеет координаты 0,0, для текстовых - 1,1. Графические координаты устанавливаются в пикселах, текстовые - в символах (строки и столбцы). Все текстовые примитивы могут работать параллельно с графическими.

- 1) Программы установки графического режима, экрана, установки палитры, цветов, стиля рисования

Имя программы	Аргумент	Тип аргумента		Назначение
INITG	1 CODE	INTEGER * 4		Инициализация графики Если CODE= 0, успешно
SETCLR	-	-		Устанавливает цветовую палитру PRDIS
FINIT	-	-		Завершает графический режим
CLRSCR				Очистка экрана
CLIP	1 LEFT 2 TOP 3 RIGHT 4 BOTTOM	INTEGER INTEGER INTEGER INTEGER	4 4 4 4	Ограничивает область вывода на экран
COLOR	1 NCOL	INTEGER * 4		Устанавливает текущий цвет рисования по его порядковому номеру в палитре PRADIS

INDCOL (FUNCTION)	1 COL	INTEGER * 4	Определяет порядковый номер цвета в палитре PRADIS по параметру цвета, задаваемому пользователем в описании на входном языке. Используется совместно с п/п COLOR
LNSTYL	1 MASK	INTEGER * 4	Устанавливает маску, используемую для рисования линии

#### П1 - 1

Программа INITG устанавливает элементы поименованного COMMON-блока /GRCONF/. Его описание :

```

REAL * 8          RELYX

INTEGER * 4        XNMPXL, YNMPXL, XNMSMB, YNMSMB,
,                  NCOLOR, NMVPAG, MODE, IK,
,                  IS

COMMON /GRCONF/ RELYX,
,                  XNMPXL, YNMPXL, XNMSMB, YNMSMB,
,                  NCOLOR, NMVPAG, MODE, IK,
,                  IS

```

Назначение переменных COMMON-блока GRCONF :

RELYX - отношение масштаба изображения по оси OY к масштабу изображения по оси OX для данного типа экрана;

XNMPXL - количество доступных пикселей для данного графического экрана по оси абсцисс;

YNMPXL - количество доступных пикселей для данного графического экрана по оси ординат;

XNMSMB - количество символов текста для данного экрана по оси OX;

YNMSMB - количество символов текста для данного экрана по оси OY;

NCOLOR - количество доступных цветов;

NMVPAG - количество доступных видеостраниц (INITG всегда пытается обеспечить 2 страницы изображения, если это позволяет компьютер);

MODE - графическая мода экрана;

IK - код последней нажатой клавиши;

IS - скан-код последней нажатой клавиши.

Программа SETCLR устанавливает цветовую палитру PRADIS.

Номера цветов в палитре PRADIS для программы установки текущего цвета рисования (COLOR) :

0	-	черный	8	-	темносерый
1	-	яркосиний	9	-	светлосиний
2	-	зеленый	10	-	светлозеленый
3	-	коричневый	11	-	бело-голубой
4	-	темнофиолетовый	12	-	светлокрасный
5	-	розовокрасный	13	-	яркофиолетовый
6	-	светлокоричневый	14	-	яркожелтый
7	-	светлосерый	15	-	белый

## П1 - 2

### 2) Программы работы с видеостраницами

Имя программы	Аргумент	Тип аргумента	Назначение
AVISP	-	-	Делает активной видимую страницу изображения
ANVISP	-	-	Делает активной невидимую страницу изображения
VNVISP	-	-	Делает невидимую страницу видимой изображения

### 3) Программы отрисовки графических примитивов

Имя программы	Аргумент	Тип аргумента		Назначение
MOVEAB	1 XTEK 2 YTEK	INTEGER * 4 INTEGER * 4		Перемещение графического курсора в текущую позицию экрана
DRWPXL	1 X 2 Y	INTEGER INTEGER	4 4	Отрисовка отдельного пиксела в т. X, Y
DRAWAB	1 XTEK 2 YTEK	INTEGER * 4 INTEGER * 4		Рисование линии от текущей позиции до заданной
RECTAB	1 XA 2 YA 3 XB 4 YB 5 FLOOD	INTEGER * 4 INTEGER * 4 INTEGER * 4 INTEGER * 4 LOGICAL		Рисует прямоугольник со сторонами, парал. сторонам экрана. Если FLOOD = .TRUE., то прямоугольник "залит".
TRGLAB	1 XA 2 YA 3 XB 4 YB 5 FLOOD	INTEGER * 4 INTEGER * 4 INTEGER * 4 INTEGER * 4 LOGICAL		Рисует треугольник (XA,YA-XB,YB координаты ограничивающего прямоугольника. При движении от XA,YA к XB,YB треугольник остается слева. Стороны треугольника параллельны сторонам экрана. Если FLOOD = .TRUE., то треугольник "залит".

П1 - 3

### Программы отрисовки графических примитивов (продолжение)

Имя программы	Аргумент	Тип аргумента	Назначение
---------------	----------	---------------	------------

TRGLAN	1 XA 2 YA 3 XB 4 YB 5 XC 6 YC 7 FLOOD	INTEGER * 4 INTEGER * 4 INTEGER * 4 INTEGER * 4 INTEGER * 4 INTEGER * 4 LOGICAL	Рисует треугольник общего вида.  Если FLOOD = .TRUE., то треугольник "залит".
CIRCLE	1 XC 2 YC 3 R 4 FLOOD	INTEGER * 4 INTEGER * 4 INTEGER * 4 LOGICAL	Рисует на экране окр. радиусом R с центром в точке XC,YC. Если FLOOD = .TRUE., то рисуются круг.
ELLIPS	1 LEFT 2 TOP 3 RIGHT 4 BOTTOM 5 FLOOD	INTEGER * 4 INTEGER * 4 INTEGER * 4 INTEGER * 4 LOGICAL	Рисует на экране эллипс
CURSOR	1 XTEK 2 YTEK	INTEGER * 4 INTEGER * 4	Перемещение текстового курсора в заданную позицию
OUTTXT	1 STRING 2 N	CHAR * ( ) INTEGER 4	Печать текстовой строки длиной N

4) Дополнительные программы для работы в текстовом режиме  
(не используются  
непосредственно в ПК PRADIS)

Имя программы	Аргументы	Тип аргуме	Назначение
SCOLFN	1 COLOR	INTEGER *	Устанавливает текущий цвет фона



## П1 - 4

### П1.2. Получение в программе пользователя кода нажатой клавиши.

Для интерактивного получения кода нажатой клавиши используется вызов программы INKEY :

CALL INKEY (IK, IS),

где IK и IS - соответственно, код нажатой клавиши и ее скан-код.  
Переменные IK и IS - INTEGER \* 2.

Следует иметь ввиду, что после завершения текущего шага интегрирования буфер клавиатуры очищается.

Гарантируется получение кода клавиши и скан-кода следующих клавиш (если они есть на клавиатуре компьютера) :

Клавиша	Код клавиши	Скан	код
<ENTER>	28	13	
<ALT-C>	46	0	
<ALT-R>	19	0	
<F1>	59	0	
<F2>	60	0	
<F3>	61	0	
<Page Up>	73	0	
<Page Down>	81	0	

Программы пользователя, использующиеся в составе вычислительного ядра, не должны вызывать программу INKEY напрямую. Чтобы получить код и скан-код нажатой клавиши, нужно использовать переменные IK и IS COMMON блока GRCONF.



## ПРИЛОЖЕНИЕ 2.

Список зарезервированных имен

## СПИСОК ЗАРЕЗЕРВИРОВАННЫХ ИМЕН

ANVISP	ELLIPS	INITSC	RDVCT
ARCFL	ERRCOD	INKEY	RECTAB
AVISP	ERRFIL	INTEGR	REPLAC
BLOCKD	ERROR	KTRGVG	RESTR
CIRCLE	ERRPRN	LNSRCH	RUN
CHECKP	FILEND	LNSTYL	SAVERS
CLIP	FILOPN	MESSAG	SCOLFN
CLIPWN	FINDRC	MOVEAB	SETCLR
CLRSCR	FINIT	MSGSHI	STATE
COLOR	FLDSCR	NEXTX	STOPT
CURSOR	GAUSS	NMSHOW	STTITR
DBGLOC	GETADD	NOTAT	SWCFFY
DBGNTN	GETFLK	OUTRSL	SWSTP
DBGPAR	GLASS	OUTTIM	TESTG
DELSTP	GRCONF	OUTTXT	TRANS
DIROUT	GRFIDN	PARINT	TRANSO
DRAWAB	GRID	PREDCT	TRGLAB
DRWPXL	INDCOL	PRNCOD	TRGLAN
DSPGRF	INITG	PSTOUT	VISUAL
DSPIDN	INITR	RAD	VNVISP

Кроме того, для использования вычислительным ядром PRADIS зарезервированы имена S0001 - S9999

-----  
 | Y (1,1) | - частная производная первой силы (момента)  
 | | по первому перемещению (подсчитывается  
 ----- с учетом того, что  $dV1/dX1 = 0, dA1/DX1 = 0$ );  
 | |

. . .

	Y (N,1)	частная производная первой силы (момента) по N - му перемещению;
	Y (N+1,1)    ----- 	частная производная второй силы (момента) по первому перемещению;
	. . . 	
	Y (N*(J-1)+I,1)    ----- 	частная производная J - й силы по I - му перемещению;
	. . . 	
	Y (N*(J-1)+I,2)    -----    . . . 	частная производная J - й силы по I - й скорости (имея ввиду, что $dX_i/dV_i=0$ и $dA_i/dV_i = 0$ )

       - -	Y (N*(J-1)+I,3)   -----	частная производная J - й силы по I - му ускорению (здесь тоже dXi/dAi=0, dXi/dAi=0)
------------------------	----------------------------	--

Рис. 3.2. Структура якобиана модели элемента.

а) производные по перемещениям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1)	Y (2)	Y (3)
2 ( по оси Y)	Y (4)	Y (5)	Y (6)
3 (момент)	Y (7)	Y (8)	Y (9)

б) производные по скоростям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (10)	Y (11)	Y (12)
2 ( по оси Y)	Y (13)	Y (14)	Y (15)
3 (момент)	Y (16)	Y (17)	Y (18)

в) производные по ускорениям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (19)	Y (20)	Y (21)
2 ( по оси Y)	Y (22)	Y (23)	Y (24)
3 (момент)	Y (25)	Y (26)	Y (27)

Рис. 3.6. Структура якобиана в случае описания его как одномерного массива.



а) производные по перемещениям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1,1)	Y (2,1)	Y (3,1)
2 ( по оси Y)	Y (4,1)	Y (5,1)	Y (6,1)
3 (момент)	Y (7,1)	Y (8,1)	Y (9,1)

б) производные по скоростям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1,2)	Y (2,2)	Y (3,2)
2 ( по оси Y)	Y (4,2)	Y (5,2)	Y (6,2)
3 (момент)	Y (7,2)	Y (8,2)	Y (9,2)

в) производные по ускорениям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1,3)	Y (2,3)	Y (3,3)
2 ( по оси Y)	Y (4,3)	Y (5,3)	Y (6,3)
3 (момент)	Y (7,3)	Y (8,3)	Y (9,3)

Рис. 3.7. Структура якобиана в случае описания его как двухмерного массива.

а) производные по перемещениям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1,1,1)	Y (2,1,1)	Y (3,1,1)
2 ( по оси Y)	Y (1,2,1)	Y (2,2,1)	Y (3,2,1)
3 (момент)	Y (1,3,1)	Y (2,3,1)	Y (3,3,1)

б) производные по скоростям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1,1,2)	Y (2,1,2)	Y (3,1,2)
2 ( по оси Y)	Y (1,2,2)	Y (2,2,2)	Y (3,2,2)
3 (момент)	Y (1,3,2)	Y (2,3,2)	Y (3,3,2)

в) производные по ускорениям

N усилия	N потенциальной переменной		
	1	2	3
1 ( по оси X)	Y (1,1,3)	Y (2,1,3)	Y (3,1,3)
2 ( по оси Y)	Y (1,2,3)	Y (2,2,3)	Y (3,2,3)
3 (момент)	Y (1,3,3)	Y (2,3,3)	Y (3,3,3)

Рис. 3.8. Структура якобиана в случае описания его как трехмерного массива .