ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В.Келдыша РАН

А.В. Воронков, С.Л. Головков

Монитор пакета прикладных программ «РЕАКТОР»

Москва
2005

А.В. Воронков, С.Л. Головков
Монитор пакета прикладных программ «РЕАКТОР»

Аннотация

Описывается Интерфейс пользователя и Подсистема динамической визуализации, входящие в состав Монитора пакета прикладных программ «РЕАКТОР». Интерфейс пользователя ориентирован на прикладного специалиста и обеспечивает визуальное управление пакетом. Подсистема динамической визуализации обеспечивает визуальный контроль процесса выполнения задачи.

Описываются архитектура подсистем, функции и методы использования.

A.V. Voronkov, S.L. Golovkov
Monitor of the application package «REACTOR»

Abstract

The User Interface and the Subsystem for dynamic visualization, which are parts of the Monitor of the applications package "REACTOR" are described. The User Interface is oriented to the applied specialist and ensures visual control of the package. The Subsystem for dynamic visualization ensures visual monitoring of the task evaluation process. The architecture of subsystems, functions and the methods of use are described.

Работа выполнена при финансовой поддержке Российского Фонда Фундаментальных Исследований (проект № 05-01-08004-офи_э «Разработка и внедрение программного пакета полномасштабного моделирования основных нейтронно-физических и тепло-гидравлических процессов в ядерных реакторах различного типа и назначения»)

Содержание

Введение

Современный пакет прикладных программ представляет собой сложную систему, включающую в свой состав не только компоненты, собственно выполняющие расчеты, но и большой набор компонент, обеспечивающих эффективную эксплуатацию пакета. Фактически счетные компоненты пакета «погружены» в «управляющую среду», предоставляющую пользователю (прикладному специалисту) удобные сервисы для управления пакетом программ на всех этапах работы с задачей. Типичные сервисы, предоставляемые «управляющей средой», включают в себя

- подготовку исходных данных,
- подготовку и осуществление запуска задачи,
- хранение исходных, промежуточных и результирующих данных,
- контроль выполнения задачи,
- анализ результатов.

С нашей точки зрения оптимальным способом построения «управляющей среды» является создание набора инструментов, каждый из которых решает отдельную задачу управления и способен функционировать автономно, но в то же время в совокупности составляющих взаимосвязанную систему, обладающую всей необходимой функциональностью.

Такой подход обладает следующими преимуществами:

- возможность формировать управляющую среду, наиболее удобную конкретному пользователю;
- такую среду удобно проектировать и реализовывать, поскольку разработку каждого компонента можно поручать специалисту (специалистам), наиболее сведущему в данной области.

В предлагаемой статье описываются некоторые компоненты «управляющей среды» пакета прикладных программ «РЕАКТОР».

Краткая справка

Начиная с 1988 года, все программные разработки по нейтронно-физическим задачам в Институте прикладной математики им. М.В.Келдыша РАН проводятся в рамках модульной идеологии, что позволило объединить все программы в единый комплекс - пакет прикладных программ «РЕАКТОР» [1], который уже в течение многих лет эксплуатируется в ИПМ РАН и ряде других организаций. Пакет «РЕАКТОР» позволяет проводить численный расчет как стационарного состояния, так и расчет кампании ядерного реактора. Счет может проводиться в одно-, двух- и трехмерных геометриях, в различных приближениях по энергии (одно-, мало- и многогрупповое), с использованием различных приближений уравнения переноса (диффузионное, P1, кинетическое).

Пакет представляет собой совокупность большого числа независимых программ – ***функциональных модулей****. Каждая такая программа (модуль)*

решает логически завершенный фрагмент общей задачи, например, ввод геометрической информации, подготовка групповых констант, расчет нейтронных полей, выгорание, послерасчетная обработка и т.д. В настоящее время в пакете «РЕАКТОР» имеется несколько десятков функциональных модулей. Кроме того, пакет является открытой системой, позволяющей относительно легко наращивать функциональные возможности и подключать "чужие" программы.

Прикладной специалист для решения своей задачи должен сформировать и выполнить ***цепочку функциональных модулей****, каждый из которых в качестве входных данных использует выходные данные предшествующих модулей.*

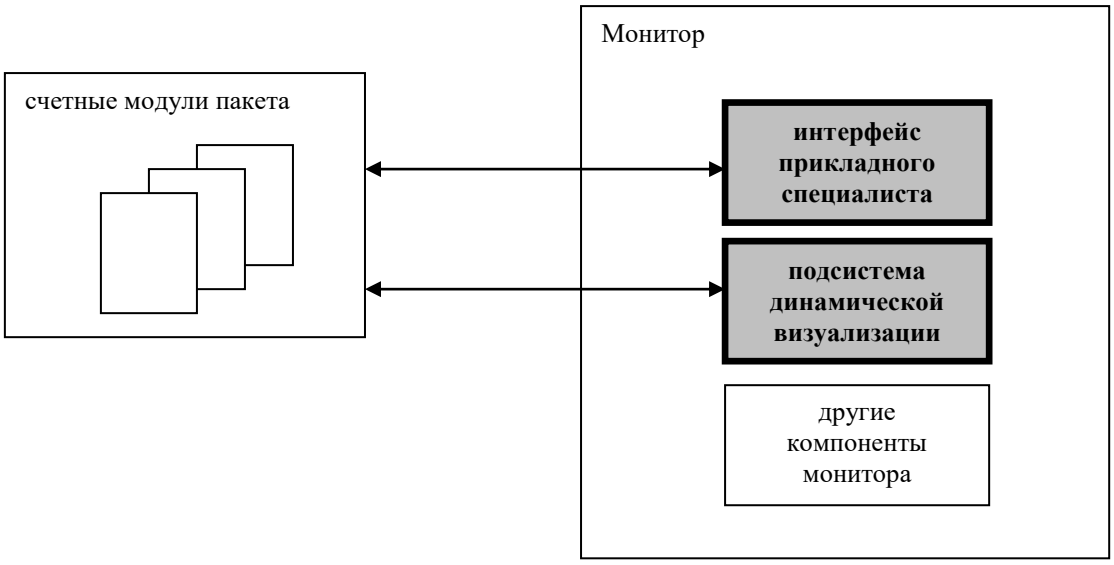
Пакет прикладных программ «РЕАКТОР» обладает «управляющей средой» (называемой «Монитор»), предназначенной для подготовки исходных данных, запуска задач на различных платформах, анализа результатов счета и т.п.

Монитор обеспечивает единую дисциплину (технологию) работы на различных программно-аппаратных платформах (рабочие станции, параллельные компьютеры). Характеристики Монитора обеспечивают наглядную, эффективную и безопасную эксплуатацию пакета.

Монитор представляет собой совокупность компонент (инструментов), обеспечивающих различные сервисы пользователя. Компоненты могут взаимодействовать друг с другом, а могут быть автономными подсистемами. Но в любом случае они построены в рамках единой идеологии и «покрывают» всю требуемую пользователю функциональность.

В настоящей статье описываются два компонента управляющей среды:

- **интерфейс прикладного специалиста**, обеспечивающий определение сценария задачи, запуск задачи и контроль выполнения задачи;
- **подсистема динамической визуализации**, обеспечивающая визуальный контроль содержательных объектов, возникающих в процессе счета.



1. Интерфейс прикладного специалиста

1.1. Основной тип пользователя и характер решаемых им задач

Приступая к разработке Интерфейса прикладного специалиста, чрезвычайно важно иметь хотя бы общее представление о пользователе, для которого предназначена разрабатываемая система, а также понимать характер решаемых им задач. Это сможет помочь в оценке целей проектирования системы, определить основные требования и концепции Интерфейса.

Пользователя создаваемого Интерфейса можно охарактеризовать следующим образом - это, как правило, прикладной специалист, общение которого с системой является составной и неотъемлемой частью его повседневной работы. Его работа регламентирована четкими правилами и соответствует вполне определенной области деятельности - решению задач вычислительного характера. Наш пользователь, как правило, уже имеет опыт работы с диалоговыми системами на персональном компьютере. Он воспринимает Интерфейс как рабочий инструмент. Его цель - используя Интерфейс, решить определенную прикладную задачу на вычислителе.

Решаемая прикладная задача является, как правило, достаточно хорошо структурированной, в которой данные и методы хорошо проверены. Эта задача предполагает непрерывную систематическую работу над ней с необходимостью периодической корректировки исходных данных на основе результатов уже проделанной работы.

Следует отметить, что при разработке Интерфейса нас не интересовали такие категории пользователей как профессиональные программисты или аналитики, а также случайные пользователи, потребности которых непредсказуемы, а работа которых не является обязательной и систематической. Необходимо отличать прикладного специалиста (т.е. пользователя, выполняющего свою профессиональную работу) от пользователя эпизодического. Нас не интересовали также начинающие пользователи без опыта работы с диалоговыми системами.

1.2. Базовая функциональность

Ориентируясь на выбранный тип пользователя, определим характеристики и функциональность **графического интерфейса прикладного специалиста** (далее – **Интерфейса**), исходя из потребностей простого и наглядного доступа к ресурсам пакета прикладных программ:

- общая визуальная организация Интерфейса должна обеспечивать наглядное графическое представление структуры и состава пакета.
- на этапе подготовки запуска задачи Интерфейс должен обеспечивать возможность визуального определение сценария прикладной задачи, в частности:
  - выбор последовательности выполняемых модулей пакета;

- указание для каждого модуля параметров запуска (имена входных и выходных файлов, имя Архива и т.п.).

- на этапе выполнение задачи Интерфейс должен обеспечивать
  - визуальный контроль процесса выполнения задачи;

- на этапе анализа результатов выполнения Интерфейс должен обеспечивать
  - возможность просмотра выходных файлов любого модуля задачи.

1.3. Сценарий выполнения задачи

Основным понятием, положенным в основу архитектуры Интерфейса, является **сценарий выполнения задачи** – описание последовательности работы модулей пакета прикладных программ, необходимых для выполнения прикладной задачи.

Грубо говоря, сценарий представляет собой перечень модулей пакета, упорядоченный в соответствии с последовательностью их выполнения, и набор параметров, определяющий режим работы каждого модуля.

Набор параметров, описывающий сценарий, можно разделить на две группы: **параметры сценария** и **параметры отдельного модуля**.

К первой группе относятся, например, такие параметры:

- имя рабочего каталога;
- имя и местонахождение файла с исходными данными;
- имя и местонахождение библиотеки констант;
- имя Архива.

Ко второй группе относятся, например, такие параметры:

- имя и местонахождение exe-файла модуля;
- имя файла с выходными данными;
- контрольная строка.

Для управления сценариями выбрана традиционная парадигма файла, т.е. над сценариями определены операции «прочитать», «сохранить», «сохранить как» и т.п.

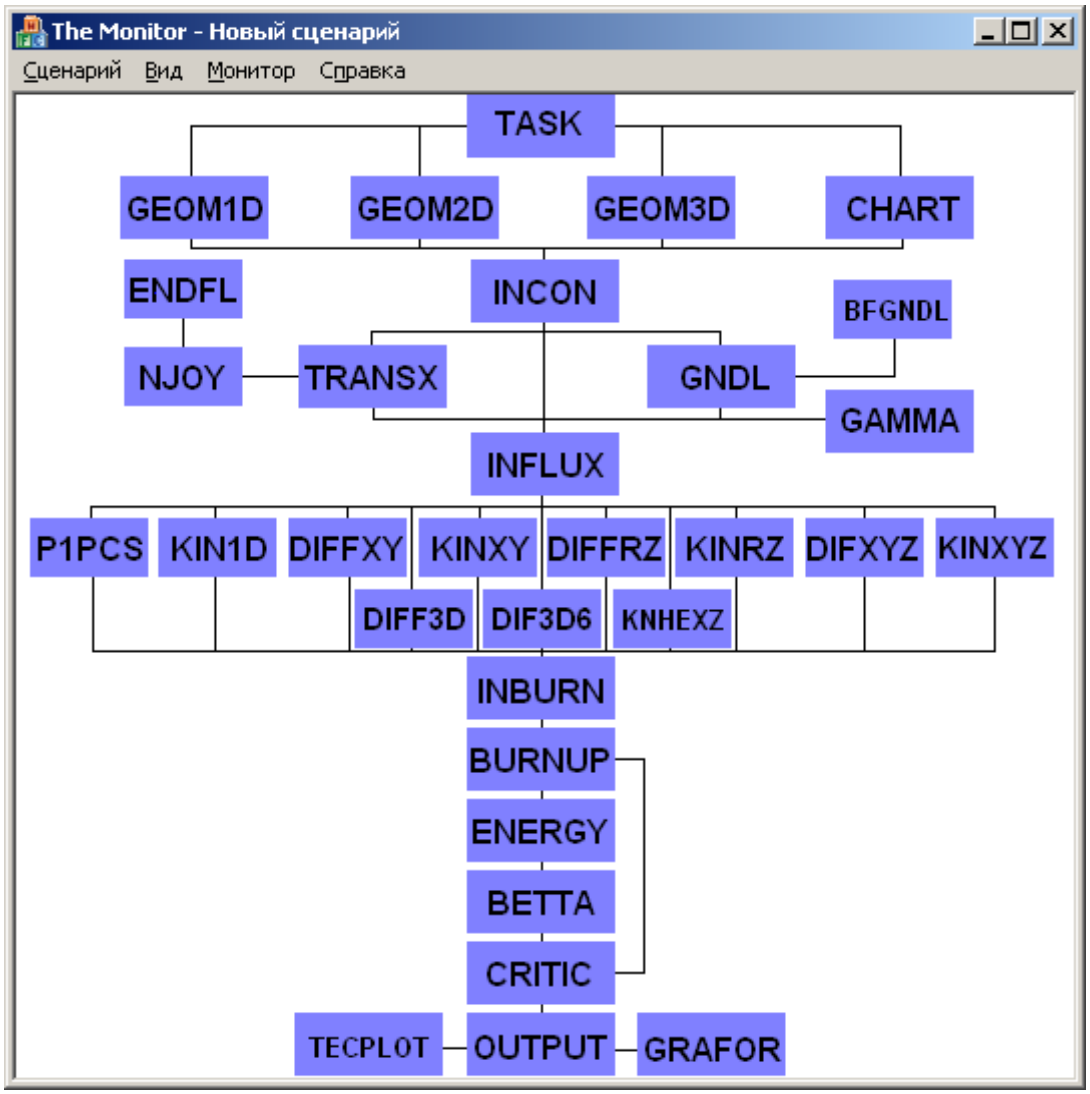
Понятие сценария выполнения задачи определяет основную **логику работы** пользователя, состоящую из следующих шагов:

- определить сценарий задачи;
- многократно выполнять сценарий, анализируя результаты выполнения и модифицируя исходные данные.

Рассмотрим возможности Интерфейса на примере выполнения пользователем сформулированных выше типовых действий, а именно:

- определение сценария задачи;
- выполнение сценария (т.е. выполнения задачи) и визуальный контроль этого выполнения;
- анализ результатов выполнения.

После запуска Интерфейса на экране появляется окно, содержащее графическое представление фрагмента пакета «РЕАКТОР»:

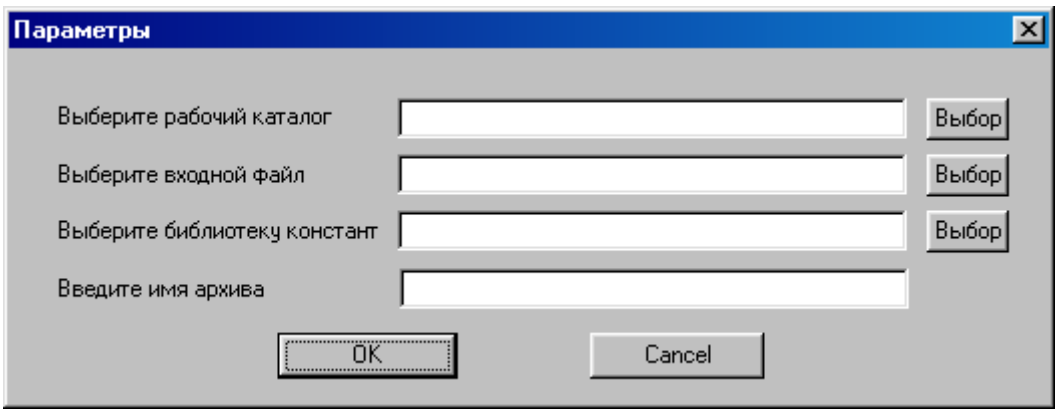


При этом Интерфейс создает новый пустой сценарий, имеющий имя «Новый сценарий».

Работа со сценарием схожа с работой с файлом в текстовом редакторе. Пользователь имеет возможность сохранять сценарий, создавать новый сценарий, загружать ранее сохраненный сценарий.

1.4. Создание сценария

Процесс создания сценария начинается с указания параметров сценария. Для этого необходимо выполнить команду меню «Монитор/Параметры». На экране появится панель:



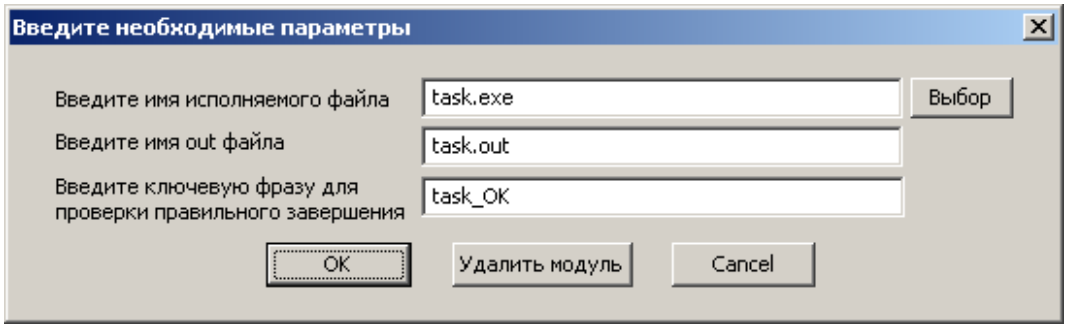
Введем необходимые параметры «вручную» или пользуясь кнопками «Выбор».

Замечания

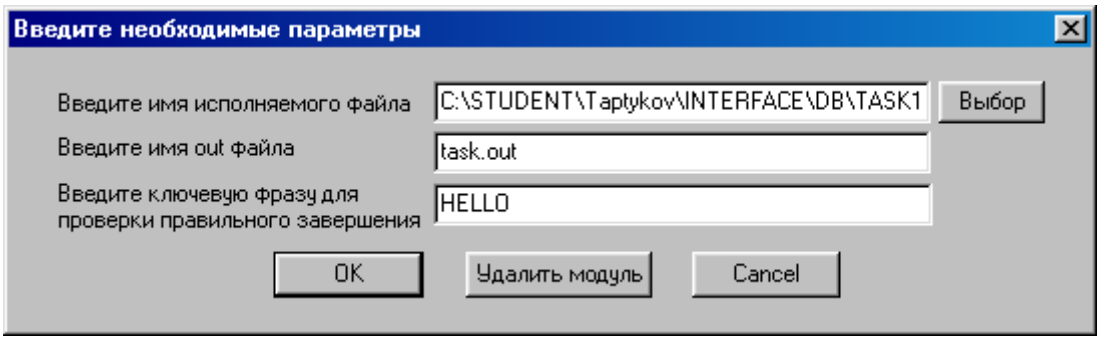
1. Рабочий каталог – это тот каталог, в котором будет выполняться задача. Он может быть пустым, в частности, он может быть создан с помощью панели, вызываемой нажатием кнопки «Выбор». В нем необязательно должны находиться exe-файлы модулей задачи и другие необходимые файлы. Интерфейс сам управляет всеми файлами: некоторые файлы (такие как файл исходных данных или библиотеку констант) он копирует в рабочий каталог, exe-файлы он вызывает на выполнение из тех каталогов, где они «дислоцируются».
2. Входной файл и библиотека констант могут находиться в любом месте.
3. Выходные файлы (out-файлы, Архив и временные файлы) создаются в рабочем каталоге.

После определения общих параметров сценария, необходимо указать цепочку модулей, составляющих задачу, и параметры запуска этих модулей. Делается это следующим образом.

Щелкаем левой кнопкой мыши по первому модулю в требуемой цепочке (это, конечно, модуль TASK). На экране появляется диалоговое окно, предлагающее пользователю ввести необходимые для работы модуля параметры:



Меняем при необходимости значения полей, установленных по умолчанию. Например, так:

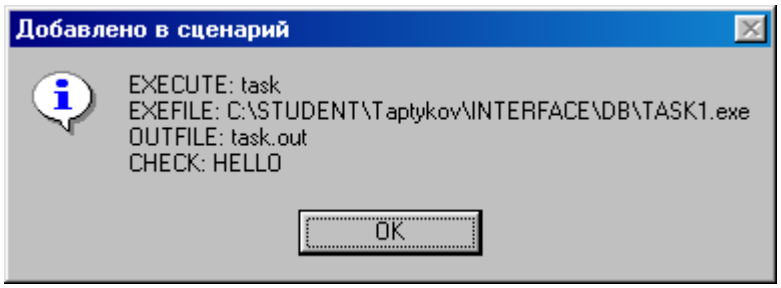


Последнее поле этой панели требует пояснения. В настоящей версии Интерфейса контроль правильности завершения работы модуля осуществляется следующим образом:

- порожденный процесс выполнения модуля должен завершиться с нулевым кодом;
- в результирующем out-файле должна присутствовать строка, указанная в поле «Введите ключевую фразу …».

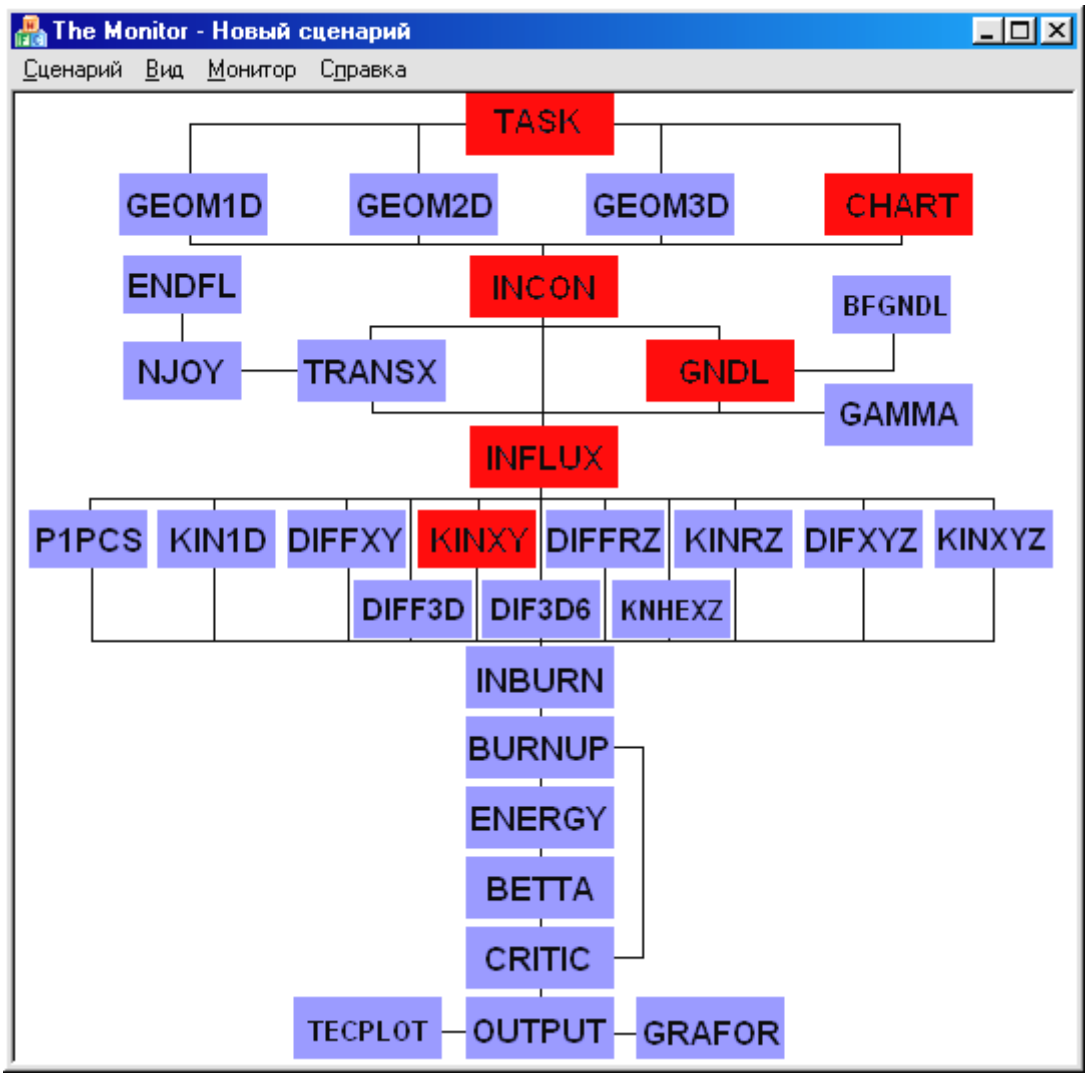
При выполнении этих двух условий Интерфейс считает, что модуль выполнился правильно и переходит к выполнению следующего модуля в цепочке. В противном случае выполнение сценария прерывается и выдается сообщение об ошибке.

Итак, после нажатия кнопки «ОК» на панели «Введите необходимые параметры» на экране появляется информационная панель:



Нажимаем кнопку «ОК» и информация о модуле заносится в сценарий, а модуль подсвечивается красным цветом:

Формируемый сценарий задачи индицируется главным окном Интерфейса. Например, в некоторый момент сценарий может выглядеть так:



Теперь мы можем перейти к выполнению сценария.

1.5. Выполнение сценария

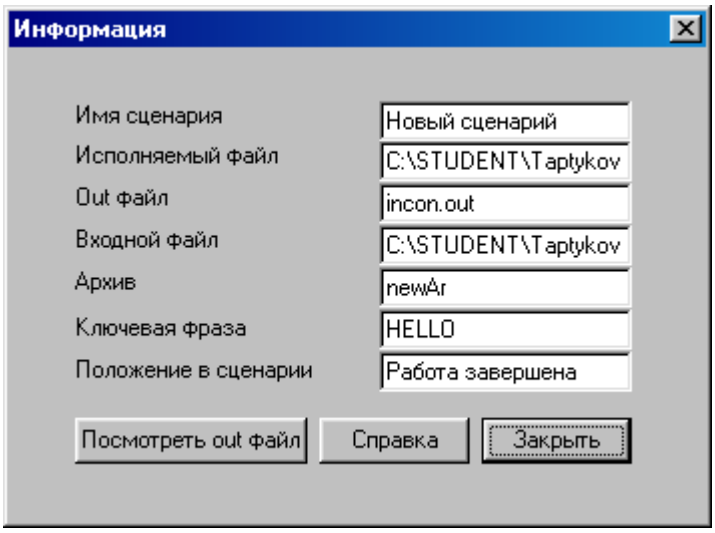
Интерфейс поддерживает два режима работы: автоматическое выполнение (всей задачи целиком) и пошаговое выполнение (с остановкой после выполнения каждого модуля).

В случае успешного выполнения очередного модуля, он подсвечивается зеленым цветом, а в случае ошибки работа приостанавливается и выводится сообщение об ошибке.

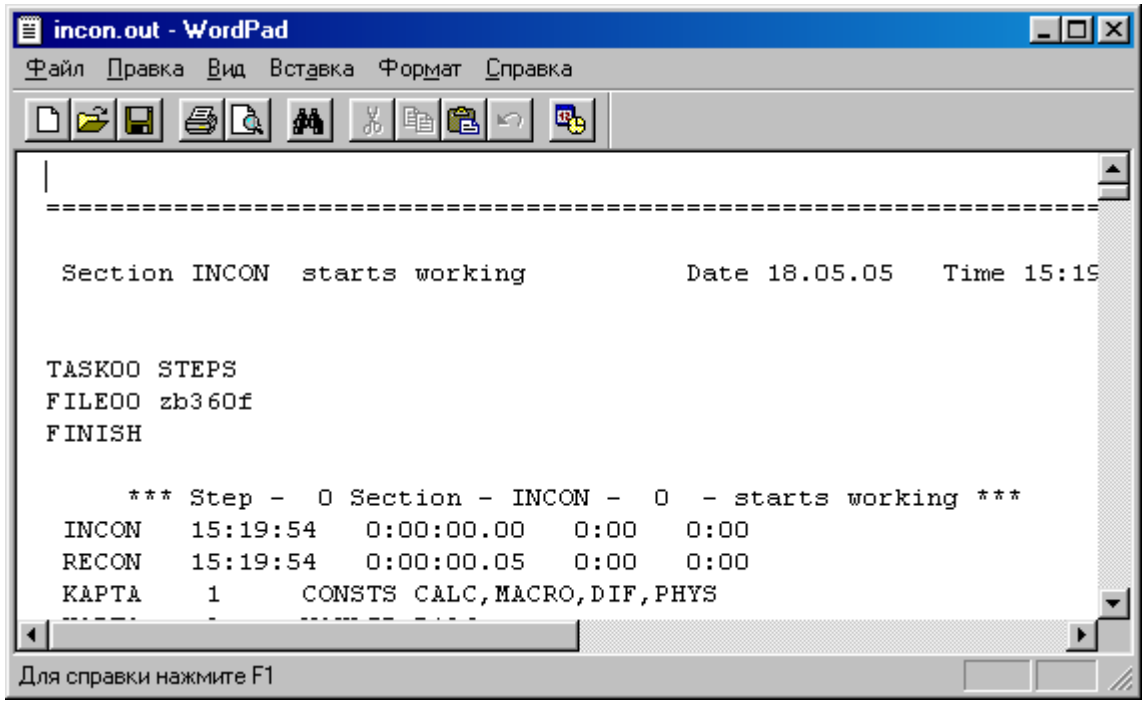
1.6. Анализ результатов

После выполнения модуля (в автоматическом или пошаговом режиме) мы может посмотреть out-файл этого модуля. Для этого щелкнем **правой** кнопкой мыши по интересующему нас модулю. Например, по модулю INCON. На

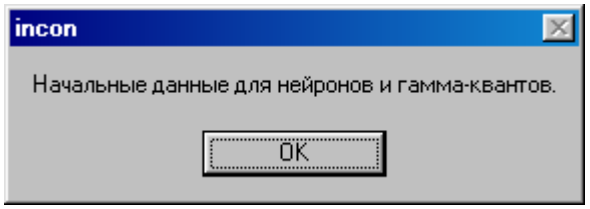
экране появится панель “Информация”, которое предоставляет информацию о параметрах данного модуля:



Если нажать кнопку «Посмотреть out-файл», то на экране появится окно редактора, содержащее требуемый out-файл:



Мы можем также получить краткую справку о функциональном назначении модуля, для чего требуется нажать кнопку «Справка»:



2. Подсистема динамической визуализации

Время счета реальных задач может быть весьма велико - многие сутки и даже недели. Поэтому возможность динамического контроля промежуточных данных, возникающих в процессе счета и, следовательно, возможность «раннего обнаружения» неправильного хода вычислений может привести к значительной экономии времени.

«Неправильный» ход вычислений может возникать либо из-за ошибок при формировании исходных данных, либо при расчете варианта, который оказался заведомо «неинтересным». Конечно, пользователь может следить за ходом вычислений по содержимому выходных файлов (out-файлов), однако такой мониторинг крайне неэффективен. Во-первых, потому, что в эти файлы поступает большое количество числовых данных, которые трудно анализировать «на лету». Во-вторых, несмотря на то, что в эти файлы поступает большое количество информации, большая ее часть все равно остается «за кадром». В частности, основные массивы данных попадают в Архив, который представляет собой совокупность бинарных файлов со сложной внутренней структурой и, следовательно, сложен для «визуального» контроля.

Поэтому возникла необходимость в средствах, позволяющих осуществлять автоматический или полуавтоматический мониторинг процесса выполнения задачи.

Часто наиболее удобной формой динамического контроля хода вычислений является визуальный анализ графиков, представляющих некоторые содержательные объекты, возникающие в процессе счета, например, каких-либо функционалов или физических полей. Пользователь – прикладной специалист - как правило, по внешнему виду такого графика сразу может сказать, есть ли нежелательные отклонения в ходе вычислений.

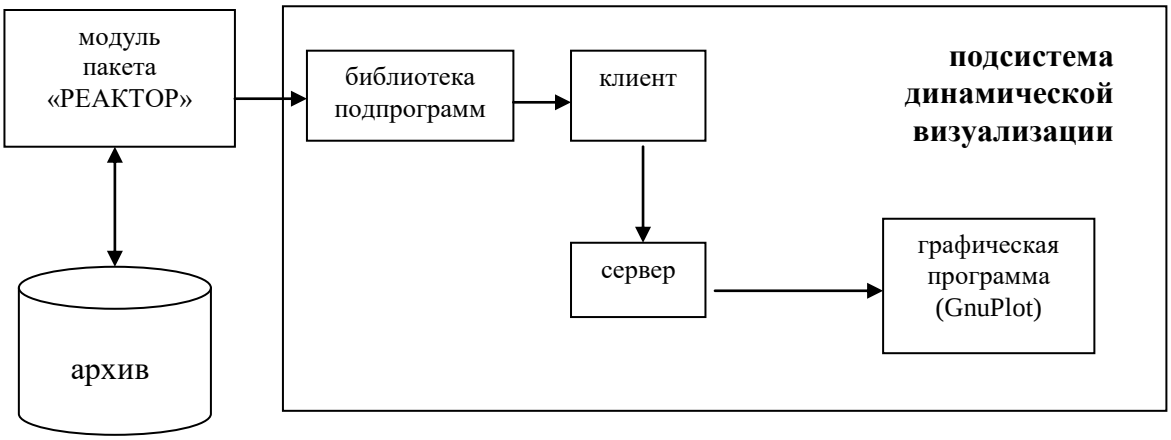
Для осуществления такого рода контроля были реализованы средства (названные **Подсистемой динамической визуализации**), позволяющие в заданные моменты счета (например, после очередной итерации) выводить на экран визуальное представление интересующего пользователя объекта.

2.1. Архитектура

Основная часть информации, порождаемой модулями пакета «РЕАКТОР», представляет собой массивы действительных чисел, записываемых в Архив пакета. Обычно анализ этих данных возможен только после завершения задачи.

Подсистема динамической визуализации позволяет сделать такой анализ оперативным и простым. Они позволяет без усилий следить за ходом вычислений и обнаруживать нежелательные отклонения на ранней стадии выполнения задачи.

Подсистема динамической визуализации реализованы в клиент-серверной архитектуре, использующей протокол TCP/IP. Архитектура этого инструмента изображена на следующем рисунке:



В требуемый момент модуль пакета обращается к Подсистеме динамической визуализации с требованием визуализировать некоторый массив данных. Это обращение имеет вид вызова подпрограммы визуализации с указанием имени массива и некоторых дополнительных параметров.

Подпрограмма визуализации формирует файл, содержащий элементы массива в том виде, который требуется графической системой. Затем вызывается Клиента, которому посылается запрос на перерисовку графика, используя новые данные. В свою очередь, Клиент связывается с Сервером и передает ему команду перерисовки графика. Сервер и Клиент общаются по протоколу TCP.

Сервер, получив команду, передает ее графической программе GnuPlot, которая в свою очередь отображает полученные данные в своем окне.

Такая, на первый взгляд несколько усложненная архитектура, была выбрана в расчете на эксплуатацию Подсистемы динамической визуализации в распределенной вычислительной среде, в которой модуль пакета «РЕАКТОР» выполняется на удаленном вычислителе (например, на многопроцессорном суперкомпьютере), а пользователь следит за выполнением модуля на своем персональном компьютере. В этом общем случае часть Подсистемы динамической визуализации (а именно – Клиент) будут функционировать на удаленном вычислительном узле, а часть (Сервер и графическая система) – на локальной машине пользователя.

Выбор графической системы GNUplot обусловлен тем обстоятельством, что она является свободно распространяемым программным обеспечением.

2.2. Использование

Для того чтобы воспользоваться Подсистемой динамической визуализации, необходимо выполнить следующие шаги:

- вставить в нужных местах программы модуля пакета вызовы подпрограммы визуализации и скомпилировать программу;
- перед запуском задачи (модуля) запустить Сервер Подсистемы динамической визуализации (впрочем, Сервер можно запустить и позже, во время счета);
- во время выполнения задачи можно управлять визуализацией с помощью **Панели управления**.

2.2.1. Подготовка программы

Интерфейс с Подсистемой динамической визуализации реализован в виде библиотеки подпрограмм, позволяющей указывать имена массивов, подлежащих визуализации и места в программе модуля, где необходимо их визуализировать.

Вызов библиотечной подпрограммы имеет, например, следующий формат:

```
CALL viewArray(point, info, param1, param2, param3, ARRAY,J)
```

где

```
integer                 point
character(len=*)        info
integer                 param1,param2,param3
double precision        ARRAY(1)
integer                 J(1)
```

Целочисленный параметр **point** является идентификатором конкретной точки вызова подпрограммы. Он позволяет управлять визуализацией во время счета задачи. С помощью Панели управления (см. ниже) можно указать идентификатор тех вызовов подпрограммы **viewArray**, которые должны «срабатывать». Если, например, указать значение идентификатора, равное 3, то тогда во время счета задачи все вызовы подпрограммы **viewArray**, у которых параметр **point** имеет значение 3, будут «срабатывать» и визуализировать соответствующие массивы. А все остальные вызовы (у которых иное значение параметра **point**) не будут выполнять каких-либо действий.

Следующие четыре параметра (**info**, **param1**, **param2** и **param3**) предназначены для идентификации графика. Параметр **info** представляет собой текстовую строку длиной не более 100 символов, которая выдается в качестве заголовка графика.

Более точно, эта текстовая строка является неким аналогом форматной строки языка Си. То есть, если в ней имеется фраза «**%1**», то при выдаче на экран (в окне графика) она будет заменена значением целочисленного параметра **param1**. Аналогично, фраза «**%2**» будет заменена значением целочисленного параметра **param2**, а фраза «**%3**» - значением целочисленного параметра **param3**.

Опишем более формально. Параметр **info** имеет следующий формат:

« . . . . %i1 . . . . %i2 . . . %i3 . . . . %iN . . . . #»

где i1, i2, i3, . . . , iN - числа в диапазоне [1,3].

Каждое включение «**%1**» заменяется на значение параметра **param1**, «**%2**» заменяется на значение параметра **param2**, «**%3**» заменяется на значение параметра **param3**.

Такой способ формирования заголовка графика достаточно удобен для того, чтобы указывать различную числовую информацию, идентифицирующую конкретный график, например – номер группы, номер итерации и т.п.

Важное замечание.

Обратите внимание, что строка, являющаяся значением параметра info, заканчивается символом «#». Этот символ необходим для корректной работы Подсистемы.

Параметр **ARRAY** указывает массив типа FLUX, который необходимо визуализировать.

Параметр **J** указывает массив типа LSTR, который необходим для правильной интерпретации массива **ARRAY**.

Приведем пример вызова подпрограммы **viewArray**. Например, вызов может иметь следующий вид:

```
CALL viewArray(3, 'Array: F0M  Point 3. NG=%1 ITER=%2#',          &
                                                NG, ITINT, 0, F0M(1,1,NG), JPR)
```

Идентификатор этого вызова – 3. То есть этот вызов будет «срабатывать», если указать с помощью Панели управления номер точки визуализации, равный трем.

Строка, указанная параметром info, содержит включения %1 и %2, которые при выдаче строки в качестве заголовка графика будет заменены значениями перемнных NG (номер группы) и ITINT (номер итерации) соответственно. Ниже, на скриншотах окна графики, мы увидим, как выглядит эта строка.

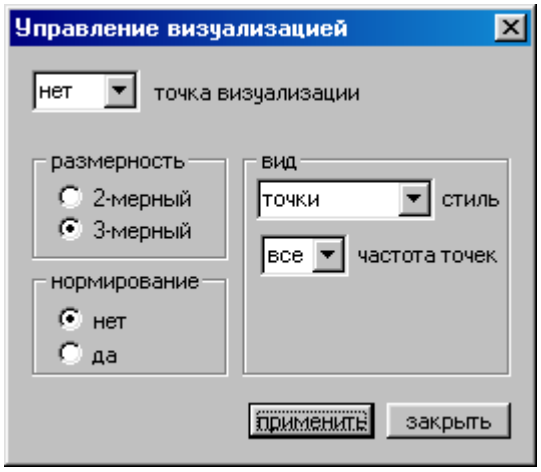
Визуализируется вырезка массива FOM с помощью массива JPR.

Подпрограмма **viewArray** обрабатывает данные, находящиеся в указанных массивах, формирует файл, содержащий эти данные в формате графической системы GNUplot, а затем с помощью программы Клиента обращается к Серверу и передает ему данные для визуализации.

2.2.2. Запуск Сервера

Сервер является автономной программой, поэтому его запуск можно сделать в любой момент: можно до запуска задачи, можно во время счета, можно завершить его выполнение, а затем вновь запустить и т.д.

После запуска Сервера на экране компьютера пользователя появляется Панель «Управление визуализацией»:



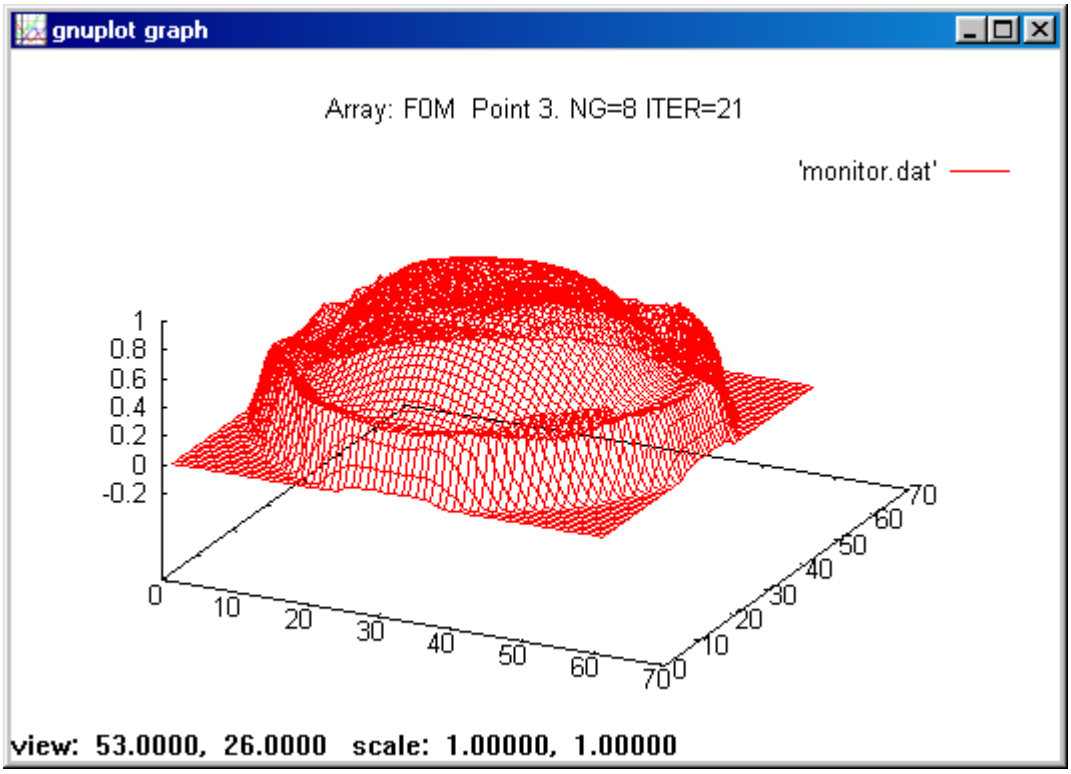
Управляющие элементы панели «Управление визуализацией» предназначены для манипулирования представлением графика.

После того, как на Панели сделаны какие-либо установки (например, в поле «точка визуализации» установлено значение 3, в поле «вид» установлено значение «линии», а переключатель «нормирование» установлен в положение «да»), необходимо нажать кнопку «**применить**». После этого очередной график, соответствующий вызову подпрограммы viewArray с параметром point, равным 3, будет построен в соответствии с новыми установками.

2.2.3. Управление визуализацией

Когда выполняемый модуль первый раз вызовет подпрограмму viewArray, на экране появится два «технических» окна, связанных с графической системой GNUplot. На них не следует обращать внимание, но и не следует закрывать – просто уберите их с экрана (минимизируйте).

Затем появится окно с требуемым графиком, которое далее будет обновляться по мере процесса счета задачи:



Изображение можно «развернуть» нужным ракурсом (с помощью мышки и/или клавиш), изменить цвет (с помощью меню GNUplot, распоможенному в левом верхнем углу окна), можно скопировать в буфер (клавиши Alt+PrintScrn).

Панель «Управление визуализацией» позволяет изменить точку визуализации и изменить представление графика.

Точка визуализации изменяется с помощью поля «точка визуализации». При выполнении модуля расчетного комплекса будут “срабатывать” те и только те вызовы подпрограммы viewArray, параметр point которых равен значению, указанному в поле «точка визуализации».

Остальные управляющие элементы панели «Управление визуализацией» предназначены для манипулирования представлением графика.

Переключатель «размерность». Этот переключатель определяет: двух- или трехмерный график будет выводиться на экран.

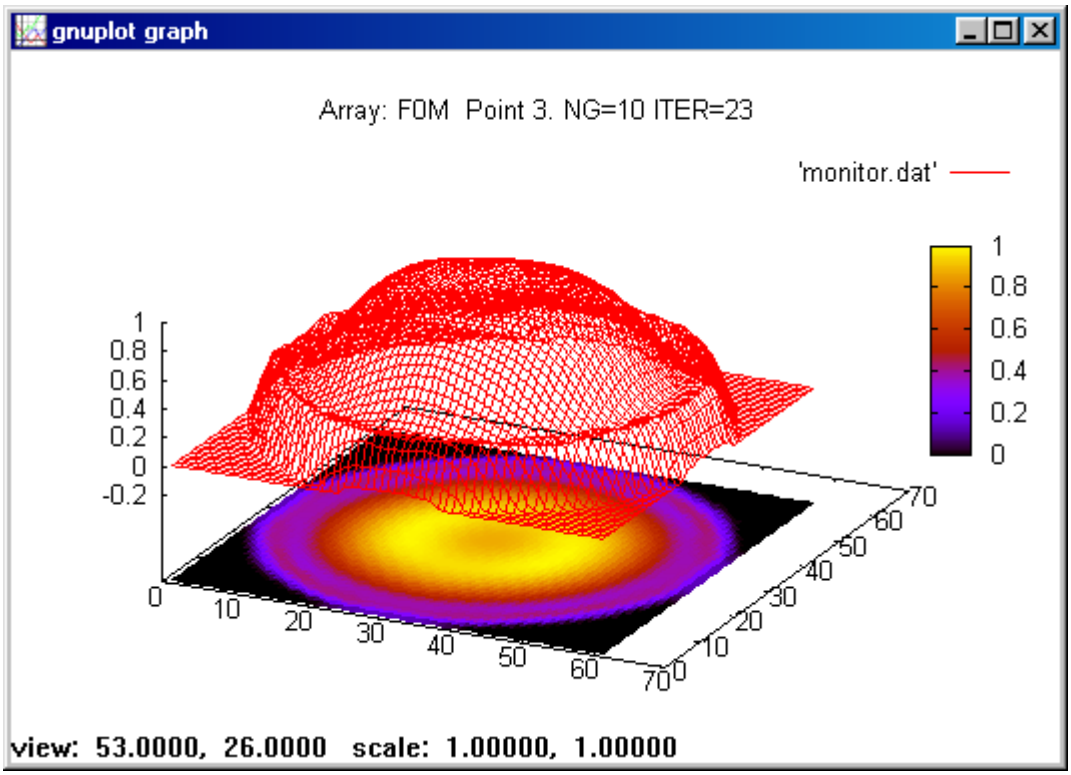
Переключатель «нормирование». При позиции «да» этого переключателя значения выводимого на график массива нормируется в диапазоне [0,1]. Такая нормировка позволяет избежать «дерганий» последовательно выводимых графиков.

Поле «частота точек». По умолчанию значение этого поля «все», что означает – на график выводятся все точки, соответствующие элементам массива. Если же график получился слишком «густой», то можно сделать его более «редким». Для этого установим в поле «частота точек» иное значение. Например, если значение этого поля будет равно «4», то на график будут выдаваться каждый четвертый элемент массива.

Поле «стиль графика». По умолчанию значение этого поля «точки». Если значение сменить на «линии», то получится график, приведенный выше. Другие значения этого поля: «поверхность», «дно», «контур», «контур (дно)».

Мы не будем описывать все эти стили – проще и нагляднее проверить их на практике. Ограничимся одним примером.

Так, после установки стиля “линии”, а потом стиля “дно” очередной график будет иметь следующий вид:



2.2.4. Файл предустановок графической системы

Каталог, в котором находится исполняемый файл Сервера Подсистемы динамической визуализации, содержит также **файл предустановок графической системы (monitor.plot).** Этот файл содержит команды графической системы GNUplot и выполняется один раз при первом вызове подпрограммы viewArray.

В настоящей версии файл предустановок содержит следующий набор команд:

```
set pointsize 0.5
set grid
```

Однако при необходимости в этой файл можно вставить любые другие команды, устанавливающие необходимые режимы работы системы GNUplot.

Заключение

В настоящем документе описаны два компонента Монитора, обслуживающего работу пакета прикладных программ «РЕАКТОР»: **Интерфейс прикладного специалиста** и **Подсистема динамической визуализации**.

Интерфейс является важной составляющей системы управления любым расчетным комплексом. От того, насколько интерфейс удобен той или иной категории пользователей и от того, насколько полно отражены их требования, зависит эффективность работы пользователей при решении прикладных задач.

Для пакета «РЕАКТОР» разработан пилотный вариант **Интерфейса**, ориентированного на пользователя – прикладного специалиста. Визуальная организация Интерфейса обеспечивает наглядное графическое представление структуры и состава пакета прикладных программ. Интерфейс обладает следующими функциональными возможностями:

- на этапе подготовки запуска задачи обеспечивает возможность визуального определение сценария прикладной задачи,
- на этапе выполнение задачи обеспечивает визуальный контроль процесса выполнения задачи,
- на этапе анализа результатов выполнения обеспечивает возможность просмотра выходных файлов любого модуля задачи.

Время счета реальных задач может быть весьма велико: многие сутки и даже недели. Поэтому возможность динамического контроля промежуточных данных, возникающих в процессе счета, и, следовательно, возможность «раннего обнаружения» неправильного хода вычислений может привести к значительной экономии времени.

Для решения этой задачи разработана **Подсистема динамической визуализации**, обеспечивающая оперативный анализ промежуточных данных, возникающих в процессе работы с пакетом «РЕАКТОР».

Эта Подсистема позволяет в заданные моменты счета (например, после очередной итерации) выводить на экран визуальное представление интересующего пользователя объекта. Такая возможность позволяет без усилий следить за ходом вычислений и обнаруживать нежелательные отклонения на ранней стадии выполнения задачи.

Содержащиеся в данном документе описание Интерфейса пользователя и Подсистемы динамической визуализации могут служить Инструкцией для работы с этими компонентами Монитора пакета «РЕАКТОР».

Литература

1. A.Voronkov, V.Arzhanov. REACTOR – Program System for Neutron-Physical Calculations. Proc. International Topical Meeting: Advances in Mathematics, Computations, and Reactor Physics, USA, Vol5, April 28 – May 2, 1991.