НИВА - интерактивная программа для моделирования динамических систем

А.В. Воротынцев (ВЦ РАН, Москва)

Программа Нива предназначена для: 1) расчета систем взаимосвязанных динамических моделей в интерактивном режиме; 2) информационной поддержки расчетов с помощью баз данных и средств графического представления и ввода данных.

В работе описываются основные свойства Нивы на примере моделирования динамических процессов, протекающих в почве и растениях. Структура Нивы позволяет использовать ее для моделирования других сложных динамических систем.

Введение

Почва и растение - сложный комплекс взаимосвязанных биологических, физических и химических процессов, протекающих на фоне случайной метеорологической среды. Комплекс процессов выделяется большим количеством разнородных параметров, а также широким диапазоном характерных времен процессов. Достаточно отметить, что система моделей, включающая модели водного режима почвы, циклов почвенного углерода, азота, фосфора и калия, а также модель растения содержат до сотен констант и до десятков фазовых переменных с характерными временами от часа до года. Таким образом, при обычных расчетах с естественным суточным шагом модели используют числовые массивы данных объема порядка 15-20 тыс. чисел. Во многом из-за отсутствия формального описания модели в этой области оказываются сложными. Перечисленные качества объекта моделирования послужили основанием для разработки программы Нива.

Сложным моделям – абстракциям имеющихся знаний о данных и процессах, свойственны иерархичность и противоречивость представления знаний, многообразие и недостаток комплексных экспериментальных данных, необходимость экспертных оценок данных и параметров моделей предметными специалистами, в частности с помощью вариантных экспериментов с моделями.

Этим хочется отметить единство и равноправие моделей данных и моделей процессов, а также значение пользовательских интерфейсов и самих пользователей, как неотъемлемых составляющих сложной модели.

Помощь в разработке сложных моделей оказывает объектно-ориентированная методология моделирования, представляющая изучаемую область как совокупность иерархий взаимодействующих объектов. Объект инкапсулирует функции и данные, например, функции, моделирующие отдельный процесс, и данные, участвующие в процессе. Множество объектов, имеющих одинаковый набор функций и данных и отличающихся только значениями данных, описывается классом, каждый объект – это экземпляр своего класса. Классы, имеющие общие функции или данные, объединяются в иерархию классов. Классу данного уровня в иерархии доступны функции и данные базовых классов, находящихся на более низких уровнях, приближенных к корню иерархии. Класс дополняет функции и данные своих базовых классов собственными функциями и данными, которые доступны (наследуются) другими классами-наследниками, находящихся на более высоких уровнях, относительно корня иерархии. Иерархии обладают свойством полиморфизма. Оно заключается в том, что объект-наследник может изменять объекты базовых классов, подменяя их функции

собственными функциями с тем же прототипом, если в базовых классах они объявлены абстрактными (или виртуальными).

Полиморфизм объектов и наследование позволяют реализовать сложную модель с помощью иерархий абстракций. Поясним это на примере, иллюстрирующем также организанию Нивы. Пусть требуется получать из базы данных параметры для динамической модели M1, $\dot{y} = f^1(x, t)$, выполнять вариантные расчеты модели, например методом Рунге-Кутта, а результаты показывать пользователю в табличной и графической формах. Такую задачу метеодология предлагает представлять соответствующими иерархиями объектов, выполняющими извлечение данных из базы, реализацию динамической модели M1, вывод результатов расчетов в формы.

Так, реализацию модели M1 можно представить иерархией 2-х объектов некоторых классов TDynamic и TModel1. Корневой объект oDynamic1 базового класса TDynamic реализует собственно расчетный метод Рунге-Кутта, т.е. свойство «динамическая» модели M1, оперируя функциями правой части $f^1(x, t)$, объвленными абстрактными. oDynamic1 может содержать и другие абстрактные функции: load(), загружающую параметры модели; init(), начинающую расчет; output(), выводящую результаты расчетов. Объект-наследник oModel1 класса TModel1 реализует функции $f^1(x, t)$, специфичные для модели M1. В результате подмены в объекте oDynamic1 абстрактных функций на специфичные, создается иерархия 2-х объектов (oDynamic1, oModel1), реализующая конкретную модель M1.

Разработчик может предусмотреть несколько подобных иерархий (oDynamic2, oModel2), …, (oDynamicN, oModelN), реализующих другие динамические модели M2, …, MN.

Замечательное свойство базового класса, в данном случае TDynamic, описывающего общие свойства иерархий, заключается в том, что с помощью его объектов можно выполнять базовые операции, не зависящие от специфики моделей M1, …, MN, например единообразно загружать данные в модели, выполнять их расчет и вывод результатов, оперируя абстрактными функциями load(), init(), $f^1(x, t)$, output() как реализованными вне объектов. Поэтому базовый класс TDynamic называют также интерфейсом для моделей M1, …, MN.

Чтобы передать моделям параметры из баз данных, конструируется другая иерархия объектов. Ее базовые объекты (СУБД), взаимодействуя с базовыми объектами oDynamicX, выполняют стандартные операции запроса параметров из базы, их загрузку в модели, сохранение результатов в базе данных и т.д. И в этой иерархии объект-наследник через свои базовые объекты приспосабливает среду к конкретной специфике, например, реализуя правила проверки корректности параметров моделей. Набор иерархий и базовых классов можно еще дополнить и, например, дать пользователю возможность выбирать нужные модели и данные к ним, указывая мышкой на их изображения в окне программы.

Такие среды программирования как Delphi, C++ Builder, Visual C++ предоставляют стандартные библиотеки разнообразных интерфейсных классов для разработки взаимодействия с базами данных и визуализации результатов. Они позволяют разрабатывать вычислительные среды для моделирования, включающие условно: слой 1 стандартной функциональности самой среды; слой 2 функцональности для целого класса задач, решаемых вычислительной системой; слой 3, реализующий конкретную задачу пользователя системы и ее дан-

ные. Через механизм подмены абстрактных функций объекты слоя 3 приспосабливают базовые объекты слоя 1 и 2 к своим задачам, а через наследование получают их функциональность, в частности функции для обмена с базами данных, для представления данных и результатов в табличных и графических формах, а также для визуализации управления вычислениями.

Программа Нива, разработанная в среде C++ Builder, на уровне слоя 2 обеспечивает: а) формирование системы уравнений из заданного набора моделей; б) решение системы уравнений методами Рунге-Кутта и Эйлера; в) поддержку баз данных для хранения параметров моделей; г) представление результатов расчета в табличной и графической форме, д) оперативное управление и наблюдение за ходом вычислительного процесса; е) оценку параметров моделей методом наименьших квадратов; ж) управление всеми операциями визуальными средствами, стандартными для Windows. На уровне верхнего 3-го слоя Нива содержит данные и уравнения, моделирующие динамические процессы в почве и растениях.

Нива - весьма трудоемкая программа, насчитывающая в пересчете на печатные порядка 600 страниц программного кода.

Структура и интерфейсы пользователя

Обощенная схема блоков и потоков данных программы Нива представлена на рис. 1.

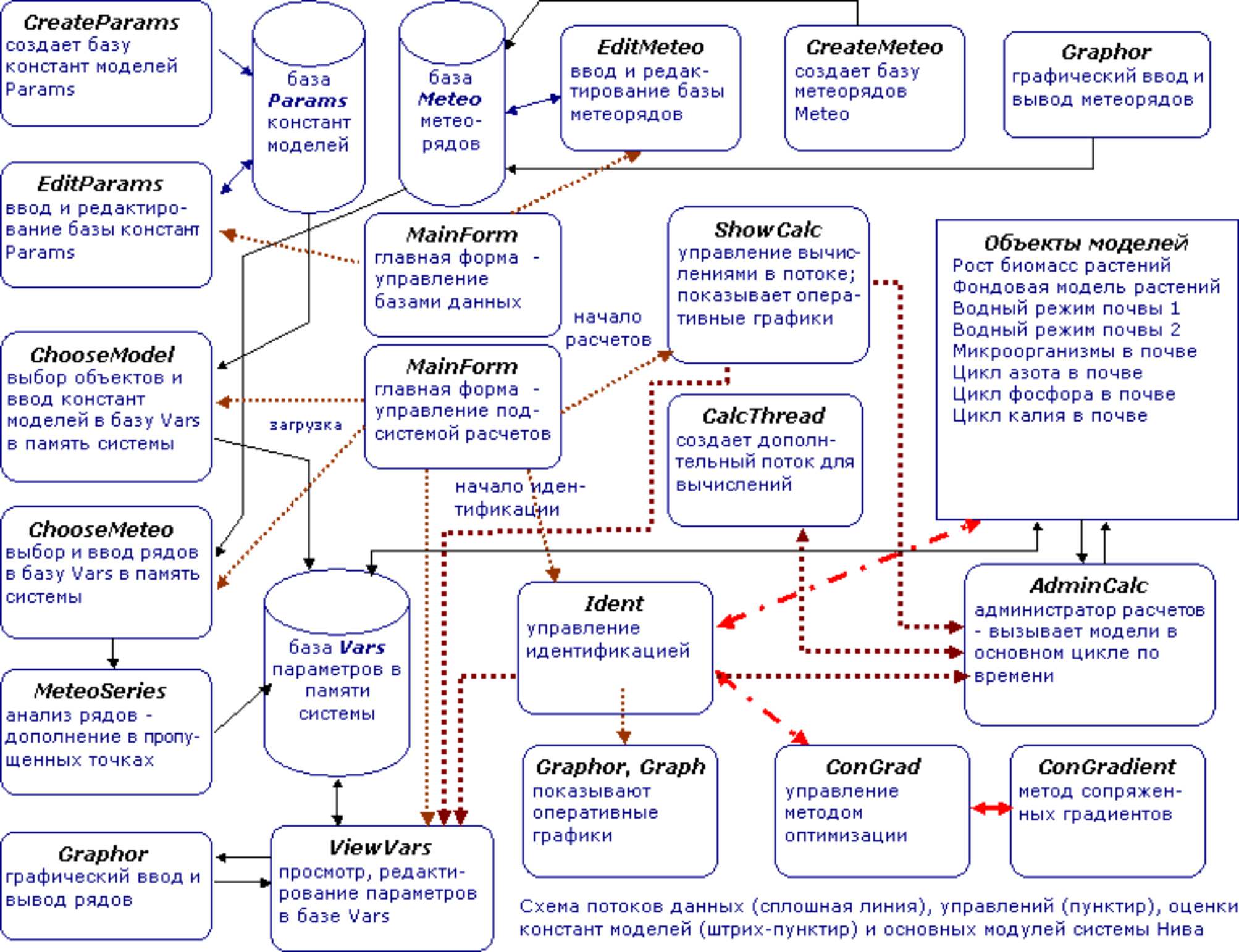

Схема потоков данных (сплошная линия), управлений (пунктир), оценки констант моделей (штрих-пунктир) и основных модулей системы Нива

Нива включает две Paradox 7 базы данных Params и Meteo для хранения описания параметров и метеорядов, используемых моделями при расчетах. Пользователь может вводить, редактировать данные с помощью форм, например формы EditParams для редактирования базы Params, приведенной на рис. 2

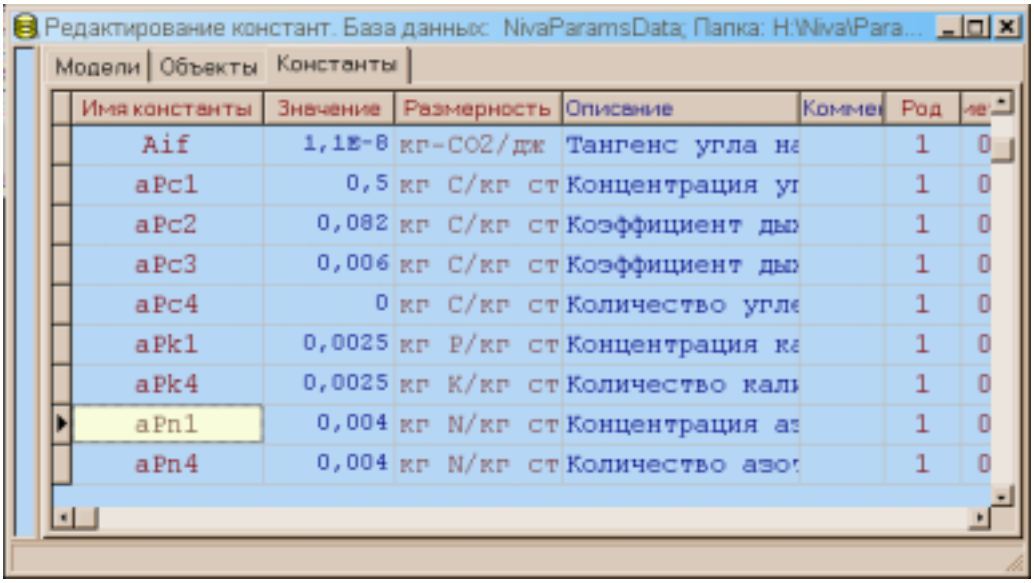

Рис. 2. Форма EditParams для редактирования база данных Params

На рис. 2 в заголовке видны три ярлычка форм "Модели", "Объекты" и "Константы", изображающих данные моделей, их объектов и параметров. Соответственно в базе Params каждый параметр принадлежит одному из объектов, а каждый объект принадлежит одной их моделей. К параметрам также отнесены переменные модели. Здесь под объектом понимается набор констант, достаточный для расчета модели. Таким образом, пользователь может выполнять вариантные расчеты модели, создавая новые объекты, принадлежащие данной модели, и наполняя их параметрами вариантов.

Помимо значения параметра, база содержит имя, размерность, а также текстовое описание и типы для каждого параметра. Особенность Нивы заключается в том, что значения констант и переменных всегда доступны пользователю по их именам. Это позволяет наблюдать и изменять их оперативно, приостанавливая исполнение программы.

При запуске Нивы вызывается главная форма MainForm (нижняя из двух на рис. 1), через меню которой выполняется вся работа с системой. С помощью формы ChooseModel (рис. 2) пользователь мышкой указывает объекты тех моделей, которые нужно рассчитывать. Далее на форме ChooseMeteo также мышкой указываются метеоданные, необходимые для расчета выбранных объектов. Нива автоматически извлекает параметры объектов и метеоданные из баз Params и Meteo и после некоторой обработки (например, в MeteoSeries) помещает их в базу Vars, которую Нива организует в памяти компьютера.

В базе Vars сохраняются также результаты расчетов по мере их выполнения. Пользователь в любое время, даже во время выполнения расчетов, может просматривать и редактировать Vars с помощью формы ViewVars (рис. 3). На дереве в левой части формы виден перечень моделей, загруженных в Vars из базы Params. На сетке справа представлены описания констант объекта "Стандартный" модели "Nitrogen", в частности значения и имена констант. Описание каждого узла дерева можно получить, щелкнув

мышкой по узлу. На дереве, кроме узла "Константы:", видны также узлы "Переменные:", "Входные переменные:", "Выходные переменные:", "Управления:" для других параметров модели Nitrogen. Одну из этих ролей пользователь должен указать для каждого параметра при его размещении в базе Params.

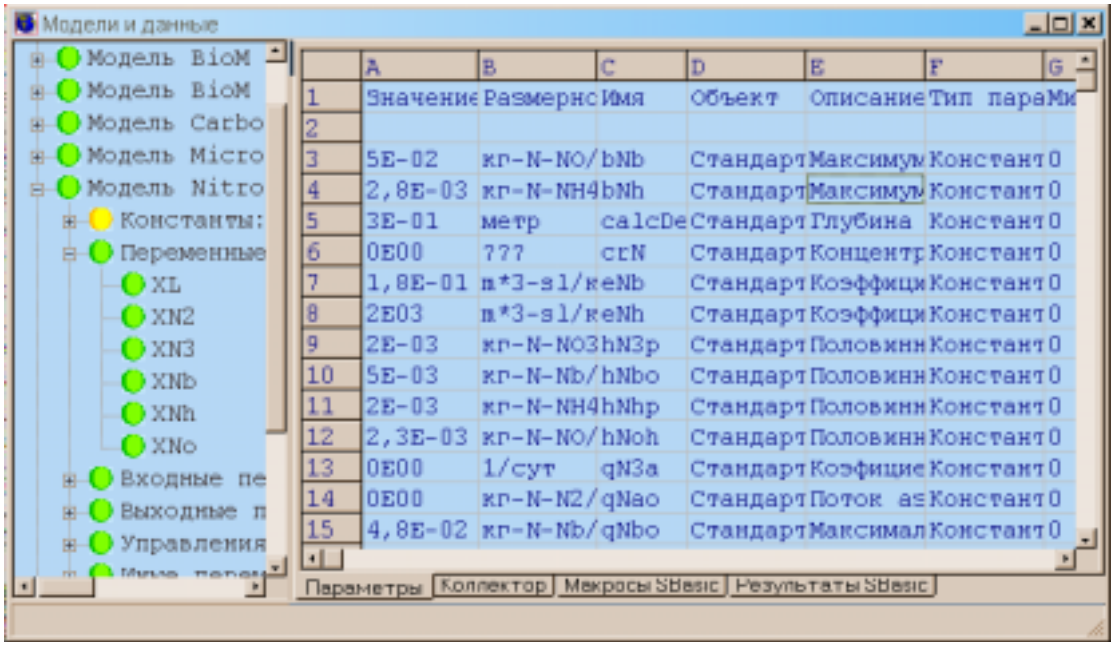

Рис. 3. Форма ViewVars показывает текущие параметры моделей

Форма ViewVars на рис. 3 позволяет вводить в модели новые значения констант и входных переменных, приостанавливая расчет, а затем продолжать расчет с новыми значениями. Мышкой на дереве можно указывать переменные, оперативные графики которых требуется наблюдать в течение расчета моделей.

После просмотра, редактирования констант и указания переменных для оперативного вывода пользователь из главного меню вызывает форму ShowCalc. Форма позволяет начинать, приостанавливать и возобновлять расчет моделей, а также наблюдать оперативные графики переменных в течение их расчета (см. рис. 3).

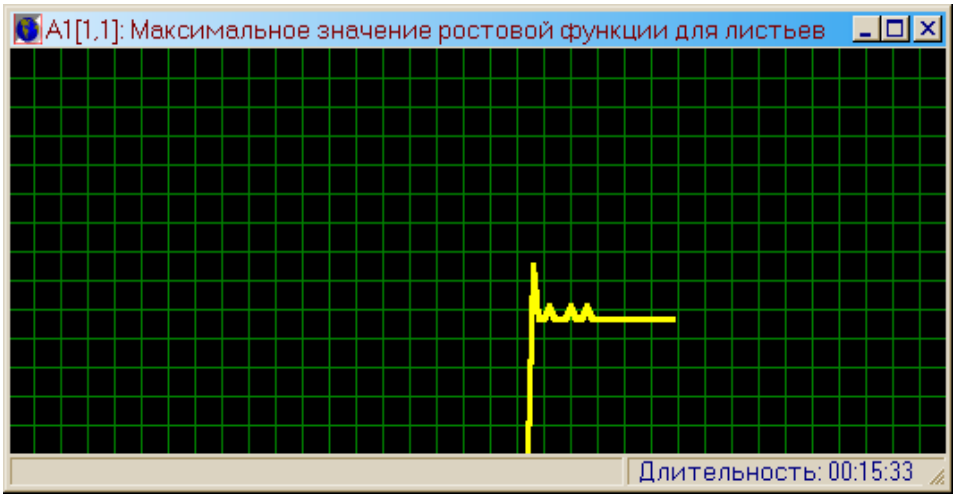

Рис. 4. Оперативный график переменной

В текущей версии Нива рассчитывает переменные выбранных моделей (точнее выбранных объектов – моделей и их данных) одним их двух мето-

дов – Рунге-Кутта или Эйлера. Метод выбирает пользователь. На каждой итерации модели получают значения своих констант и внешних переменных из базы Vars; рассчитанные новые значения переменных, в частности переменных, используемых другими моделями, помещаются в Vars перед началом следующей итерации.

Чтобы пользователь мог наблюдать ход вычислений, они в программе вынесены в дополнительных поток. Процессор компьютера с заданной частотой переключается между главным потоком, в котором исполняются команды пользователя, и дополнительным потоком, в котором исполняются вычисления. Щелчком кнопки на форме ShowCalc, пользователь из главного потока может приостановить поток вычислений; просмотрев и оценив результаты, он может изменить, например, константы моделей и продолжить вычисления.

Программа Нива позволяет оценивать параметры моделей минимизируя сумму квадратов отклонения расчетных и экспериментальных значений переменных. Пользователь мышкой на форме ViewVars (рис. 3) указывает: а) параметры, подлежащие оценке; б) оцениваемые параметры, значения которых программа будет показывать на оперативном графике в ходе их оценивания; в) переменные для сравнения с экспериментальными рядами. На рис. 4 приводится пример графика процесса оценивания одного из параметров модели.

После выполнения расчета пользователь может очистить базу Vars в памяти; обратившись к дисковым базам данных, выбрать другой набор объектов моделей и выполнить новые расчеты.

Нива легко пополняется новыми моделями, если они сводятся к системе уравнений, решаемых стандартными методами Рунге-Кутта или Эйлера. Для этого требуется присвоить имена константам, переменным новой модели, ввести их в базу Params (см. рис.2). Затем программируется и компилируется класс – наследник базового класса TModel, определяющий для упомянутых методов функции правой части `f(x, t)`.

Заключение

Описанная выше "прозрачная" организация вычислительного процесса и данных, выглядит весьма перспективной и привлекательной. В частности, такая организация допускает подключение к программе анализаторов вычислительного процесса, например, подключение экспертной системы к вычислительному процессу, обработку и отображение результатов вычислений в реальном времени; достаточно простое пополнение новыми моделями.

В своей области Нива отвечает современной тенденции управлять компьютерными задачами так, как это делает пользователь в Windows.

Конечно, такая организация вычислений требует ресурсов, несколько понижает производительность самих вычислений, частично ограничивает свободу квалифицированного исследователя. Однако, для классов задач, характерных для моделирования, такая схема, думается, увеличивает интегральную производительность в конечном счете за счет унифицированной, рациональной и наглядной организации данных и вычислений, существенно повышающей надежность моделирования. Наглядные и надежные программы – это также единственный путь довести реальные модели до потребителя.