РОССИЙСКАЯ АКАДЕМИЯ НАУК

ИНСТИТУТ
ПРОБЛЕМ БЕЗОПАСНОГО
РАЗВИТИЯ АТОМНОЙ ЭНЕРГЕТИКИ

RUSSIAN ACADEMY OF
SCIENCES

NUCLEAR
SAFETY INSTITUTE

Препринт № IBRAE–95–05

Preprint IBRAE–95–05

В. В. Варенков, В. А. Первичко, А. Г. Пошков, В. В. Чуданов

ПРОГРАММНАЯ СИСТЕМА RCS: ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ ДЛЯ МОДЕЛИРОВАНИЯ ФИЗИЧЕСКИХ ПРОЦЕССОВ

Москва
1995

Moscow
1995

УДК 681.3.06

Варенков В.В., Первичко В.А., Попков А.Г., Чуданов В.В. ПРОГРАММНАЯ СИСТЕМА RCS: ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ ДЛЯ МОДЕЛИРОВАНИЯ ФИЗИЧЕСКИХ ПРОЦЕССОВ. Препринт № IBRAE–95–05. Москва: Институт проблем безопасного развития атомной энергетики РАН, 1995. 16 с.

Аннотация

В работе рассматривается программная система для организации решения задач математической физики. Описываются принципы работы с вычислительными модулями и данными. Приводится архитектура системы и типы обрабатываемых объектов. На конкретном примере демонстрируется организация вычислительной среды.

Varenkov V.V., Pervichko V.A., Popkov A.G., Chudanov V.V. PROGRAM SYSTEM RCS: ORGANIZATION OF CALCULATIONS FOR MODELING OF PHYSICS PROBLEMS (in Russian). Preprint IBRAE–95–05. Moscow: Nuclear Safety Institute, 1995. 16 p.

Abstract

This paper describes a program system for organization of problem solving in field of mathematical physics. Principles of manipulations with calculation modules and data are considered. System structure and types of handled objects are described. An example demonstrates the organization of computing environment.

ПРОГРАММНАЯ СИСТЕМА RCS: ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ ДЛЯ МОДЕЛИРОВАНИЯ ФИЗИЧЕСКИХ ПРОЦЕССОВ

В. В. Варенков, В. А. Первичко, А. Г. Попков, В. В. Чуданов

ИНСТИТУТ ПРОБЛЕМ БЕЗОПАСНОГО РАЗВИТИЯ АТОМНОЙ ЭНЕРГЕТИКИ

113191 Москва, ул Б.Тульская, 52

тел.: (095) 952–24–21, факс: (095) 230–20–29, эл. почта: pbl@ibrae.msk.su

Введение

При моделировании сложных физических процессов приходится иметь дело с большим количеством вычислительных модулей и данных. К такой ситуации приводит, в частности, моделирование процессов, происходящих при тяжелых авариях на АЭС. В этих задачах для различных сценариев аварии в определенные моменты времени подключаются либо отключаются те ли иные вычислительные блоки, причем для одних и тех же процессов могут существовать различные математические модели и их реализации. Возникает необходимость в удобных средствах для подготовки входных данных и для обработки и анализа большого объема полученных результатов.

За десятилетия использования вычислительной техники накоплен значительный опыт в организации вычислительного эксперимента, в том числе для моделирования тяжелых аварий на АЭС. Опираясь на этот опыт, на возросшие возможности вычислительной техники и собственные наработки в математическом моделировании и обработке данных, авторы разработали программную систему RCS с целью поддержки проводимых работ по моделированию тяжелых аварий на АЭС.

Система включает средства для работы с вычислительными модулями и данными. Эти средства позволяют подключать различные наборы вычислительных модулей для решения конкретных задач и обеспечивают базовые механизмы для подготовки и обработки данных, а также поддерживают обмен данными между отдельными подсистемами вычислительного комплекса.

Обычно в вычислительных модулях на языке FORTRAN данные передаются через списки аргументов и COMMON–блоки. В описываемой ниже программной системе обмен между модулями производится по именам данных, хранимых как в оперативной памяти, так и в файлах, что позволяет добавлять или исключать общие данные локально внутри одного модуля без какого–либо внесения изменений в других модулях. Добавление новых модулей не изменяет общего вида системы.

При разработке комплекса авторами были проанализированы различные вычислительные коды по тяжелым авариям и известные коды для решения задач тепломассопереноса.

Рассматривались следующие аспекты:

— взаимодействие с пользователем
— типы данных
— способы задания и инициализации данных
— хранение данных
— преобразования данных
— формы представления данных (графика, текст)

Многие командные языки в существующих кодах имеют привязку к предметной области либо к особенностям конкретной реализации. С учетом того, что основные блоки большинства из этих кодов создавалось давно, когда вычислительная техника обладала небогатыми возможностями:

эти коды несут "отпечаток перфокарт". Тем не менее в процессе развития вычислительных кодов наработано много идей и имеется богатая практика по работе с данными. Так, авторам представляются заслуживающими внимания элементы архитектуры кода ICARE [1], в частности, организация внутренней базы данных и подсистема CHECKER, реализованные в составе системы RSYGAL.

Разработанные средства по обработке данных являются логическим продолжением идей, заложенных в постпроцессоре VV–2D [2].

Разработчики стремились к тому, чтобы реализуемая система была переносима на широкий диапазон машин с минимальными усилиями. В качестве основы для вычислительных модулей был взят язык FORTRAN, а для обработки данных — язык C. В настоящее время существует реализация на персональном компьютере IBM PC и на графстанции SUN SPARCstation.

1 Назначение системы

— обеспечить программные средства и унифицированные технологические правила для поддержки моделирования физических процессов в области математической физики;

— обеспечить удобство манипулирования с большим количеством вычислительных модулей и большим объемом разнотипных данных;

— предоставить единый механизм обмена данными между вычислительными модулями и подсистемами обработки данных;

— предоставить широкий набор универсальных средств подготовки данных и обработки результатов;

— дать возможность пользователю подключать свои модели и настраивать систему на работу с конкретными данными.

2 Основные черты и возможности

— Язык команд, позволяющий работать как в интерактивном режиме, так и посредством вызова предварительно заготовленных процедур.

— Наличие подсистем для подготовки, преобразования и визуализации данных.

— Переносимость данных (специальные текстовые и двоичные форматы) и программ (базовые ФОРТРАН, C) на различные платформы.

— Механизмы вызова вычислительных модулей с аргументами.

— Гибкость в построении программных кодов: компановка вычислительных модулей, написанных на различных алгоритмических языках и различных подсистем работы с данными.

— Возможность использования отдельных компонент системы или их комбинаций в качестве независимых подсистем: отдельный препроцессор, независимый постпроцессор, визуализатор данных, генератор форм для выдачи содержимых файлов данных.

3 Принципы работы системы

Функционирование программного кода и его отдельных подсистем рассматривается как процесс преобразования данных: вычислительные процедуры потребляют входные данные, преобразуют их в выходные или порождают новые. В этом смысле, данные можно трактовать как некоторое пространство переменных, а код — как сложный оператор, действующий на эти переменные.

Разработанная система ориентирована на решение задач в области математической физики, в которых один и тот же набор переменных имеет разные значения для некоторой последовательности моментов времени. Для поддержки работы в этом классе задач авторами было разработано специальное средство хранения данных — "самодокументируемый файл данных" [3], для краткости называемый SDF–файлом. SDF–файл помимо самих данных содержит их описание (имя,тип,размерность и т.п.), что дает основу для построения универсальных средств подготовки и обработки данных. На основе SDF–файлов организуются информационные потоки в интегральном коде RASPLAV [4].

Данные в SDF–файле понимаются всеми компонентами системы, включая вычислительные процедуры, программы обработки и визуализации, а также процедуры, введенные пользователем. Это обеспечивается за счет используемой единой структуры файлов, и соответствующих библиотек программ чтения/записи данных, позволяющих производить операции обмена с внешней памятью из различных процедур, в том числе, написанных на разных алгоритмических языках. Файл такого формата представляет собой базу данных, в котором содержится вся необходимая информация по переменным, в том числе их имена, типы, размеры, значения и комментарии. Ключевым параметром является имя переменной.

Предусмотрен механизм переносимости данных на различные типы платформ с использованием файлов как текстового так и двоичного форматов. На первом этапе разработок в качестве базовых платформ используются IBM PC и SUN SPARCstation.

Базы данных по свойствам материалов, необходимые константы, управляющие параметры, сеточные функции и т.д. хранятся в том же формате и соответственно поддерживаются единым аппаратом чтения/записи.

Вместе с тем, предусмотрена возможность использования простейших текстовых форматов (типа столбцов чисел) непосредственно, без дополнительных преобразований. В случае необходимости использования данных уникальных форматов (в качестве входных данных или для связи с другими кодами) требуется написание специальных конверторов между форматами. Имеется опыт преобразования файлов форматов кодов WECHSL, DYNA, VANESSA.

Проверка входных данных на допустимый дипазон, соответствие типов при передаче параметров и ряда других ошибочных ситуаций осуществляется блоком контроля данных.

Пользователь может просмотреть данные в цифровом или графическом виде. Для просмотра значений большого массива предусмотрены возможности по разбивке выдачи на порции, а также выделения части массива. Графически данные могут быть представлены в одномерном, двухмерном и трехмерном видах.

Вычислительные модули системы могут быть реализованы на разных языках программирования. В настоящее время в качестве базовых используются алгоритмические языки С и ФОРТРАН. При необходимости возможно расширение на другие языки. Предполагается, что вычислительное ядро системы в основном будет реализоваться на ФОРТРАНЕ, а собственно интерпретатор командного языка системы, блоки подготовки данных, визуализации, обработки — на языке С. Система обеспечивает согласование модулей при выполнении ряда принятых соглашений на передачу параметров. Обмен данными между вычислительными модулями осуществляется через файлы.

Система является открытой: пользователь системы в соответствии с принятыми правилами может создавать новые модули и включать их в систему. Гибкость в подключении различных вычислительных модулей с различными наборами входных и выходных данных достигается за счет использования специального механизма передачи параметров.

Часто употребимые вычислительные модули собраны в специальной системной директории.

Входные и выходные данные, общие для всех моделей или группы моделей хранятся в общем информационном поле и могут претерпевать изменения после прохода каждой вызываемой модели. Входные и выходные данные, необходимые только одной модели хранятся в локальных файлах этой модели.

При работе с системами обработки и представления данных (которые обычно носят название постпроцессор) пользователь применяет специальные языки, чтобы формулировать необходимые действия и указывать наборы данных. Язык может иметь форму текстовых команд, либо реализовываться через диалоговые окна и пункты меню графической оболочки.

Диалоговые оболочки удобнее для пользователя в силу их наглядности. Однако они требуют более сложного и обьемного программного обеспечения, к тому же возможны проблемы при переходе на другие вычислительные платформы.

Даже имея развитый диалоговый интерфейс, пользователю часто требуется многократно выполнять одни и те же последовательности действий, что порождает необходимость в средствах задания процедур, а это приводит к необходимости иметь язык обработки данных. Так, при проведении вычислительного эксперимента пользователю удобнее написать некоторые процедуры по обработке и визуализации данных и затем их запускать автоматически вместо многочисленных выборов пунктов меню и диалоговых панелей.

Стандартизованный командный язык может быть нижнем уровнем для построения диалоговых систем. В процессе работы диалоговая оболочка просто генерирует команды, единые для

всех платформ, при этом реализация команд может быть машинно–зависимой.

Процедуры, записанные в текстовых файлах на стандартизованном командном языке удобно передавать по сетям или через магнитные носители на другие аппаратные средства. Можно создавать библиотеки процедур.

Реализованный командный язык обеспечивает гибкость в построении программных кодов: позволяет компоновать вычислительные модули и различные подсистемы работы с данными. Обеспечивает возможность введения новых пользовательских модулей, использования их в различных комбинациях. Работа с командным процессором происходит либо в режиме интерпретации команд, путем последовательного набора команд, либо путем запуска командного файла. Командный файл предварительно создается с помощью текстового редактора; в перспективе возможно создание такого файла с помощью интерактивных процедур.

При работе системы генерируютя сообщения нескольких уровней, информирующие о состоянии системы и возникающих ошибках.

Планируется создание подсистемы ведения архивов расчетных данных, обеспечивающей автоматизированное сопровождение вычислительного эксперимента.

4 Типы объектов системы

Основные типы обьектов, с которыми работает пользователь в системе RCS:

— **Переменная**
содержит данные и их описание;

— **SDF–файл**
содержит набор переменных с их описаниями и значениями для разных моментов времени; все переменные каждого открытого SDF–файла образуют свой список в памяти;

— **Номер записи SDF–файла**
часто интерпретируется пользователем как "момент времени": значение выделенной переменной для идентификации записи в SDF–файле;

— **SDT–файл**
текстовый вариант SDF–файла;

— **ASCII–файл**
текстовый файл с данными (обычно в виде колонок чисел), который может генерироваться любой программой;

— **Модуль**
вычислительный модуль в виде отдельного исполняемого файла; каждый модуль сопровождается текстовым файлом *.ARG с описанием формальных параметров и файлом *.PAR с соответствием формальных и фактических параметров; файл *.PAR создается автоматически при работе с командным языком либо формируется с помощью редактора текстов;

— **Графический обьект**
обьект одного из следующих типов: текст, одномерные зависимости, двухмерные линии уровня, линии уровня на поверхности в трехмерном пространстве, трехмерные траектории, карта области, сетка, векторное поле, легенда, а также составные объекты: график и область графика;

— **Фильм**
последовательность графиков, последовательно воспроизводимых на экране дисплея; часто — это один и тот же график, показывающий данные в графическом виде на разные моменты времени, отображая таким образом динамику процесса;

— **PCX–файл**
графический файл в формате PCX;

— **PLT–файл**
графический файл в формате PLT;

— **PS–файл**
графический файл в формате PostScript;

— **CMD–файл**
командный файл в текстовом формате, содержащий некоторую процедуру обработки и визуализации данных в виде последовательности команд; в форме командных файлов хранится также текущее состояние системы.

Пользователь открывает существующие и создает новые SDF–файлы. Каждый SDF–файл содержит свой набор переменных, значения которых хранятся в некотором количестве записей. SDF–файл может содержать входные или выходные данные для вычислительных модулей.

Пользователь создает новые переменные или уничтожает существующие, а также модифицирует атрибуты и значения существующих переменных. Модификация значений переменных может производится в виде формульного выражения от других переменных, непосредственным присваиванием, а также чтением из текстовых файлов (при реализации командного языка в диалоговой оболочке способов задания может быть больше).

5 Основные компоненты среды

На рисунке 1 показана общая структура вычислительной среды RCS. Ниже приводятся пояснения для каждого блока, представленного на этом рисунке.

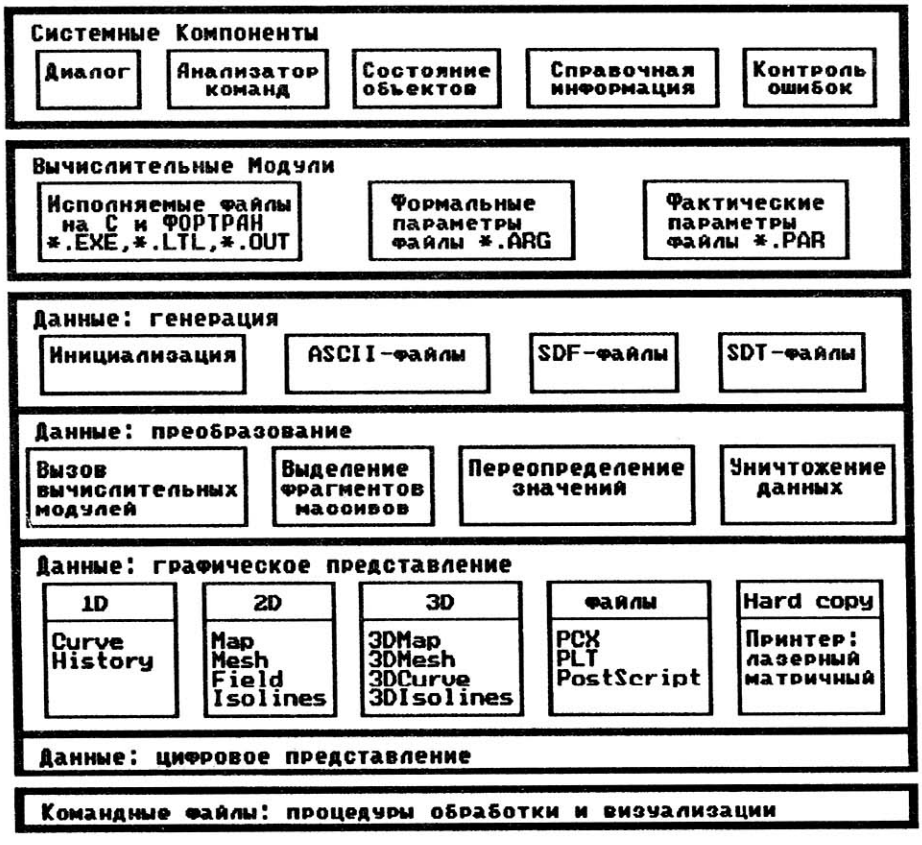

Рис.1. Основные компоненты среды RCS

— Блок Диалога с Пользователем.
Обеспечивает ввод команд пользователя и выдачу на экран дисплея диагностических и информационных сообщений.

— Анализатор Команд.
Анализирует синтаксис команды, определяя задаваемое действие и его параметры.

— Блок Выдачи Информации о Состоянии Системы и Отдельных Компонент.
Позволяет получить на экране дисплея обобщенную информацию о текущем состоянии системы или ее отдельных компонент или подробную информацию об отдельных объектах системы : файлах, переменных, графических объектах и т.п.

— Блок Выдачи Справочной Информации.
Выдает пользователю информацию о допустимых командах и их синтаксисе.

— Блок Контроля Ошибок и Выдачи Диагностических Сообщений.
Диагностирует как семантическую неправильность команд (например, отсутствие в системе объекта с заданным именем), так и динамические ошибки (например, недостаточно памяти для запуска вычислительного модуля).

— Блок Генерации Переменных.

  — Инициализация из языка (явное перечисление, цикл).
  Пользователь имеет возможность средствами командного языка определить новую переменную и проинициализировать ее. При задании массива в явном виде перечисляются его значения; для инициализации массивов существуют более компактные формы записи. Пользователь может также изменить отдельные значения массива или проинициализировать переменную значениями некоторой существующей переменной.

  — Чтение и запись табличных данных в текстовом виде.
  Данные изначально могут подготавливаться в текстовых файлах, порождаемых из различных вычислительных и обрабатывающих программ или заносятся текстовым редактором. Командный язык предусматривает возможность вводить задаваемые таким образом данные и выводить в аналогичном формате.

  — SDF–файлы.
  SDF–файлы хранят именованные данные в двоичном виде. Эти данные могут являться результатом работы некоторых программ, но для конкретного вычислительного модуля выступают в роли начальных данных. В свою очередь, вычислительный модуль может сгенерировать входные данные для другого модуля.

  — SDT–файлы.
  Текстовый аналог SDF–файлов. Специальными командами можно преобразовать файлы из одного представления в другое. Текстовый формат облегчает перенос данных между различными платформами.

— Блок Вызова Вычислительных Модулей.
Вычислительный модуль в нашем контексте предполагает вызов из командного языка и рассматривается как некий преобразователь данных, хранимых как правило, в SDF–файлах. Причем модуль может дополнительно порождать новые данные и SDF–файлы.
Вычислительный модуль часто реализуется на языке ФОРТРАН в силу его распространенности для решения задач математического моделирования. Поэтому в описываемой системе для данного языка программирования реализованы библиотеки по работе с SDF–файлами. Имеется также поддержка для языка С. Кроме того, любая программа на любом языке программирования может порождать текстовые табличные данные, воспринимаемые средствами командного языка.
Вычислительный модуль сопровождается файлом формальных параметров *.ARG.
При вызове модуля используется файл соответствия формальных и фактических параметров *.PAR , как правило, порождаемый средствами командного языка.

— Блок Преобразования Переменных.

  — Вызов вычислительных модулей.
  Преобразует набор данных в существующих SDF–файлах или порождает новые данные. В частном случае может добавлять новые записи в файлы (например, значения массивов на очередном временном шаге). Общеупотребительные вычислительные процедуры хранятся в системной директории описываемой системы.

— Сечение, фрагмент массива.
Командный язык предоставляет ряд средств для преобразования данных, такие как переприсваивание массивов и их отдельных элементов, выделение сечений массивов и их фрагментов, а таже порождение новых и уничтожение существующих переменных. Такого рода действия могут быть собраны в командных файлах и их запуск аналогично запуску вычислительных модулей производит преобразование данных.
Общеупотребительные командные файлы хранятся в системной директории описываемого комлекса.

— Блок Визуализации Данных в Цифровом Виде.
Позволяет просмотреть на экране дисплея данные в цифровом виде. Имеется возможность задания формата чисел и количества столбцов для выдачи. Длинные массивы нарезаются полосами на экране и сопровождаются дополнительными полями с номерами строк и столбцов. Можно выдать на экран дисплея фрагмент или сечение массива.

— Блок Визуализации Данных в Графическом Виде.
Графическая визуализация данных облегчает понимание существа исследуемых процессов.
Графические обьекты в нашем комлексе имеют три уровня.
Верхний уровень — график (тип GRAPH): именованный обьект, содержащий набор графических примитивов следующих типов:

- TITLE
- REGION

Средний уровень (тип REGION) — область экрана для отображения данных в виде кривых и изолиний; является именованным обьектом, содержащим графические примитивы следующих типов:

- TITLE — заголовок (текст в рамке)
- AXIS — ось
- CURVE — кривая
- ISOLINES — изолинии
- FIELD — векторное поле в двухмерном пространстве
- MESH — сетка в двухмерном пространстве
- MAP — карта расчетной области в двухмерном пространстве
- 3DCURVE — траектория (трехмерная кривая)
- 3DISOLINES — изолинии на поверхности в трехмерном пространстве
- 3DMESH — сетка в трехмерном пространстве
- 3DMAP — карта расчетной области в трехмерном пространстве
- LEGMAX — легенда для отображения максимальных и минимальных значений набора кривых;
- LEGCURVE — легенда для идентификации отображаемых кривых по цветам, маркерам и типам линий;
- LEGISO — легенда для изолиний, содержащая шкалу цветов.

Нижний уровень составляют базовые объекты типов TITLE, AXIS, CURVE, ISOLINES, FIELD, MESH, MAP, 3DCURVE, 3DISOLINES, 3DMESH, 3DMAP, LEGMAX, LEGCURVE и LEGISO.
Объекты типа GRAPH и REGION являются составными, т.к. включают в себя другие объекты. Они содержат также описание области экрана для рисования включенных в них графических обьектов. Эта область (PORT) является прямоугольной и содержит такие параметры, как цвет, тип штриховки, рамку.
Список объектов, содержащихся в составном графическом объекте может модифицироваться путем добавления и удаления объектов из списка.
Все графические объекты имеют имена.

— Блок Вызова Командных Файлов.
Командные файлы состоят из последовательности инструкций командного языка и реализуют некоторую процедуру по визуализации и преобразованию данных. Общеупотребительные командные файлы хранятся в системной директории описываемой системы. Пользователь имеет возможность создать командные файлы, специфичные для его задач и хранить их в локальных директориях.

— Блок Сохранения и Восстановления Текущего Состояния.
В процессе работы с комплексом пользователь создает различные обьекты: переменные, графики и т.п. и настраивает различные параметры системы и отдельных объектов под свои задачи. Пользователю предоставляется возможность сохранить текущее состояние в файле и восстановить его по определенной команде. Состояние хранится в виде командного файла, в котором описана последовательность команд, приводящая к требуемому состоянию.

— Файлы.
Описываемая система работает с большим количеством разнотипной информации. Разумно группировать все файлы, относящиеся к каждой отдельной задаче в своей директории. Директория с командным процессором хранит только файлы, необходимые для функционирования командного процессора и общеупотребительные вычислительные модули и командные файлы — будем называть эту директорию статической. Директорию для конкретной задачи пользователя будем называть динамической или локальной.
В обоих типах директорий файлы группируются по следующим поддиректориям:

- /EXE — вычислительные модули:
  - *.EXE,*.LTL (исполняемые файлы)
  - *.ARG (файлы формальных параметров)
  - *.PAR (файлы фактических параметров)
- /CMD — командные файлы, содержащие:
  - процедуры по обработке и визуализации данных
  - состояние системы
- /SDF — двоичные файлы данных в SDF–формате
- /SDT — текстовые файлы данных:
  - SDT–файлы (аналог SDF–файла)
  - ASCII–файлы (столбцы с числами)
- /GRF — графические файлы данных
  - *.PCX (файлы в формате PCX)
  - *.PLT (файлы в формате PLT)
  - *.PS (файлы в формате PostScript)
- /SYS — поддиректория для системных файлов в статической директории

6 Организация вычислительной среды

С целью достижения максимальной переносимости в качестве языка программирования предлагается стандарт ФОРТРАН–77.

Входные и выходные данные вычислительного модуля имеют имена и размещаются в базе данных, доступ к которой осуществляется посредством специальных функций. В эту базу данных могут заносится также и локальные переменные модуля. С точки зрения структуры база данных занимает несколько больших сегментов машинной памяти — для каждого типа переменных резервируется свой массив. Эта память располагается в головной части вычислительного модуля, а ее размеры определяются исходя из конкретных вычислительных потребностей. Библиотечные функции по работе с базой данных поддерживают необходимое позиционирование в буферных массивах на данные по запрашиваемым именам.

Каждое из данных характеризуется именем, типом, размерами по каждой из трех размерностей.

Поддерживается работа со всеми основными типами данных: INTEGER*2, INTEGER*4, REAL*4, REAL*8, CHAR*8 .

Для работы с SDF–файлами используется библиотека SDFDBV.FOR .

С точки зрения внутренней организации вычислительную среду можно разбить на три основные части: препроцессор, вычислительные модули, постпроцессор. В программном комплексе RCS все эти компоненты представляются отдельными исполняемыми модулями, а связь между ними осуществляется через файлы.

На этих принципах авторами был реализован пакет, подготавливающий исходные данные для задач математической физики [5].

Остановимся подробнее на структуре вычислительного модуля.

В нашей концепции вычислительный модуль состоит из трех отдельных файлов: исполняемый программный файл, файл формальных параметров, файл соответствия формальных и фактических параметров.

Вычислительный модуль можно интерпретировать как процедуру преобразования данных, хранящуюся в отдельном файле или группе файлов.

Файл формальных параметров *.ARG описывает все параметры, задаваемые пользователем для конкретного вычислительного модуля. Этот файл создается разработчиком расчетной программы и всегда должен сопровождать ее. Вычислительный модуль "знает" только формальные параметры и по их именам осуществляет ввод необходимой для расчета информации и вывод получаемых результатов. В этом смысле имя параметра является жестко "зашитым" внутри расчетной программы.

Файл соответствия формальных и фактических параметров *.PAR задает для каждого формального параметра имя переменной в указываемом SDF–файле. Таким образом файл *.PAR отражает конкретный вариант расчета. Этот файл можно формировать как из текстового редактора, так и из командного процессора. Имя файла *.PAR должно быть известно внутри расчетной программы. Нами принято соглашение о том, что имя этого файла отличается от имени исполняемого модуля только расширением PAR.

Применение описанных двух типов файлов позволяет достичь достаточной гибкости ввода и вывода данных для расчетной программы.

В начале работы вычислительного модуля входные данные зачитываются из файлов во внутреннюю базу данных, которая находится в оперативной памяти. Затем в процессе работы происходит преобразование данных, добавление новых или уничтожение существующих данных, а также запись данных в файлы.

Как правило, в задачах эволюционного типа вычислительный модуль производит запись одного и того же набора выходных данных для некоторой последовательности моментов времени, а также некоторые интегральные величины моделируемого процесса.

7 Пример вычислительного процесса

Рассмотрим процесс организации вычислительного процесса с использованием командного процессора на примере решения задачи растекания расплавленного топлива по горизонтальному основанию [6]. Постановка задачи намеренно упрощена (простая геометриия, постоянные коэффициенты) для демонстрации использования двух модулей и нескольких файлов входных данных.

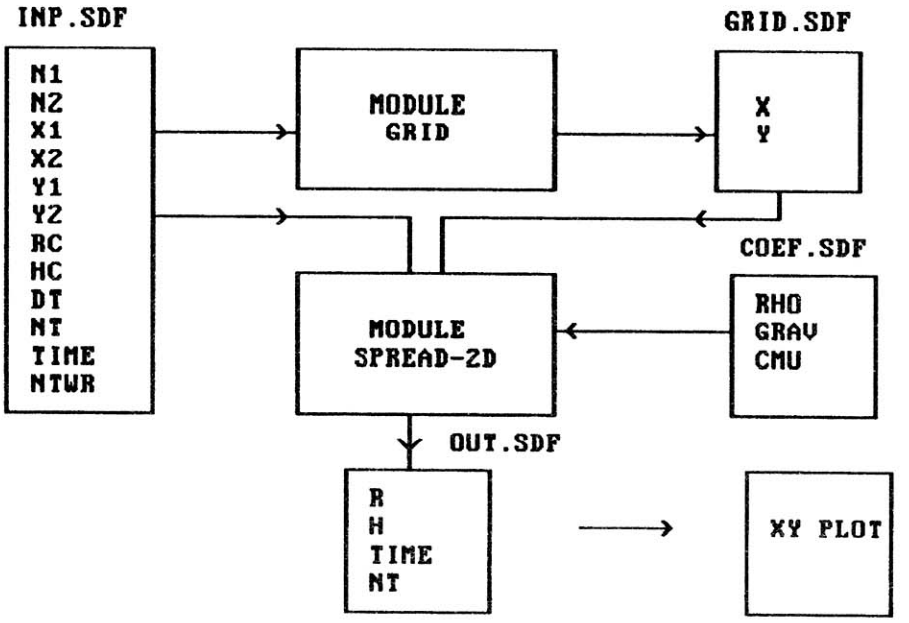

Рис.2. Потоки данных в задаче растекания топлива

В начальный момент времени жидкость занимает область цилиндрической формы. Под действием силы тяжести просходит растекание жидкости по подложке. Рассматривается течение жидкости с плотностью RHO, постоянным коэффицентом динамической вязкости CMU в поле силы тяжести с ускорением GRAV.

Вычислительный процесс разбит на два этапа и реализован в виде двух соответствующих модулей: генератор сетки GRID и основной вычислительный модуль SPREAD_2D.

Модуль GRID строит простейшую равномерную прямоугольную сетку размерностью N1*N2. Размеры прямоугольная области определяются парой точек (X1,Y1), (X2,Y2). Начальная высота жидкости — H0, начальный радиус — R0.

Модуль SPREAD_2D рассчитывает динамику течения вязкой несжимаемой жидкости по модели течения в приближении сильной вязкости.

Используются следующие данные, хранящеся на диске в SDF формате:

INP.SDF — геометрия области, начальная конфигурация, управляющие параметры;

GRID.SDF — массивы, хранящие координаты узлов расчетной сетки (X,Y);

PROP.SDF — константы и коэффициенты (плотность, ускорение силы тяжести, вязкость);

OUT.SDF — результаты расчетов (форма свободной поверхности, радиус растекания, текущее время и шаг).

Полученные в ходе расчета данные могут быть дополнительно обработаны и отображены на экран, бумагу или записаны в файл графического формата в соответствии с возможностями системы.

Рисунок 2 показывает основные блоки и файлы данных для приведенного выше примера. Рисунок 3 демонстрирует графическое представление формы поверхности на заданные моменты времени.

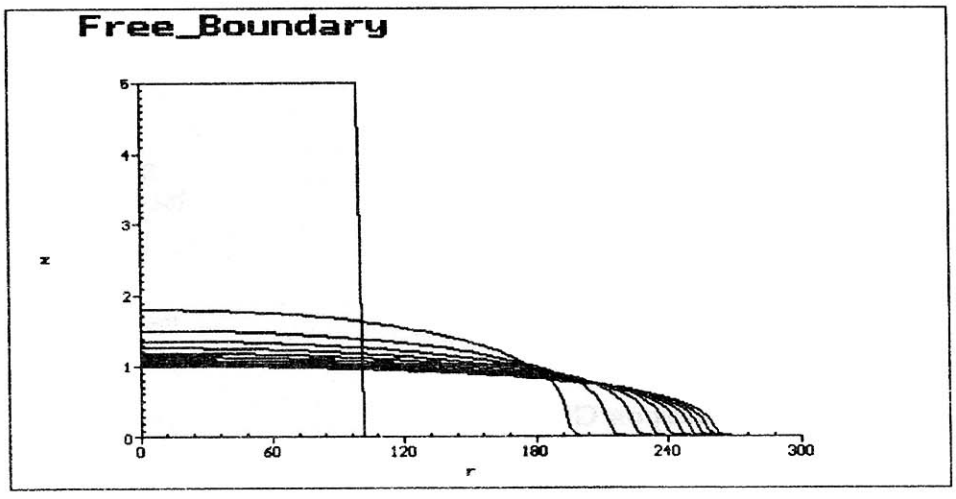

Рис.3. Графическое представление поверхности топлива

Литература

1. *Gonzalez R., Chatelard P., Jacq F.* ICARE 2 — A Computer Program for severe core damage Analysis in LWRs. France, IPSN, Cadarache, 07/05/93.

2. *Aksenova A.E., Chudanov V V., Goloviznin V.M., Pervichko V.A., Popkov A.G., Varenkov V V* Interactive Postprocessor VV–2D for scientific and engineering applications. Preprint NSI–3–94. Moscow: Nuclear Safety Institute, 1994.

3. *Варенков В.В., Первичко В.А., Попков А.Г.* Самодокументированный файл данных. Препринт NSI–16–93. М.: Институт проблем безопасного развития атомной энергетики РАН, 1993.

4. *Арутюнян Р.В., Большов Л.А., Головизнин В.М., Варенков В.В., Попков А.Г., Стрижов В.Ф., Чуданов В.В.* Комплекс программ "РАСПЛАВ" для анализа взаимодействия расплава с бетоном. Сб. Проблемы безопасного развития атомной энергетики. М.: Наука, 1993.

5. *Аксенова А.Е., Варенков В.В., Первичко В.А., Попков А.Г., Чуданов В.В.* Интерактивный пакет подготовки данных для решения задач математической физики. Препринт NSI–32–94. Москва: Институт проблем безопасного развития атомной энергетики РАН, 1994.

6. *Chudanov V.V., Popkov A.G., Strizov V.F., Vabishevich P.N., Aksenova A.E.* Modeling of core spreading processes. Preprint NSI–13–93, Moscow: Nuclear Safety Institute, 1993.

Приложение 1.
Вычислительный модуль GRID.

Файл формальных параметров GRID.ARG

```
prefix "ndprun "
    int N1_
    int N2_
    float X1_
    float X2_
    float Y1_
    float Y2_
```

Файл фактических параметров GRID.PAR

```
FILE SDF/gdata.sdf
    N1_=N1
    N2_=N2
    X1_=X1
    X2_=X2
    Y1_=Y2
    Y2_=Y2
```

Приложение 2.
Вычислительный модуль SPREAD.

Файл формальных параметров SPREAD.ARG

```
prefix "ndprun "
    int N1_
    int N2_
    float X1_
    float X2_
    float Y1_
    float Y2_
    float H0_
    float R0_
    float DT_
    int NT_
    float TIME_
    int NTWR_
    float DENS_
    float GRAV_
    float CMU_
    float X_
    float Y_
```

Файл фактических параметров SPREAD.PAR

```
FILE SDF/gdata.sdf
    N1_=N1
    N2_=N2
    XMIN_=X1
    XMAX_=X2
    YMIN_=Y1
    YMAX_=Y2
```

```
HCMAX_=H0
XCMAX_=R0
DT_=DT
NT_=NT
TIME_=TIME
NTWR_=NTWR
FILE SDF/prop.sdf
DENS_=RHO
GRAV_=GRAV
CMU_=CMU
FILE SDF/grid.sdf
X_=X
Y_=Y
```

Приложение 3.
Пример визуализации и обработки результатов.

Командный файл G.CMD

```
open fd "gdata.sdf"
   open fp "prop.sdf"
   open fg "grid.sdf"
   set file fd
   set model grid (N1_=N1, N2_=N2, X1_=X1, X2_=X2, Y1_=Y1,Y2_=Y2)
   set model spread (N1_=N1, N2_=N2, X1_=X1, X2_=X2, Y1_=Y1,
   Y2_=Y2, H0_=H0, R0_=R0, DT_=DT, NT_=NT, TIME_=TIME,
   NTWR_=NTWR, DENS_=fp:RO, GRAV_=fp:GRAV, CMU_=fp:CMU0,
   X_=fg:X, Y_=fg:H1)
   call grid
   call spread
   define limits_auto xmin=auto,xmax=auto,ymin=auto,ymax=auto
   open f "res.sdf"
   set file f
   create GRAPH ( NAME=spr1d, type=CURVE )
   set curve ( NAME=spr1d, xname=X, yname=H, limits_auto)
   set title ( NAME=spr1d, text=" Front position",color=15,b_color=0,dy=30)
   set record 1
   view graph
   set record /next
   view graph
   set record /next
   view graph
   set record /next
   view graph
```

Содержание.