Министерство образования Российской Федерации

Томский Государственный Университет

Факультет информатики

Кафедра теоретических основ информатики

Допустить к защите в ГАК
зав. кафедрой, к.т.н., доцент
_____ Ю. Л. Костюк
«____»_____1999г.

Сургутский Евгений Борисович

Разработка транслятора языка DYNAMO PLUS для системы имитационного моделирования DELTA

Дипломная работа

Научный руководитель
доктор технических наук
_____ В.В. Поддубный

Автор работы
студент группы 1441
_____ Е.Б. Сургутский

Томск 1999

РЕФЕРАТ

Дипломная работа 88 стр., 13 рис., 8 прил., 10 лит.

ДИНАМИЧЕСКИЕ СИСТЕМЫ, МАТЕМАТИЧЕСКИЕ МОДЕЛИ, ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ, ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, ТРАНСЛЯТОР, DYNAMO, WINDOWS, DELPHI.

Объект исследования: машинные имитационные эксперименты с математическими моделями динамических систем, языки описания математических моделей.

Цель исследования: разработка языка описания динамических моделей DYNAMO PLUS, реализация транслятора с этого языка для системы имитационного моделирования DELTA.

Методы исследования: аналитический и экспериментальный на ЭВМ.

Основные результаты: в рамках системы DELTA разработан язык DYNAMO PLUS и реализован для него транслятор.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Для исследования различных систем, экономических, социологических, технических и др., решения разного рода задач, таких как планирование, управление хозяйством, задач, возникающих в военном деле, используют, как минимум, три способа [1]. Во-первых, есть возможность, по крайней мере, теоретическая, производить эксперименты непосредственно с самой системой. Однако на практике такая возможность обычно оказывается неосуществимой, так как возникает множество проблем. Например, при изучении мировой динамики, динамики развития страны или города мы не можем вносить какие-либо изменения в реальную систему только лишь с целью эксперимента, так как система может войти в такое состояние, из которого ее уже будет невозможно вернуть в исходное.

Во-вторых, если есть данные о развитии системы за некоторый период времени, можно попытаться провести эксперимент на них, но так как эти данные были получены не в управляемом эксперименте, есть вероятность того, что различия между эндогенными (зависимыми) переменными, полученными из выборки в различные моменты времени, были вызваны некоторым случайным возмущением («шумом»). В таком случае нельзя слишком доверять решениям, полученным на основе данных о развитии системы в прошлом.

Когда же нельзя провести управляемый эксперимент и нет данных о развитии системы в прошлом (или в этих данных слишком велики случайные возмущения), остается единственная возможность: построить модель рассматриваемой системы.

Наиболее универсальной формой записи динамической модели является система дифференциальных уравнений. При этом, если модель описывается системой сравнительно небольшого

числа линейных дифференциальных уравнений первого или второго порядка, для оценки воздействия различных решений можно применить аналитические методы. Если же модель представляет собой систему большего числа нелинейных дифференциальных или разностных уравнений, имеющих высокий порядок и содержащих случайные возмущения, то аналитические методы могут оказаться практически бесполезными. В этом случае для оценки различных решений приходится обратиться к численному анализу, или имитации.

В настоящее время существует достаточно компьютерных систем для проведения имитационных экспериментов, таких как IThink компании HPS, Powersim, Vensim, GPSS и др., главным недостатком которых является их высокая стоимость. Поэтому возникает задача создания собственной системы имитационного моделирования, сочетающей в себе удобство управления, универсальность, приемлемую скорость работы, а также качественные средства анализа.

1. ПОСТАНОВКА ЗАДАЧИ

Целью дипломной работы являлось создание компьютерной системы имитационного моделирования DELTA, а точнее разработка языка описания модели для данной системы и реализация транслятора с этого языка, которые должны отвечать следующим требованиям:

- Возможность работы с языком пользователю, не владеющему навыками программирования, а также не обладающему глубокими познаниями в математике (точнее, в теории дифференциальных уравнений).
- Обеспечение максимально возможного отслеживания ошибок во время трансляции во избежание проблем на этапе моделирования.
- Удобное представление данных после этапа трансляции для дальнейшего их использования в методах численного интегрирования дифференциальных уравнений.
- Приемлемая скорость работы транслятора, а также методов интерпретации транслированных данных.

Кроме того, необходимо было создать удобный интерфейс пользователя системы DELTA, учитывающий все современные требования, предъявляемые к Windows-ориентированным системам, а также разработать удобные и исчерпывающие формы анализа, в том числе экспорт в различные системы хранения данных и их статистической обработки.

2. ТЕОРИЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

2.1. Понятие модели и классификация

Основой любого имитационного эксперимента на ЭВМ служит модель имитируемой системы, поэтому введем определение модели. В математике существует теория моделей, в которой модель определяется следующим образом.

Определение 2.1. *Модель – это произвольное множество с заданным на нем набором свойств и отношений [2].*

В естественных же науках моделями называют некоторые вспомогательные объекты исследования, применяющиеся для анализа исходных основных объектов. Таким образом, даже в круге наук, связанных с использованием математики, нельзя дать общее определение понятия модели. Выход из создавшегося положения может быть найден, если ограничиться пониманием термина «модель», которое используется в широко распространенном методе исследования, называемом моделирование.

Определение 2.2. *Моделирование – это исследование объектов познания не непосредственно, а косвенным путем, при помощи анализа некоторых других вспомогательных объектов [2].*

Такие вспомогательные объекты мы и будем называть *моделями*.

Рассмотрим основные типы моделей. Модели можно классифицировать на основе различных характеристик: по характеру моделируемых объектов, по сферам приложения, по глубине моделирования, и т.д. Поскольку нас, прежде всего, интересует роль математических моделей в исследовании социологических, экономических систем и в прогнозировании, остановимся на

классификации по характеру модели, т.е. по средствам моделирования [2]. По этому признаку методы моделирования делятся на две большие группы: материальное (предметное) моделирование и идеальное (например, мысленное, знаковое, математическое) моделирование.

Материальным называется моделирование, в котором исследование ведется на основе модели, воспроизводящей основные геометрические, физические, динамические и функциональные характеристики изучаемого объекта. Частным случаем материального моделирования является *физическое моделирование,* при котором моделируемый объект и модель имеют одну и ту же физическую природу. Наиболее известным примером использования физического моделирования является исследование моделей летательного аппарата на основе экспериментов с помощью аэродинамической трубы.

Другим частным случаем материального моделирования является *аналоговое моделирование*, основанное на аналогии явлений, имеющих различную физическую природу, но описываемых одинаковыми математическими уравнениями. Наиболее простой пример такого рода – изучение механических колебаний с помощью электрической системы, описываемой теми же дифференциальными уравнениями.

Предметное моделирование по своей природе является экспериментальным методом, так как исследование такой модели как материального объекта может состоять лишь в материальном воздействии на него.

Второй класс методов моделирования - *идеальное моделирование* – это моделирование, основывающееся на идеальной мыслимой аналогии объекта и модели. Идеальное моделирование можно разбить на два подкласса: знаковое моделирование и интуитивное моделирование.

При *знаковом* (формализованном) моделировании моделями служат знаковые образования какого-либо вида: схемы, графики, чертежи, формулы и т.д. Важнейшим видом знакового моделирования является *математическое* моделирование, осуществляемое средствами языка математики и логики.

При *интуитивном* моделировании не используют четко фиксированных знаковых систем; оно протекает, как принято говорить, «на модельном уровне». Такое моделирование часто встречается в тех областях науки, где познавательный процесс находится еще на начальной стадии.

Далее под понятием модели и метода моделирования будет подразумеваться идеальное, а точнее, математическое моделирование.

2.2. Определение имитации

Определение 2.3. *Имитация – это численный метод проведения на ЭВМ экспериментов с математическими моделями, описывающими поведение сложных систем в течение продолжительных периодов времени [1].*

В дальнейшем нас будут интересовать только динамические системы. Для полного понимания определения имитации для динамических систем нам понадобится следующий набор понятий.

Пусть Y обозначает вектор, составленный из n выходных (эндогенных) переменных системы, которые мы желаем изучить, а X – вектор, составленный из k переменных (экзогенных переменных или переменных управления). По предположению переменные X воздействуют на переменные Y в соответствии с некоторым функциональным соотношением. Динамические системы обычно описываются соотношением вида (дифференциальными уравнениями):

$$\frac{dY}{dt} = \dot{Y} = f(t, X, Y), \quad Y(0) = Y_0, \qquad (2.1)$$

где f - некоторая функция, а Y_0 - вектор начальных условий. Переменная Y, называется *реакцией системы*, переменные X_i (i=1,2,…,k) – *факторами*. Частным случаем соотношения (2.1) является простая линейная модель

$$\frac{dY}{dt} = \dot{Y} = AY + BX, \quad Y(0) = Y_0, \qquad (2.2)$$

где A-nxn матрица, B-nxk матрица параметров системы. Если бы была возможность проводить эксперименты над реальной системой, то, изменяя вектор X и наблюдая реакцию Y, можно было бы оценить параметры A и B (идентифицировать модель) и затем исследовать свойства построенной модели:

$$\frac{d\hat{Y}}{dt} = \dot{\hat{Y}} = \hat{A}Y + \hat{B}X, \quad \hat{Y}(0) = Y_0, \qquad (2.3)$$

где $\hat{A}$, $\hat{B}$ и $\hat{Y}$ обозначают оценки величин A,B и Y соответственно.

К сожалению, часто бывает нельзя или неразумно проводить управляемые эксперименты с системами. Однако иногда оказываются возможными некоторые квазиэксперименты с моделями этих систем на основе методов машинного моделирования. Пусть в нашем примере прямой эксперимент невозможен. Тогда можно имитировать реакцию, изменяя либо A и B, либо X, либо все величины одновременно, но сохраняя вид модели (2.1) или (2.2).

Линейная модель (2.2) слишком проста для изучения имитационными методами. Вероятно, ее можно было бы исследовать аналитически, не применяя численный анализ, или имитацию. Однако реальная система часто оказывается под воздействием случайных факторов. Чтобы сделать линейную модель реали-

стичнее, можно добавить случайную величину ε и переписать модель в виде

$$\frac{dY}{dt} = \dot{Y} = AY + BX + \varepsilon, \quad Y(0) = Y_0 \,. \qquad (2.4)$$

Функция плотности вероятностей вектора случайных возмущений ε задана в виде $f(\varepsilon,\mu)$, где μ - вектор параметров распределения. Модель можно сделать еще более реалистичной (и сложной), если рассматривать правую часть уравнения (2.4) как нелинейную функцию переменных t, X, Y (как в уравнении (2.1)). Правая часть может содержать дополнительные параметры.

Чтобы описать наличие или отсутствие в определенные моменты времени некоторых переменных, а также, чтобы задать блоки переменных, можно ввести логические переменные δ_{ijt} принимающие значения 0 и 1. Кроме того, на переменные и параметры модели можно наложить ограничения. Ясно, что после всего этого модель нельзя будет исследовать аналитическими методами. Поэтому для ее анализа придется обратиться к имитации.

На практике, обычно приходится иметь дело с моделями динамических систем, для которых характерны следующие черты:

1. Большое число фазовых координат Y и, может быть, вектора управления X.
2. Наличие случайных факторов (ε и др.) с их распределениями вероятностей.
3. Большое число параметров A, B, μ и др.
4. Сложная структура связей δ между элементами модели.
5. Нелинейность.
6. Ограничения различных типов.

Отсюда следует, что реализация и исследование математических моделей динамических систем возможно только с помощью ЭВМ.

2.3. Обоснования проведения имитации на ЭВМ

Для обоснования имитации на ЭВМ как средства анализа моделей систем укажем очевидное, но очень важное сходство между методом машинной имитации и некоторыми традиционными аналитическими методами, такими, как дифференциальное исчисление, математическое программирование и вариационное исчисление. Вообще говоря, одна из основных причин применения любого из этих средств анализа моделей систем состоит в поиске научных знаний о поведении данной системы. Исследование систем проходит по четырем хорошо известным этапам:

1. Наблюдение системы.
2. Формулировка математической модели, с помощью которой пытаются объяснить поведение системы.
3. Предсказание поведения системы на основании этой модели с использованием математической или логической дедукции, т.е. с помощью расчетов по модели.
4. Проведение экспериментов для проверки пригодности модели.

Машинная имитация как средство анализа систем предназначена для того, чтобы помочь осуществить перечисленные этапы исследования. Как уже говорилось выше, экспериментировать с реальными системами часто бывает невозможно, непрактично или неэкономично. Тогда на том этапе исследований, который вызывает трудность, имитация может оказаться полезной. Так, имитация позволяет провести некоторые квазиэксперименты с моделями систем.

Итак, после всего сказанного, можно сформулировать несколько доводов в пользу использования имитации для анализа динамических систем.

1. Имитация позволяет экспериментально исследовать сложные внутренние взаимодействия в рассматриваемой системе.
2. С помощью имитации можно изучать воздействие на функционирование системы различных внешних изменений; для этого в модель вносят изменения и наблюдают их влияние на поведение системы.
3. Детальное наблюдение имитируемой системы позволяет лучше понять систему и разработать такие предложения по ее улучшению, которые были бы невозможны без имитации.
4. Имитацию можно использовать как педагогический прием для обучения студентов и практиков основным навыкам теоретического анализа, статистического анализа и теории принятия решений.
5. Опыт построения имитационной модели на ЭВМ может иметь бóльшую ценность, чем имитация сама по себе. Знания, полученные во время разработки имитационной модели, часто становятся источником изменений в имитируемой системе. Влияние этих изменений можно проверить с помощью имитации еще до их практического внедрения.
6. Имитация сложных систем может дать представление о том, какие из переменных системы наиболее существенны и как эти переменные взаимодействуют.
7. Имитацию можно использовать для изучения новых ситуаций, относительно которых мало что известно или не известно ничего. Это дает возможность подготовиться к будущему.

8. Имитация может служить для предварительной проверки новых стратегий и правил принятия решений перед проведением эксперимента на реальной системе.
9. Для некоторых типов стохастических моделей особое значение может иметь последовательность событий. Данные только об ожидаемых значениях и моментах могут оказаться недостаточными для описания процесса. В этих случаях единственным удовлетворительным способом получения нужной информации может оказаться метод Монте-Карло.
10. Имитация типа Монте-Карло может служить для проверки аналитических решений.
11. Имитация позволяет изучать динамические системы в реальном или приведенном времени.
12. Имитацию можно использовать для предсказания узких мест и других трудностей, появляющихся в поведении системы при введении в нее новых элементов.

2.4. Методология машинной имитации

Под имитационным экспериментом на ЭВМ с моделями различных систем обычно понимают процедуру, состоящую из следующих пяти этапов:

1. Формулировка проблемы.
2. Формулировка математической модели.
3. Составление программы для ЭВМ.
4. Оценка пригодности модели.
5. Использование модели в имитационных экспериментах и обработка результатов исследования.

Опишем кратко каждый из этих этапов.

2.4.1. Формулировка проблемы

Как и всякое исследование, имитационные эксперименты на ЭВМ должны начинаться с формулировки проблемы, т.е. с ясного изложения целей эксперимента, поскольку эксперимент, проведенный ради самой имитации, даст мало пользы. Эти цели обычно формулируются в виде либо (1) вопросов, на которые надо ответить; либо (2) гипотез, которые надо проверить; либо (3) воздействий, которые надо оценить.

Если имитационный эксперимент проводится, чтобы получить ответы на конкретные вопросы, то ясно, что в самом начале эксперимента надо точно и детально сформулировать эти вопросы.

2.4.2. Формулировка математической модели

После того, как сформулированы цели эксперимента, надо построить математическую модель, связывающую эндогенные переменные системы с ее управляющими и экзогенными переменными. Экзогенные переменные определяются влияниями, источники которых находятся вне системы. Некоторые из них могут быть случайными, другие выражены в виде временных трендов.

Построение математической модели системы начинается с выбора переменных модели. Как правило, эндогенные переменные модели выбрать нетрудно, так как они обычно определяются уже в процессе формулировки целей исследования. Например, эндогенные переменные социально-экономической модели города могут включать доход на душу населения, занятость, уровень образования и преступности, распределения населения по возрастам и т.д. Трудности возникают при выборе входных (экзогенных и управляющих) переменных, воздействующих на выходные переменные. Если входных переменных слишком мало,

то модель может стать неадекватной реальности, если их слишком много, то из-за недостаточного объема памяти ЭВМ или сложности вычислительных процедур машинная имитация может оказаться нереализуемой. Надо строить такие математические модели, которые давали бы достаточно точное описание поведения системы и не требовали бы при этом слишком много времени на программирование и вычисления.

Далее, при выборе модели следует помнить об оценке ее адекватности. Надо определить, правильно ли описывает модель поведение системы. Пока этот вопрос не решен, ценность модели остается незначительной, а имитационный эксперимент превращается в простое упражнение в области дедуктивной логики.

Когда определена структура математической модели, описывающей поведение конкретной системы, например экономической, надо оценить величины ее параметров и проверить статистическую значимость этих оценок.

После того как на основании наблюдений реальной системы оценены параметры модели, надо решить следующие шесть вопросов, чтобы получить предварительную оценку ее адекватности:

1. Нет ли в модели несущественных переменных, которые не улучшают нашу способность предсказывать поведение эндогенных переменных системы?
2. Все ли существенные экзогенные переменные включены в модель?
3. Правильно ли сформулированы функциональные связи между входными и выходными переменными системы?
4. Верно ли оценены параметры модели и уравнений движения?
5. Являются ли оценки параметров нашей модели статистически значимыми?

В какой степени совпадают теоретические величины эндогенных переменных, полученные на основании ручного счета (машинную программу мы еще не написали), с прошлыми или фактическими значениями эндогенных переменных?

2.4.3. Составление машинной программы

Составление машинной программы начинается обычно с выбора инструментария, а точнее, языка описания имитационных моделей. В общем случае можно действовать двумя способами: написать программу на универсальном языке, таком, как ПАСКАЛЬ, СИ, ФОРТРАН, БЕЙСИК, или применить специальный имитационный язык типа GPSS, СИМСКРИПТ, ДИНАМО, СИМУЛЕЙТ.

Использование специальных языков вместо трансляторов общего назначения существенно экономит время программирования, а также избавляет пользователя от необходимости знания методов решения систем дифференциальных уравнений, которые применяются при моделировании. Специальные языки предназначены для программирования систем определенного типа. Например, язык СИМУЛЕЙТ был разработан, прежде всего, для имитации больших экономических систем, которые описываются эконометрическими моделями с большим числом уравнений. Языки GPSS и СИМСКРИПТ особенно хорошо приспособлены для исследования задач массового обслуживания (более полный обзор специализированных языков имитационного моделирования смотри в § 3.1.). Однако заметим, что, хотя использование имитационных языков сокращает время программирования, за это обычно приходится платить уменьшением гибкости модели и увеличением времени счета.

Другое важное преимущество специальных имитационных языков – наличие в них методов нахождения ошибок, значи-

тельно превосходящих соответствующие возможности универсальных языков.

2.4.4. Оценка пригодности модели

Проблема оценки пригодности имитационной модели весьма сложна, поскольку она связана со многими практическими, теоретическими, статистическими и даже философскими вопросами. Вообще говоря, для оценки пригодности имитационных моделей подходят такие два критерия: в какой степени величины эндогенных переменных, полученные в результате моделирования, совпадают с известными за прошлые периоды времени данными (если, конечно, эти данные имеются) и насколько точны предсказания имитационной модели относительно поведения реальной системы в будущем?

Какие практические вопросы возникают при оценке машинных имитационных моделей? Необходимо определить содержательные критерии, которые могли бы выделить случаи, когда траектории, полученные путем машинной имитации, в достаточной степени совпадают с наблюдаемыми траекториями, причем это совпадение не может быть просто случайным.

2.4.5. Использование модели в имитационных экспериментах и обработка результатов исследования

После того, как составлена машинная программа, можно переходить непосредственно к имитационным экспериментам. При этом можно изменять управляемые факторы модели, получая различные результаты управления.

После проведения эксперимента, полученные результаты необходимо, некоторым образом, интерпретировать. Наиболее универсальной формой представления результатов является фа-

зовая траектория, в которой результат представляется в виде графика зависимостей двух или трех переменных реакции системы.

3. ЯЗЫК DYNAMO PLUS И ЕГО ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ДЛЯ СИСТЕМЫ DELTA

3.1. Обзор существующих языков имитационного моделирования

Инструментальные средства имитационного моделирования, или, как ранее говорили, языки моделирования, появились довольно давно, почти одновременно с Алголом и Фортраном, и прошли путь от бурного развития в 70-х годах, когда они ежегодно рождались десятками, до современного стабильного состояния, когда доминирует лишь несколько языков. Наиболее широко используемые в настоящее время языки имитационного моделирования и, следовательно, инструментальные средства, их реализующие, подразделяются на три большие группы: языки имитационного моделирования непрерывных динамических систем; языки имитационного моделирования дискретных систем; универсальные языки [9].

Языки имитационного моделирования непрерывных систем предназначены для моделирования динамических объектов с непрерывным фазовым пространством и непрерывным временем. Как правило, такие объекты описываются с помощью систем дифференциальных (интегро-дифференциальных) уравнений. Системы уравнений могут быть детерминированными или стохастическими, причем в последнем случае в имитационную систему встраиваются средства статистического моделирования и обработки. Классическим языком первого типа является язык DYNAMO, разработанный Дж. Форрестером, [3,4]. Примером языка имитационного моделирования второго типа является СИМФОР, в котором к возможностям DYNAMO добавлены средства статистического моделирования и обработки. Другим примером

может служить пакет «Экспресс-Радиус», разработанный в Институте проблем управления Российской академии наук, [10].

Язык DYNAMO в тех или иных формах использовался во многих работах, но, по-видимому, наиболее впечатляющим успехом является его применение для глобального моделирования мировой экономической системы, выполненного группой японских университетов под патронажем ООН (проект «FUGI»). Подробную информацию о данной модели можно найти в Internet, по адресу HTTP://suissgate.t.soka.ac.jp/fugimodel.

К числу широко известных языков имитационного моделирования относится также язык СЛАМ, разработанный профессором Университета Пердью Аленом Прицкером в начале 70-х и с тех пор постоянно совершенствуемый созданной в 1973 году компанией Pritsker Corporation. В основе языка СЛАМ лежит простая идея - объединить достоинства DYNAMO и GPSS таким образом, чтобы, допуская раздельное применение этих языков, можно было при необходимости использовать их совместно. Реализация этого принципа на ЭВМ с цифровыми дисплеями хотя и давала некоторые преимущества при моделировании, однако не вносила качественных изменений в процесс моделирования. Переход к графическим интерфейсам раскрыл все преимущества этого принципа.

GPSS (General Purpose Simulating System) - широко известный и распространенный язык для моделирования дискретных систем, появившийся впервые еще в 1961 году, он выдержал множество модификаций для самых различных операционных систем и ЭВМ и в то же время сохранил почти неизменными внутреннюю организацию и основные блоки. Основное назначение GPSS - это моделирование систем массового обслуживания, хотя наличие дополнительных встроенных средств позволяет

моделировать и некоторые другие системы (например, распределение ресурсов между потребителями).

Надо отметить, что в последнее время интерес к GPSS стал падать, что можно объяснить двумя обстоятельствами. Во-первых, в настоящее время существенно усложнились объекты анализа: они стали в основном иерархическими, с большим количеством взаимосвязей и лучше описываются сетевыми моделями, - а надо признать, что GPSS малопригоден для моделирования сетей. Во-вторых, сейчас значительных успехов достигли языки объектно-ориентированного программирования, позволяющие строить исключительно гибкие инструментальные средства имитационного моделирования.

Особое место среди языков имитационного моделирования занимает СИМУЛА-67, разработанный в Норвежском вычислительном центре У. И. Далом, Б. Мюрхаугом и К. Нюгордом. В нем впервые получила практическое воплощение концепция ядра языка как средства иерархического, структурированного описания класса объектов, - концепция, последующее развитие которой привело к созданию объектно-ориентированного программирования. Термином «объект» в СИМУЛА были обозначены программные компоненты, обладающие собственными локальными данными (атрибутами) и способные выполнять некоторые действия. В роли атрибутов могут выступать прочие программные компоненты, переменные, массивы и ссылки на другие объекты. Действия, выполняемые объектом, задаются с помощью последовательности операторов, называемых сценарием функционирования. Каждый объект обладает системным атрибутом, указывающим на исполняемый оператор его правил действий, который называется локальным управлением. Во время работы СИМУЛА-программы могут сосуществовать несколько объектов, находящихся на разных стадиях исполнения своих сценариев функционирования, а управляющая программа передает управление от

одного объекта к другому, активизируя их по заложенному в ней сценарию.

3.2. Язык имитационного моделирования DYNAMO

3.2.1. Описание языка

После изучения теории имитационного моделирования, анализа существующих специализированных языков, а также поставленной задачи было решено в качестве языка описания моделей систем выбрать язык DYNAMO. Основными причинами этого стали следующие особенности данного языка:

1. Легкий переход от модели, построенной в виде системы дифференциальных уравнений в форме Коши, к модели на языке DYNAMO.
2. Возможность реализации на языке DYNAMO широкого спектра имитационных моделей динамических систем.
3. Возможность работы с языком пользователю, не владеющему навыками программирования, а также не обладающему глубокими познаниями в математике (точнее, в теории дифференциальных уравнений).
4. Достаточная популярность данного языка, а также наличие разработанных и проанализированных моделей на этом языке, что может обеспечить быструю отладку создаваемой системы.
5. Наличие описания языка DYNAMO, его грамматических конструкций, а также функций, применяемых в языке.

Теперь можно приступить к описанию данного языка.

Прежде всего, отметим, что при использовании языка DYNAMO исследование обычно начинается с построения концеп-

туальной диаграммы. После этого строится так называемая диаграмма потоков, в которой происходит конкретизация концептуальной модели, производится классификация переменных и связей между ними. Диаграмма потоков оказывает большую помощь при переходе от концептуальной диаграммы к программе на языке DYNAMO.

Переменные языка DYNAMO разбиваются на четыре типа: уровни, темпы, вспомогательные переменные, дополнительные переменные.

В качестве *уровней* обычно берутся те переменные, которые описывают состояние системы. Относительно таких переменных необходимо знать начальное состояние для того, чтобы было возможно промоделировать развитие системы во времени.

Скорости изменения уровней называются *темпами*. С одним уровнем может быть связано несколько темпов, причем каждый из них приводит обычно к уменьшению или к увеличению значения уровня. Изменение уровня определяется только связанными с ним темпами, сами же темпы могут зависеть от любых уровней, других темпов, а также вспомогательных переменных.

Вспомогательные переменные обычно служат для описания воздействия уровней на темпы, а также на некоторые величины, интересующие исследователя.

Дополнительные переменные служат для описания величин, которые интересуют исследователя, но не являются ни уровнями, ни темпами, ни вспомогательными переменными.

Программа на языке DYNAMO – совокупность уравнений, описывающих взаимную зависимость переменных. Для каждого из видов переменных уравнение имеет специальный вид. Для уровней оно имеет следующую форму:

$$L\ K.K = K.J + (DT)(I.JK)$$

Это уравнение означает, что величина *K* в момент *K* равна значению величины в момент *J*, предшествующий моменту *K*, сложенному с изменениями, которые внес в нее поток, управляемый величиной *I*. Здесь момент *J* соответствует моменту *t* в математической модели, момент *K* – моменту *t*+1. Поскольку величина *I* является потоком, она связана с двумя моментами времени *J* и *K*, и показывает, что под ней имеется в виду величина потока между моментами *J* и *K* (т.е. между моментами *t* и *t*+1).

Уравнения для темпов имеют следующий характерный вид:

R I.JK = (S1.J)(Y.J)

Здесь две переменные в скобках означают умножение.

Кроме того, встречаются уравнения типа *A* - для вспомогательных переменных, типа *C* – для постоянных, типа *N* – для начальных условий, типа *T* – для таблиц.

В программе также задаются три особые переменные: постоянная *DT* – шаг моделирования, постоянная *LENGTH* – период моделирования и переменная *TIME*, которая является текущим системным временем и для нее необходимо задать начальное значение.

3.2.2. Анализ грамматики языка

При анализе грамматики языка DYNAMO [4,5] было выявлено два существенных недостатка.

Во-первых, в качестве умножения используются скобки, т.е. выражение типа A*B на языке DYNAMO будет записано как (A)(B). Такая форма записи плохо читается и, кроме того, не соответствует понятиям о форме записи арифметических выражений, принятым в современных языках высокого уровня.

Во-вторых, в грамматике данного языка нет универсальной формы записи уравнений, а точнее, арифметических выражений. Все возможные типы уравнений пронумерованы и имеют строго определенный вид. При написании же программы перед уравнением необходимо ставить его номер, определенный грамматикой. Это очень неудобно для пользователя и, кроме того, усложняет грамматику.

Поэтому в качестве языка для системы DELTA было решено использовать язык DYNAMO, но с некоторыми изменениями, учитывающими современные требования предъявляемые к языкам высокого уровня, а также упрощающими процесс грамматического разбора программы. В то же время эти изменения должны быть минимальными, дабы пользователю, умеющему программировать на DYNAMO, не составило труда переписать готовые программы на новый язык.

3.3. Язык имитационного моделирования DYNAMO PLUS

Новый создаваемый язык было решено назвать DYNAMO PLUS, так как в принципе это язык DYNAMO, но все же с некоторыми изменениями.

3.3.1. Грамматика языка DYNAMO PLUS

Запишем грамматику языка в формализованном виде.

1. Основные символы. Идентификаторы

<основной символ>::= <буква> | <цифра> | <ограничитель>

1.1. Буквы

<буква>::=

A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|

a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

1.2. Цифры

<цифра>::= 0|1|2|3|4|5|6|7|8|9

1.3. Ограничители

<ограничитель>::= <знак операции> | <разделитель> | <скобка>

<знак операции>::= + | - | * | /

<разделитель>::= . | , | =

<скобка> ::= (|)

1.4. Идентификаторы

<идентификатор>::= <буква> | <идентификатор><буква> |

<идентификатор><цифра>

1.5. Числа

<число>::= <число без знака> | -< число без знака >

<число без знака>::= <десятичное число> |

<десятичное число><порядок>

<десятичное число>::= <целое без знака> | <правильная дробь> | <целое без знака><правильная дробь>

<правильная дробь>::= .<целое без знака>

<целое без знака>::= <цифра> | <целое без знака><цифра>

<порядок>::= E<целое без знака> | E+<целое без знака> |

E-<целое без знака>

2. Переменные (уровни, темпы, вспомогательные переменные, дополнительные переменные)

<переменная>::= <уровень> | <темп> |

<вспомогательная переменная> | <дополнительная переменная> | <начальное значение>

<уровень>::= <идентификатор>.<время>

<дополнительная переменная>::= <идентификатор>.<время>

<вспомогательная переменная>::= <идентификатор>.<время>

<темп>::= <идентификатор>.<интервал времени>

<время>::= J | K

<интервал времени>::= JK | KL

<начальное значение>::= <идентификатор> | <константа>

3. Уравнения

<начало уравнения> ::= <идентификатор'>.K = <идентификатор'>.J+

Символ ' указывает на то, что все идентификаторы, входящие в данную металингвистическую формулу и помеченные символом ', одинаковы.

<уравнение>::= L <начало уравнения><формула> |
<тип> <переменная> = <формула> |
C <идентификатор константы> =
<число> |
T <идентификатор таблицы> =
<список чисел>

<формула>::= <S>

<S>::= <S>+<T> | <T>

<T>::= <T>*<F> | <T>/<F> | <F>

<F>::= <A> | (<S>)

<A>::= <число> | <переменная> | <константа> | <функция>

<функция>::= EXP(<A>) |
LOGN(<A>) |

SQRT(<A>) |
SIN(<A>) |
COS(<A>) |
ABS(<A>) |
NORMRN(<A>,<A>) |
NOISE(<A>) |
PULSE(<A>,<A>,<A>) |
STEP(<A>,<A>) |
RAMP(<A>,<A>) |
SAMPLE(<A>,<A>) |
MAX(<A>,<A>) |
MIN(<A>,<A>) |
CLIP(<A>,<A>,<A>,<A>) |
SWITCH(<A>,<A>,<A>) |
TABLE(<идентификатор таблицы>,
<A>,<A>,<A>,<A>) |
TABHL(<идентификатор таблицы>,
<A>,<A>,<A>,<A>) |
DELAY(<идентификатор темпа>,<константа>)

<тип>::= A | R | S | N
<константа>::= <идентификатор константы> | <число>
<идентификатор константы>::= <идентификатор>
<идентификатор таблицы>::= <идентификатор>
<список чисел>::= <число> | <список чисел>,<число>

Кроме того, в программе могут встречаться комментарии, которые записываются в фигурных скобках.

3.3.2. Программа на языке DYNAMO PLUS

Программа на языке DYNAMO PLUS представляет из себя совокупность уравнений, отделенных друг от друга знаком «;». В языке определено семь видов уравнений:

1. Обозначается буквой L. Используется для определения переменных уровней.
2. Обозначается буквой R. Используется для определения переменных темпов.
3. Обозначается буквой A. Используется для определения вспомогательных переменных.
4. Обозначается буквой S. Используется для определения дополнительных переменных.
5. Обозначается буквой N. Используется для задания начальных значений уровней и системной переменной TIME.
6. Обозначается буквой C. Используется для задания констант.
7. Обозначается буквой T. Используется для задания таблиц.

Семантически все эти уравнения аналогичны уравнениям в языке DYNAMO.

В языке DYNAMO PLUS введено шесть специальных системных переменных: DT – константа, определяющая шаг моделирования для методов с постоянным шагом; AnalysisDT – константа, определяющая шаг, с которым будут сохраняться значения выбранных в системе переменных уровней и дополнительных переменных для дальнейшего анализа; LENGTH – константа, определяющая период интегрирования; TIME – специальная переменная уровня, которая обозначает системное время; AbsError – константа, обозначающая абсолютную погрешность для методов моделирования с переменным шагом; RelError – константа, обо-

значающая относительную погрешность для методов моделирования с переменным шагом.

Кроме того, в язык введены следующие функции:

1. EXP(X) – экспонента.
2. LOGN(X) – логарифм натуральный.
3. SQRT(X) – возведение в квадрат.
4. SIN(X) – синус.
5. COS(X) – косинус.
6. ABS(X) – абсолютное значение.
7. NOISE(X) – случайные равномерно распределенные числа на участке от 0 до X.
8. NORMRN(X,Y) – нормально распределенные числа с мат. ожиданием X и дисперсией Y.
9. MAX(X,Y) – максимум.
10. MIN(X,Y) – минимум.
11. CLIP(X,Y,U,W) – секущая функция, равна Y, если U>W и X иначе.
12. SWITCH(X,Y,U) – функция-переключатель, равна X, если U=0 и Y иначе.
13. TABLE(T,U,X,Y,W) – кусочно-линейная функция, которая строится с помощью таблицы T; X,Y – соответственно нижняя и верхняя границы изменения независимой переменной U, которая обязательно должна лежать в этих границах; W –шаг таблицы. Если U совпадает с одним из узлов таблицы, то в качестве функции берется табличное значение, иначе функция находится с помощью линейной интерполяции.
14. TABHL(T,U,X,Y,W) – то же, что и TABLE, только U может изменяться в любых пределах, а значение функции в случае выхода U за пределы X,Y будет равно одной из крайних узловых точек таблицы.

15. DELAY(X,Y) – функция задержки, используется только для темпов, равна значению темпа X в момент времени t-Y, где t – текущий момент времени.

3.4. Транслятор с языка DYNAMO PLUS

Работу транслятора можно разбить на четыре этапа:

1. Лексический анализ.
2. Синтаксический анализ.
3. Проверка ошибок в полученных уравнениях.
4. Интерпретация.

Рассмотрим каждый из этих этапов отдельно.

3.4.1. Лексический анализ

На данном этапе трансляции происходит выделение идентификаторов, констант, служебных слов, кроме того, происходит определение типа уравнения и его передача на следующий этап трансляции.

В ходе лексического анализа программа анализирует очередной символ входной строки и, в зависимости от текущего состояния, переходит в некоторое другое состояние, при необходимости вызывает процедуру (или процедуры) обработки перехода, а также записывает в строку текущего выделяемого уравнения обработанный код на внутреннем языке. Всего в анализаторе используется 24 состояния, а также 21 специализированная процедура обработки перехода. Процесс перехода из состояния в состояние происходит с использованием вспомогательной таблицы переходов, которая в данной работе не приводится в силу ее громоздкости. В этой таблице также указываются процедуры, необходимые для обработки перехода.

Из всех процедур обработки перехода можно выделить четыре основные:

1. Определено начало уравнения. Происходит обнуление строки текущего выделяемого уравнения, а также определение типа уравнения.
2. Определен конец уравнения. Если число ошибок в массиве обнаруженных ошибок равно 0, то вызывается процедура синтаксического анализа, в которую передается строка текущего выделенного уравнения на внутреннем языке.
3. Выделен идентификатор/служебное слово. Если это идентификатор, то занесение его в таблицу идентификаторов и запись в строку текущего выделяемого уравнения символа «i», за которым записывается два байта – номер идентификатора в таблице. Если это служебное слово, то определение его номера по внутренней таблице служебных слов и запись в строку текущего выделяемого уравнения символа «k», за которым записывается два байта – номер служебного слова в таблице.
4. Выделено число. Занесение этого числа в таблицу констант и запись в строку текущего выделяемого уравнения символа «c», за которым записывается два байта – номер числа в таблице.

В ходе лексического анализа могут обнаруживаться ошибки, при этом анализатор переходит в наиболее подходящее состояние, которое определяется в зависимости от предыдущего состояния. При обнаружении ошибки, она заносится в массив текущих ошибок, но анализ продолжается дальше, только для последующих выделяемых уравнений не вызывается процедура синтаксического анализа.

3.4.2. Синтаксический анализ

После анализа существующих методов синтаксического анализа было решено использовать алгоритм грамматического разбора для грамматики с операторным предшествованием [6]. Не вдаваясь в детальное рассмотрение грамматик и их особенностей, о грамматике операторного предшествования можно сказать следующее. Вообще, *операторной грамматикой* называется грамматика, среди правил подстановок которой нет правил вида

$$C \to \alpha A B \beta$$

где A, B – нетерминальные символы и α, β - строки символов, каждая из которых может быть пустой строкой. Для операторных грамматик также вводятся отношения предшествования $\doteq, \cdot>, <\cdot$, которые определяются следующим образом:

1. $a \doteq b$, если есть правило подстановки

$$C \to \alpha a b \beta$$

или

$$C \to \alpha a A b \beta$$

где A – нетерминальный символ, a, b – терминальные символы.

2. $a \cdot > b$, если есть правило подстановки

$$C \to \alpha A b \beta$$

и A порождает φa или $\varphi a D$, где φ - некоторая строка, а D – нетерминальный символ.

3. $a <\cdot b$, если есть правило подстановки

$$C \to \alpha a A \beta$$

и A порождает $b\varphi$ или $Db\varphi$, где φ - некоторая строка, а D – нетерминальный символ.

Грамматикой с операторным предшествованием называется операторная грамматика, в которой для любой пары терминаль-

ных символов отношение единственно. Эти отношения называются *отношениями предшествования* терминальных символов.

Алгоритм грамматического разбора с операторным предшествованием был выбран в силу его простоты и высокой эффективности (его трудоемкость T=O(2n), где n – число анализируемых символов в строке).

На вход алгоритма синтаксического анализа поступает уравнение, записанное на промежуточном языке, а также переменная, определяющая его тип. Затем уравнение, записанное в виде строки символов, подается на вход алгоритма грамматического разбора, который имеет некоторые специфические особенности. Из-за жестких требований, налагаемых на программу на языке DYNAMO PLUS, не удалось построить единой системы порождающих правил и таблицы отношений предшествования, поэтому было выделено семь различных типов уравнений, для которых были построены свои порождающие правила и таблицы отношений предшествования. Для всех уравнений, кроме того, были выделены общие правила порождения. Итак, рассмотрим порождающие правила:

1. Общие правила.

S := S+T | T | S-T | -T | +T

T := T*F | T/F | F

F := I | (S) | I(Q) | V

Q := S | Q,S

I := a

V := c

где a – идентификатор или служебное слово, c – константа.

2. Правила для уравнений типа «L».

E := I=I+S

Причем это правило разбирается не «в чистом виде». При разборе уравнения анализируются символы вплоть до знака «+», затем начинает работу стандартный алгоритм разбора.

3. Правила для уравнений типа «R», «A», «S».

E := I=S

4. Правила для уравнений типа «N».

E := I=G

G := I | V

5. Правила для уравнений типа «T».

E := I=D

D := V | D,V

6. Правила для уравнений типа «C».

E := I=V

Вообще-то, все эти правила можно объединить и заменить все нетерминальные символы одним (а это можно сделать для грамматики операторного предшествования), но это ухудшит наглядность правил.

В ходе грамматического разбора формируется уравнение в виде обратной польской строки.

Как уже было сказано ранее, в алгоритме синтаксического анализа используется семь различных таблиц операторного предшествования, в данной работе эти таблицы не приводятся из-за их громоздкости.

В ходе синтаксического анализа отслеживается ряд возможных ошибок, которые нарушают соответствующие правила языка:

1. Для уравнений уровней и темпов в левой части могут находиться только идентификаторы текущего состояния (моменты времени K и KL соответственно), причем для каждой переменной должно быть одно и только одно такое уравнение (определяющее этот уровень или темп).

2. Уровень может зависеть только от самого себя в предыдущий момент времени, а также от темпов и вспомогательных переменных в предыдущий момент времени.
3. Темпы могут зависеть от уровней, других темпов и вспомогательных переменных текущего момента времени.
4. Вспомогательные переменные могут зависеть от уровней, других вспомогательных переменных и темпов текущего момента времени.
5. Дополнительные переменные могут зависеть от уровней, вспомогательных переменных и темпов текущего момента времени.
6. Для всех уровней должны быть заданы начальные условия.

Если обнаруживается какая-либо ошибка, она немедленно записывается в массив текущих ошибок, а алгоритм анализа прекращает свою работу.

На выходе алгоритма синтаксического анализа формируется набор уравнений в виде обратной польской строки, отсортированных по типам для удобства их дальнейшего использования. Эти уравнения передаются на следующие два этапа работы системы: этап дополнительной проверки ошибок и этап интерпретации.

3.4.3. Проверка ошибок в полученных уравнениях

На этом этапе происходит дополнительная проверка на наличие ошибок, которые невозможно было отследить на этапе синтаксического анализа, при этом просматриваются все уравнения, полученные на предыдущем этапе. Возможные ошибки соответствуют правилам, приведенным в § 3.4.2.

3.4.4. Интерпретация

Вообще-то, этап интерпретации в алгоритм трансляции не входит, так как уравнения, полученные на этапе синтаксического анализа и проверенные на этапе дополнительной проверки ошибок, передаются на вход алгоритма моделирования, главной составной частью которого является алгоритм решения систем дифференциальных уравнений. Но в этап интерпретации входят еще две процедуры. Во-первых, это процедура определения последовательности расчета уравнений типа «R», «A», «S» и проверка их на наличие циклов. Во-вторых, это процедура интерпретации обратной польской строки.

3.5. Система имитационного моделирования DELTA

При создании системы DELTA в качестве объекта для сравнения использовался пакет IThink компании High Performance Systems (смотри рис. 3.1.). В нашем распоряжении находилась демо-версия данного продукта. IThink использовался потому, что версия, имеющаяся в распоряжении, обладала практически всеми возможностями коммерческой версии, за исключением возможности сохранения моделей и редактирования текста программы (для создания модели в IThink используются визуальные компоненты), а также потому, что в качестве языка описания моделей здесь используется язык DYNAMO.

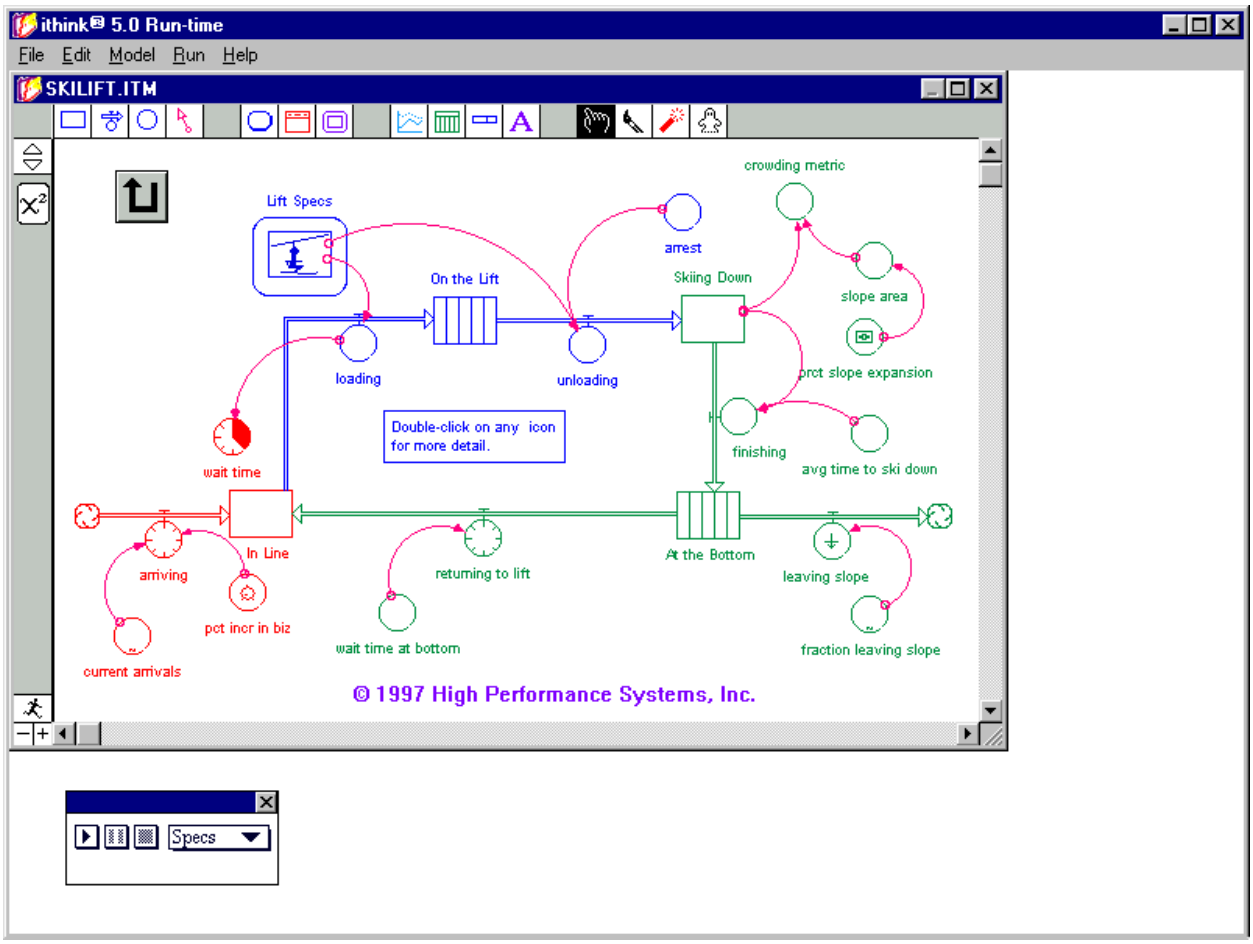


рис. 3.1 Система имитационного моделирования IThink.

Система DELTA разрабатывалась совместно автором данной работы и В.А. Соболевым.

При создании системы DELTA были выдвинуты следующие требования:

1. Удобный интерфейс пользователя, отвечающий современным требованиям, предъявляемым к программным продуктам для операционных систем Windows NT/95/98.
2. Приемлемая скорость работы программы.
3. Удобный язык описания моделей.
4. Качественные средства анализа моделей.
5. Возможность экспорта данных в наиболее популярные форматы хранения данных для возможности дальнейшей их обработки с помощью различных статистических пакетов.

В качестве средства для создания системы был выбран продукт Delphi 3.0 компании Borland (Inprise). Это было сделано по следующим причинам. Система Delphi обладает удобным интерфейсом, а также средствами, увеличивающими скорость разработки программ. В то же время система предоставляет возможность работы непосредственно с операционной системой через интерфейс Win32 API. Delphi имеет средства объектно-ориентированного программирования, что немаловажно при разработке больших проектов, а также при групповой разработке. Кроме того, система Delphi в основе своей имеет язык Паскаль, почитателем которого является автор данной работы.

Структурно система состоит из трех блоков: блок трансляции, блок моделирования и блок анализа (смотри прил. 1). Рассмотрим каждый из них.

Блок трансляции представляет собой класс Delphi, который реализует транслятор с языка DYNAMO PLUS и создает на выходе систему уравнений в виде обратной польской строки (данный блок подробно описан ранее в данной главе, а также в приложении 5). Кроме того, в данном блоке реализована процедура интерпретации обратной польской строки.

Блок моделирования входит как набор процедур в класс транслятора для удобства дальнейшего использования, но находится в отдельном модуле. В этом блоке реализованы процедуры моделирования, а также процедура построения последовательности решения системы уравнений. Данный блок полностью разрабатывался В.А. Соболевым и подробно описан в работе [8].

Блок анализа входит как набор процедур в основной класс программы. В данном блоке реализованы процедуры графического представления результатов моделирования, а также проце-

дуры экспорта результатов в различные форматы хранения статистических данных.

Интерфейсная часть системы разрабатывалась совместно автором и В.А. Соболевым и подробно описана в приложениях 5,6, а также в работе [8].

При создании системы DELTA возникла проблема ее отладки, поэтому необходимо было создать демо-программу, описывающую какую-либо модель системы. В качестве такой модели было решено выбрать модель мировой динамики, разработанной в 70-х годах Дж. Форрестером [3].

После разработки система была подвергнута сравнительным тестам на демо-программе «ШЛЯПА.dyn» (смотри прил. 7) с системой IThink. Тесты показали, что по времени моделирования система DELTA работает быстрее системы IThink, хотя система IThink обладает большей функциональностью по сравнению с системой DELTA.

4. МОДЕЛЬ МИРОВОЙ ДИНАМИКИ ФОРРЕСТЕРА

Как было сказано ранее, в качестве демо-модели для системы DELTA было решено выбрать модель мировой динамики Дж. Форрестера.

Для исследования процессов глобального масштаба необходим весьма высокий уровень абстрагирования, поэтому в модели используется всего лишь пять уровней. К уровневым переменным относятся: население, фонды, часть фондов в сельском хозяйстве, природные ресурсы, загрязнение. В этом шестимерном пространстве (уровни + время) и реализуется траектория системы - основной объект анализа.

Необходимо отметить, что целью данной работы не являлось подробное изучение мировой модели, ее структуры и динамики. Поэтому не будем здесь подробно описывать структуру модели. Тем же, кто интересуется данной проблемой, советуем ознакомиться с работой [3]. Структурная диаграмма модели приведена в приложении 2. Здесь же опишем лишь основные переменные данной модели.

BR - Birth rate; темп рождаемости (чел./год).

BRN - Birth rate normal; нормальный темп рождаемости (часть/год).

BRN1 - Birth rate normal No. 1; нормальный темп рождаемости № 1 (часть/год).

CI - Capital investment; фонды (ед. фондов).

CIAF - Capital investment agriculture fraction; часть фондов в сельском хозяйстве (безразмерная).

CIAF1 - Capital investment in agriculture fraction, initial; начальная часть фондов в сельском хозяйстве (безразмерная).

CIAFN - Capital investment in agriculture fraction normal; нормальная часть фондов в сельском хозяйстве (безразмерная).

CIAFT - Capital investment in agriculture fraction adjustment time (время задержки изменения части фондов (годы).

CID - Capital investment discard; износ фондов (ед. фондов/год).

CIDN - Capital investment discard normal; нормальный износ фондов (часть/год).

CIDN1 - Capital investment discard normal No. 1; нормальный износ фондов .№ 1.

CIG - Capital investment generation; генерация фондов (фондообразование) (ед. фондов/год).

CIGN - Capital investment generation normal; нормальное фондообразование (ед. фондов/чел. год).

CII - Capital investment, initial; начальное значение фондов (ед. фондов).

CIR - Capital investment ratio; относительная величина фондов (ед. фондов/чел.).

CIRA - Capital investment ratio in agriculture; относительная величина фондов в сельском хозяйстве (ед. фондов/чел.).

CR - Crowding ratio; относительная плотность (безразмерная).

DR - Death rate; темп смертности (чел./год).

DRN - Death rate normal; нормальный темп смертности (часть/год).

DRN1 - Death rate normal No. 1; нормальный темп смертности № 1 (часть/год).

ECIR - Effective capital investment ratio; эффективность относительной величины фондов (ед. фондов/чел.).

ECIRN - Effective capital investment ratio normal; нормальная эффективность относительной величины фондов (ед. фондов/чел.).

FC - Food coefficient; коэффициент питания (безразмерный).

FC1 - Food coefficient No 1; коэффициент питания № 1 (безразмерный).

FN - Food normal; нормальный уровень питания (ед. пищи/чел. год).

FPCI - Food potential from capital investment; пищевой потенциал фондов (ед. пищи/чел. год).

FR - Food ratio; относительный уровень питания (безразмерный).

LA - Land area; площадь земли (кв. км).

MSL - Material standard of living; материальный уровень жизни (безразмерный).

NR - Natural resources; существующие природные ресурсы (ед. природных ресурсов).

NRI - Natural resources, initial; первоначальные запасы природных ресурсов (ед. природных ресурсов).

NRUN - Natural resource usage normal; нормальное потребление природных ресурсов (ед. природных ресурсов/чел. год).

NRUN1 - Natural resource usage normal No. 1; нормальное потребление природных ресурсов № 1 (ед. природных ресурсов/чел. год).

NRUR - Natural resource usage rate; темп использования природных ресурсов (ед. природных ресурсов/год).

P - Population; население (чел.).

PDN - Population density normal; нормальная плотность населения (чел./кв.км).

P1 - Population, initial; начальное значение населения (чел.).

POL - Pollution; загрязнение (ед. загрязнения).

POLA - Pollution absorption; разложение загрязнения (ед. загрязнения/год).

POLO - Pollution generation; образование загрязнения (ед. загрязнений/год).

POLI - Pollution, initial; начальное значение загрязнения (ед. загрязнения).

POLN - Pollution normal; нормальное загрязнение (ед. загрязнения/чел. год).

POLN1 - Pollution normal No. 1; нормальное загрязнение № 1 (ед. загрязнения/чел. год).

POLR - Pollution ratio; относительное загрязнение (безразмерное).

POLS - Pollution standard; стандартное загрязнение (ед. загрязнения).

QL - Quality of life; качество жизни (ед. удовлетворенности).

QLS - Quality of life standard; стандартное качество жизни (ед. удовлетворенности).

Исходный текст программы находится в приложении 3.

В приложении 4 приведены графики зависимости переменных уровней и переменной качества жизни от времени. Эти данные были получены с помощью системы DELTA и в точности совпали с данными, представленными в работе [3], что подтверждает правильность работы системы.

ЗАКЛЮЧЕНИЕ

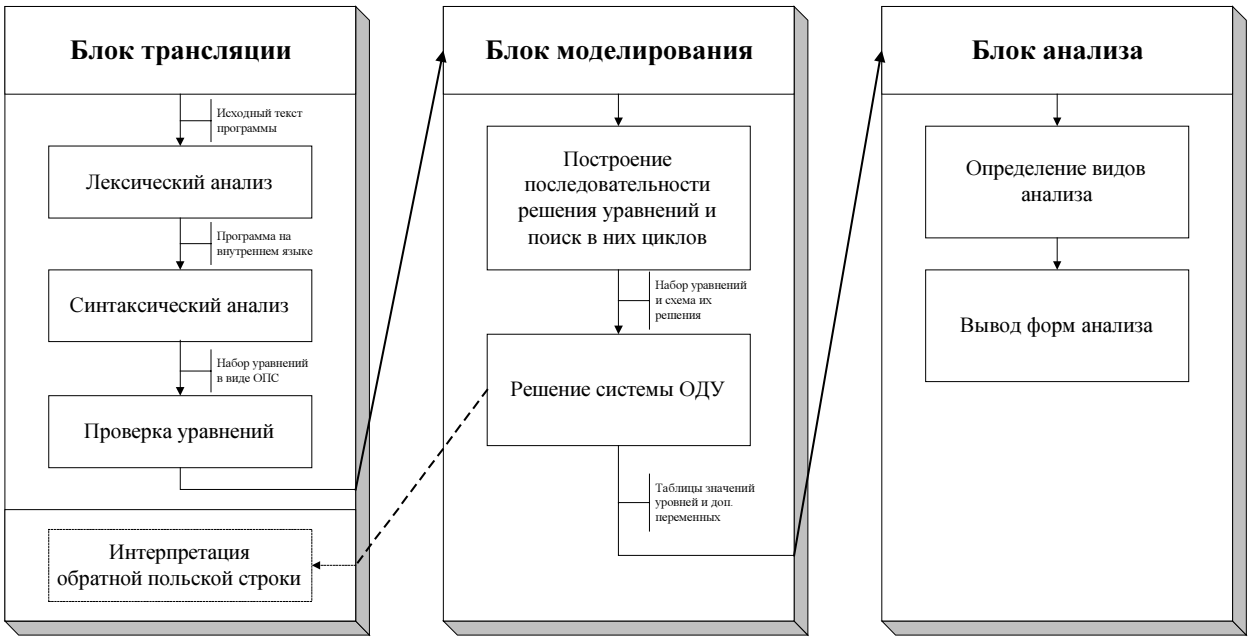
В рамках данной дипломной работы было сделано следующее:

1. Создана система имитационного моделирования DELTA.
2. Разработан язык описания моделей DYNAMO PLUS на основе известного языка DYNAMO.
3. Реализован транслятор с языка DYNAMO PLUS.
4. Разработан удобный интерфейс пользователя системы DELTA.
5. Созданы удобные средства анализа моделей в системе DELTA.
6. Работа системы была проверена с помощью программы «GlobalDynamic.dyn» (см. прил. 7), описывающей на языке DYNAMO PLUS известную модель Дж. Форрестера «Мировая динамика». Полученные при этом результаты полностью совпали с оригинальными результатами.
7. Данная система была подвергнута сравнительным тестам вместе с известной системой IThink и при равных условиях показала лучшую скорость работы, что немаловажно для систем имитационного моделирования.

СПИСОК ЛИТЕРАТУРЫ

1. Нейлор Т. Машинные имитационные эксперименты с моделями экономических систем. - М.: Мир, 1975. – 502 с.
2. Иванилов Ю.П., Лотов А.В. Математические модели в экономике. - М.: Наука, 1979. – 304 с.
3. Форрестер Дж. Мировая динамика. - М.: Наука, 1978. - 168 с.
4. Форрестер Дж. Динамика развития города. - М.: Прогресс, 1974. – 288 с.
5. Беркович Р.Н., Корявов П.П., Павловский Ю.Н., Сушков Б.Г. Динамо – язык математического моделирования (формальное описание). - М.: Вычислительный центр АН СССР, 1972. - 30 с.
6. Хопгуд Ф. Методы компиляции. - М.: Мир, 1972, - 160 с.
7. Калверт Ч. Delphi 2. Энциклопедия пользователя. - К.: НИПФ «ДиаСофтЛтд», 1996. - 736 с.
8. Соболев В.А. Методы численного дифференцирования в разработке системы имитационного моделирования DELTA: Дипломная работа. - Томск: Томский Государственный Университет, Факультет информатики, 1999. - 89 с.
9. Бахвалов Л.А. Компьютерное моделировавние: долгий путь к сияющим вершинам? // Компьютерра. - 1997. - №40.
10. Дорри М.Х., Рощин А.А. Инструментальные средства «Экспресс-радиус» для автоматизации динамических расчетов систем управления // Приборы и системы управления. - 1996. - №3.

ПРИЛОЖЕНИЕ 1. СТРУКТУРНАЯ СХЕМА РАБОТЫ СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ DELTA



ПРИЛОЖЕНИЕ 2. СХЕМА МОДЕЛИ МИРОВОЙ ДИНАМИКИ ФОРРЕСТЕРА

ПРИЛОЖЕНИЕ 3. УРАВНЕНИЯ МОДЕЛИ МИРОВОЙ ДИНАМИКИ ФОРРЕСТЕРА НА ЯЗЫКЕ DYNAMO PLUS

```
1    L P.K=P.J+(BR.JK-DR.JK);
1.1  N P=PI;
1.2  C PI=1.65E9;
2    R BR.KL=P.K*CLIP(BRN,BRN1,SWT1,TIME.K)*BRFM.K*
     *BRMM.K*BRCM.K*BRPM.K;
2.2  C BRN=.04;
2.3  C BRN1=.04;
2.4  C SWT1=1970;
3    A BRMM.K=TABHL(BRMMT,MSL.K,0,5,1);
3.1  T BRMMT=1.2,1,.85,.75,.7,.7;
4    A MSL.K=ECIR.K/ECIRN;
4.1  C ECIRN=1;
5    A ECIR.K=CIR.K*(1-CIAF.K)*NREM.K/(1-CIAFN);
6    A NREM.K=TABLE(NREMT,NRFR.K,0,1,25);
6.1  T NREMT=0,.15,.5,.85,1;
7    A NRFR.K=NR.K/NRI;
8    L NR.K=NR.J+(-NRUR.JK);
8.1  N NR=NRI;
8.2  C NRI=900E9;
9    R NRUR.KL=P.K*CLIP(NRUN,NRUN1,SWT2,TIME.K)*NRMM.K;
9.1  C NRUN=1;
9.2  C NRUN1=1;
9.3  C SWT2=1970;
10   R DR.KL=P.K*CLIP(DRN,DRN1,SWT3,TIME.K)*DRMM.K*
     *DRPM.K*DRFM.K*DRCM.K;
10.2 C DRN=.028;
10.3 C DRN1=.028;
10.4 C SWT3=1970;
```

```
11   A DRMM.K=TABHL(DRMMT,MSL.K,0,5,.5);
11.1 T DRMMT=3,1.8,1,.8,.7,.6,.53,.5,.5,.5,.5;
12   A DRPM.K=TABLE(DRPMT,POLR.K,0,60,10);
12.1 T DRPMT=.92,1.3,2,3.2,4.8,6.8,9.2;
13   A DRFM.K=TABHL(DRFMT,FR.K,0,2,.25);
13.1 T DRFMT=30,3,2,1.4,1,.7,.6,.5,.5;
14   A DRCM.K=TABLE(DRCMT,CR.K,0,5,1);
14.1 T DRCMT=.9,1,1.2,1.5,1.9,3;
15   A CR.K=P.K/(LA*PDN);
15.1 C LA=135E6;
15.2 C PDN=26.5;
16   A BRCM.K=TABLE(BRCMT,CR.K,0,5,1);
16.1 T BRCMT=1.05,1,.9,.7,.6,.55;
17   A BRFM.K=TABHL(BRFMT,FR.K,0,4,1);
17.1 T BRFMT=0,1,1.6,1.9,2;
18   A BRPM.K=TABLE(BRPMT,POLR.K,0,60,10);
18.1 T BRPMT=1.02,.9,.7,.4,.25,.15,.1;
19   A FR.K=FPCI.K*FCM.K*FPM.K*
     *CLIP(FC,FC1,SWT7,TIME.K)/FN;
19.1 C FC=1;
19.2 C FC1=1;
19.3 C FN=1;
19.4 C SWT7=1970;
20   A FCM.K=TABLE(FCMT,CR.K,0,5,1);
20.1 T FCMT=2.4,1,.6,.4,.3,.2;
21   A FPCI.K=TABHL(FPCIT,CIRA.K,0,6,1);
21.1 T FPCIT=.5,1,1.4,1.7,1.9,2.05,2.2;
22   A CIRA.K=CIR.K*CIAF.K/CIAFN;
22.1 C CIAFN=.3;
23   A CIR.K=CI.K/P.K;
24   L CI.K=CI.J+(CIG.JK-CID.JK);
24.1 N CI=CII;
```

```
24.2 C CII=.4E9;
25   R CIG.KL=P.K*CIM.K*CLIP(CIGN,CIGN1,SWT4,TIME.K);
25.1 C CIGN=.05;
25.2 C CIGN1=.05;
25.3 C SWT4=1970;
26   A CIM.K=TABHL(CIMT,MSL.K,0,5,1);
26.1 T CIMT=.1,1,1.8,2.4,2.8,3;
27   R CID.KL=CI.K*CLIP(CIDN,CIDN1,SWT5,TIME.K);
27.1 C CIDN=.025;
27.2 C CIDN1=.025;
27.3 C SWT5=1970;
28   A FPM.K=TABLE(FPMT,POLR.K,0,60,10);
28.1 T FPMT=1.02,.9,.65,.35,.2,.1,.05;
29   A POLR.K=POL.K/POLS;
29.1 C POLS=3.6E9;
30   L POL.K=POL.J+(POLG.JK-POLA.JK);
30.1 N POL=POL1;
30.2 C POL1=.2E9;
31   R POLG.KL=P.K*
     *CLIP(POLN,POLN1,SWT6,TIME.K)*POLCM.K;
31.1 C POLN=1;
31.2 C POLN1=1;
31.3 C SWT6=1970;
32   A POLCM.K=TABHL(POLCMT,CIR.K,0,5,1);
32.1 T POLCMT=.05,1,3,5.4,7.4,8;
33   R POLA.KL=POL.K/POLAT.K;
34   A POLAT.K=TABLE(POLATT,POLR.K,0,60,10);
34.1 T POLATT=.6,2.5,5,8,11.5,15.5,20;
35   L CIAF.K=CIAF.J+(CFIFR.J*CIQR.J-CIAF.J)/CIAFT;
35.1 N CIAF=CIAFI;
35.2 C CIAFI=.2;
35.3 C CIAFT=15;
```

```
36   A CFIFR.K=TABHL(CFIFRT,FR.K,0,2,5);
36.1 T CFIFRT=1,.6,.3,.15,.1;
37   S QL.K=QLS*QLM.K*QLC.K*QLP.K;
37.1 C QLS=1;
38   A QLM.K=TABHL(QLMT,MSL.K,0,5,1);
38.1 T QLMT=.2,1,1.7,2.3,2.7,2.9;
39   A QLC.K=TABLE(QLCT,CR.K,0,5,.5);
39.1 T QLCT=2,1.3,1,.75,.55,.45,.38,.3,.25,.22,.2;
40   A QLF.K=TABHL(QLFT,FR.K,0,4,1);
40.1 T QLFT=0,1,1.8,2.4,2.7;
41   A QLP.K=TABLE(QLPT,POLR.K,0,60,10);
41.1 T QLPT=1.04,.85,.6,.3,.15,.05,.02;
42   A NRMM.K=TABHL(NRMMT,MSL.K,0,10,1);
42.1 T NRMMT=0,1,1.8,2.4,2.9,3.3,3.6,3.8,3.9,3.94,4;
43   A CIQR.K=TABHL(CIQRT,QLM.K/QLF.K,0,2,5);
43.1 T CIQRT=.7,.8,1,1.5,2;
43.5 C DT=.2;
43.6 C LENGTH=200;
43.7 N TIME=1900;
44   A PRTPER.K=CLIP(PRTP1,PRTP2,PRSWT,TIME.K);
44.1 C PRTP1=0;
44.2 C PRTP2=0;
44.3 C PRSWT=0;
45   A PLTPER.K=CLIP(PLTP1,PLTP2,PLSWT,TIME.K);
45.1 C PLTP1=4;
45.2 C PLTP2=4;
45.3 C PLSWT=0;
```

ПРИЛОЖЕНИЕ 4. ГРАФИКИ УРОВНЕВЫХ ПЕРЕМЕННЫХ МИРОВОЙ МОДЕЛИ

ПРИЛОЖЕНИЕ 5. РУКОВОДСТВО ПРОГРАММИСТА

Система DELTA представлена 32 файлами, из которых 13 определяют файлы форм **Delphi** (см. [7]), 18 - модули текста на языке **Object Pascal** и один модуль – основной файл проекта **Delphi**.

DELTA.dpr - файл проекта **Delphi**.

Файлы форм (расширение **.dfm**): **Unit1**, **About**, **TransForm**, **UCommonGraphParams**, **UPhaseTrackParams**, **UParamsForm**, **VisualDataF1**, **VisualDataF2**, **UDefaultAnalysisParams**, **UDefaultParamsForm**, **UEnvironmentParams**, **UmessageBoxConfirm**, **UDBFExport**.

Кратко опишем файлы форм (они представляют лишь графическое изображение, их детальное описание смотри в приложении 6):

1. **Unit1** – основное окно программы.
2. **About** - форма, представляющая окно «О программе…».
3. **TransForm** – окно процесса трансляции.
4. **UCommonGraphParams** – окно параметров для типа анализа «Обычный график».
5. **UPhaseTrackParams** – окно параметров для типа анализа «Фазовая траектория».
6. **UParamsForm** – окно параметров модели.
7. **VisualDataF1** – окно вывода результатов анализа «Обычный график».
8. **VisualDataF2** – окно вывода результатов анализа «Фазовая траектория».
9. **UDefaultAnalysisParams** – окно параметров анализа по умолчанию.
10. **UDefaultParamsForm** – окно параметров модели по умолчанию.

11. **UEnvironmentParams** – окно параметров среды.
12. **UMessageBoxConfirm** – окно диалога подтверждения при осуществлении замены.
13. **UDBFExport –** окно экспорта данных в DBF формат.

Файлы модулей (расширение **.pas**): **About, TransForm, UCommonGraphParams, UPhaseTrackParams, UParamsForm, VisualDataF1, VisualDataF2, UDefaultAnalysisParams, UDefaultParamsForm, UEnvironmentParams, UMessageBoxConfirm, UDBFExport, Constants, NumDiff, SimThread, Translator, Types, Unit1.**

Кратко опишем файлы модулей.

1. **About** – модуль, соответствующий форме **About.**
2. **TransForm** – модуль, соответствующий форме **TransForm.**
3. **UCommonGraphParams** – модуль, соответствующий форме **UCommonGraphParams.**
4. **UPhaseTrackParams** – модуль, соответствующий форме **UPhaseTrackParams.**
5. **UParamsForm** – модуль, соответствующий форме **UParamsForm.**
6. **VisualDataF1** – модуль, соответствующий форме **VisualDataF1** (данный модуль подробно описан в работе [8]).
7. **VisualDataF2** – модуль, соответствующий форме **VisualDataF2** (данный модуль подробно описан в работе [8]).
8. **UDefaultAnalysisParams** – модуль, соответствующий форме **UDefaultAnalysisParams.**
9. **UDefaultParamsForm** – модуль, соответствующий форме **UDefaultParamsForm.**
10. **UEnvironmentParams** – модуль, соответствующий форме **UEnvironmentParams.**

11. **UMessageBoxConfirm** – модуль, соответствующий форме **UMessageBoxConfirm**.
12. **UDBFExport –** модуль, соответствующий форме **UDBFExport**.
13. **Constants** – файл констант, опишем его.

BufferLength – длина буфера, используемая при динамическом выделении памяти.

MaxVarCount – максимальное количество переменных.

MaxParams – максимальное количество параметров для функций.

XDiv, YDiv, XEndAxisDiv – смещение для осей координат в форме **VisualDataF1.**

OpersTable – таблицы отношений операторного предшествования.

ndEil – метод Эйлера.

ndRK2a – метод Рунге-Кутта 2-го порядка - вариант a.

ndRK2b – метод Рунге-Кутта 2-го порядка - вариант b.

ndRK3 – метод Рунге-Кутта 3-го порядка.

ndRK4a – метод Рунге-Кутта 4-го порядка - вариант a.

ndRK4b – метод Рунге-Кутта 4-го порядка - вариант b.

ndRK5 – метод Рунге-Кутта 5-го порядка.

ndRKM4 – метод Рунге-Кутта-Мирсона 4-го порядка.

vkL, vkR, vkA, vkS, vkB, vkC, vkT, vkDT, vkTIME, vkLENGTH – типы уравнений в трансляторе.

MaxAnalForms – максимальное количество форм анализа.

MinDouble - минимальное число типа Double.

MaxDouble - максимальное число типа Double.

Pow, A, B, E, Eps, ATolDefault, RTolDefault, BI - константы используемы в численных методах дифференцирования.

14. **Types** – файл типов.

ETooManyIdents, ETooManyConsts, EUndefinedVariable, ELeftSide – типы различных ошибок.

TIdent – идентификатор.

PIdent – указатель на идентификатор.

TTransStr – массив символов.

PTransStr – указатель на массив символов.

TArrayOfPChar - массив указателей.

PArrayOfPChar - указатель на массив указателей.

TSmallArrayOfWord – короткий массив целых чисел типа **Word**.

PSmallArrayOfWord – указатель на короткий массив целых чисел типа **Word**.

TArrayOfDouble – массив вещественных чисел.

PArrayOfDouble – указатель на массив вещественных чисел.

TArrayPassport – паспорт массива/таблицы.

PArrayPassport – указатель на паспорт массива/таблицы.

TError – ошибка.

PError – указатель на ошибку.

TVariable – переменная.

PVariable – указатель на переменную.

TLiteral – строка.

PLiteral – указатель на строку.

TEquation – уравнение.

PEquation – указатель на уравнение.

TStackItem – элемент стека синтаксического анализа.

PStackItem – указатель на элемент стека синтаксического анализа.

PBoolean – указатель на тип **Boolean**.

TOPSStackItem – элемент стека интерпретации.

POPSStackItem – указатель на элемент стека интерпретации.

TEquations – набор уравнений.

PEquations – указатель на набор уравнений.

TEquationsArray – массив наборов уравнений.

PEquationsArray – указатель на массив наборов уравнений.

TAnalysisVar – переменная для анализа.

PAnalysisVar – указатель на переменную для анализа.

TCommonAnalProps – свойства анализа типа «Обычный график».

15. **Translator** – модуль, в котором реализованы все основные алгоритмы трансляции и интерпретации.

Основным классом, описанным в модуле, является класс **TTranslator**. Ниже приводится описание основных его полей и методов:

TTranslator = class(TObject)

Errors – массив текущих ошибок.

ErrorsCount – количество ошибок в массиве.

ErrorsCapacity – количество выделенной памяти для ошибок.

ThreadFlag – флаг, определяющий выполнять ли дальше трансляцию или прервать ее.

Variables – массив переменных.

VariablesCount – количество переменных.

VariablesCapacity – количество выделенной памяти для переменных.

GVarHp,GVarTime,GVarDt,GVarLength,GVarAnalysisDT – глобальные вспомогательные переменные.

Gwhp – глобальная вспомогательная переменная.

GDrawStep - глобальная вспомогательная переменная.

procedure ChangeStrToTranslate (S: PString) – процедура изменения строки для трансляции.

procedure Translate – процедура трансляции.

procedure PrepareSimulation – процедура подготовки к процессу моделирования.

procedure Simulation – процедура моделирования.

procedure ClearErrors – процедура очистки массива текущих ошибок.

private

StrToTranslate – строка для трансляции.

Idents – массив идентификаторов.

IdentsCount – количество идентификаторов.

Consts – массив констант.

ConstsCount – количество констант.

Equations – массив уравнений.

Stack – стек.

StackCount - количество элементов стека.

StackCapacity – количество выделенной для элементов стека памяти.

OPSStack – стек обратной польской строки (ОПС).

OPSStackCount – количество элементов стека ОПС.

OPSStackCapacity – количество выделенной памяти для элементов стека ОПС.

SimSequence – массив последовательности решения уравнений.

SimSequenceCount – количество элементов массива последовательности решения уравнений.

DTVar, **TIMEVar**, **LENGTHVar**, **ABSERRORVar**, **RELERRORVar**, **ANALYSISDTVar** – индексы специальных переменных в массиве **Variables**.

RealDT – шаг моделирования.

MinDTtmp – текущий минимальный шаг моделирования.

RealDTDone – флаг, показывающий рассчитан ли шаг моделирования.

TimeForOPS – переменная, используемая в функции интерпретации ОПС вместо специальной переменной TIME.

DVT – вспомогательные таблицы для **DefineVarMethodII;**

DVTCount – количество вспомогательных таблиц.

VarDependencies – массив для **DefineVarMethodII.**

procedure AddItemToStack – добавляет **Item** в стек.

procedure ReduceStack – уменьшает стек на **Count.**

function SyntaxAnalyse – синтаксический анализ строки **Literal**, тип уравнения в строке **EquationType**, текущая строка – **Row.** Возвращает 0 в случае отсутствия ошибки, либо код ошибки.

procedure LexicalAnalyse – процедура лексического анализа текущей строки для трансляции.

procedure Initiate – инициализация процесса трансляции.

procedure DeInitiate – процедура освобождения памяти, занятой в процессе инициализации.

function CreateIdentificator – помещает строку **IdentStr** в таблицу идентификаторов, **Time** – временной тип идентификатора. Возвращает индекс идентификатора в таблице идентификаторов.

function CreateConst – помещает константу **Number** в таблицу констант.

function KeyWord – проверяет строку **IdentStr** на то, является ли она служебным словом, если да, то возвращает индекс данного ключевого слова, иначе возвращает –1.

procedure AddStringToLiteral – добавляет строку **AddingStr** к массиву символов **Literal.**

procedure AddLiteralToLiteral – добавляет массив символов **AddingLiteral** к массиву символов **Literal.**

procedure AddError – добавляет ошибку **Text** в строке **Row,** столбце **Col** к массиву текущих ошибок.

procedure CreateError – добавляет ошибку с кодом **Code,** прибавляя к ней **Text** в строке **Row,** столбце **Col** к массиву текущих ошибок.

function CreateVariable – создает переменную для идентификатора **Ident,** с временным типом **Time,** именем **Name.** Возвращает индекс переменной в массиве переменных **Variables.**

function CreateEquation – создает уравнение типа **EquationType,** для строки **Literal,** переменной **Variable.** Возвращает индекс в массиве переменных **Equations.**

function CharToTerm - преобразует **Ch** в терминальный символ.

procedure CheckErrors – проверка на наличие ошибок.

function ExecOPS интерпретирует ОПС **OPS.** Если установлен флаг **DoSet,** присваивает получившееся значение главной переменной уравнения. Возвращает получившееся значение.

procedure PushStack – добавляет элемент **Item** в стек.

function PopStack – выталкивает элемент из стека.

procedure ClearStack – очищает стек.

function DefineVar – рекурсивная функция определения последовательности вычисления уравнений, начиная с уравнения **Eq.**

function NumericDifferentiation функция решения уравнения **OPS** (смотри [8]).

function DefineVarMethodII - функция определения последовательности вычисления уравнений;

public

constructor Create – конструктор класса.

destructor Free – деструктор класса.

end;

16. **NumDiff** – модуль, в котором реализованы алгоритмы численного дифференцирования (подробно описан в работе [8]).
17. **SimThread** – модуль, в котором описан специальный класс потока **TSimThread**, рассмотрим его.

TSimThread = class(TThread)

SimTranslator – объект класса **TTranslator.**

procedure ChangeStrToTranslate – изменяет строку для трансляции.

private

Mode – текущая операция, выполняемая потоком.

DoSuspention – флаг, определяющий выполнять ли остановку потока после выполнения им очередной операции.

Owner – форма, владелец потока.

StrToTranslate – строка для трансляции.

FSetPhase, FSetProgPosition, bPrepare – вспомогательные переменные.

procedure SetPhase – устанавливает текущую фазу работы программы.

procedure SetOkButton – изменяет значение свойства **Caption** одной из кнопок формы **TransForm.**

procedure SetProgressBarPosition – устанавливает переменные текущей позиции прогресс-бара программы.

procedure SetProgressBar2Position – устанавливает позицию прогресс-бара формы **TransForm.**

protected

procedure Execute; override; - основная функция данного класса, запускается после создания потока.

public

constructor Create – конструктор класса. **CreateSuspended** – флаг определяющий, создавать ли поток задержанным, **Parent** – компонента, создающая объект данного класса.

end;

18. **Unit1** – модуль, в котором описан основной класс интерфейсной части программы, кроме того, данный модуль соответствует форме **Unit1.** В этом модуле в основном описаны функции обработки событий, поэтому их описание здесь опускается. Заголовки данных функций генерируются **Delphi** автоматически.

ПРИЛОЖЕНИЕ 6. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Данное руководство ставит своей целью описать работу с системой имитационного моделирования DELTA. Предполагается, что пользователь знаком с основами имитационного моделирования, а также с операционными системами Windows 95/98/NT, в которых функционирует система DELTA.

6.1. Установка программы

Для установки программы DELTA на компьютер достаточно скопировать файлы, хранящиеся на дистрибутивном диске в директории SETUP на жесткий диск вашего компьютера. Для этого Вам потребуется около 800 Кб свободного дискового пространства.

Необходимым условием работы программного продукта является наличие операционной системы Windows95/98/NT.

Конфигурация компьютера для работы определяется минимально возможной конфигурацией для установки Windows95/98/NT. Однако в результате тестирования программы были выработаны следующие рекомендуемые требования:

- процессор – Intel Pentium 166 Мгц;
- оперативная память – 32 Мб;
- монитор – с установкой разрешения не менее 800x600;
- свободная дисковая память – определяется величиной модели и шагом анализа.

6.2. Начало работы

На рисунке 1. Вы можете видеть общий вид системы после ее запуска и открытии файла модели.

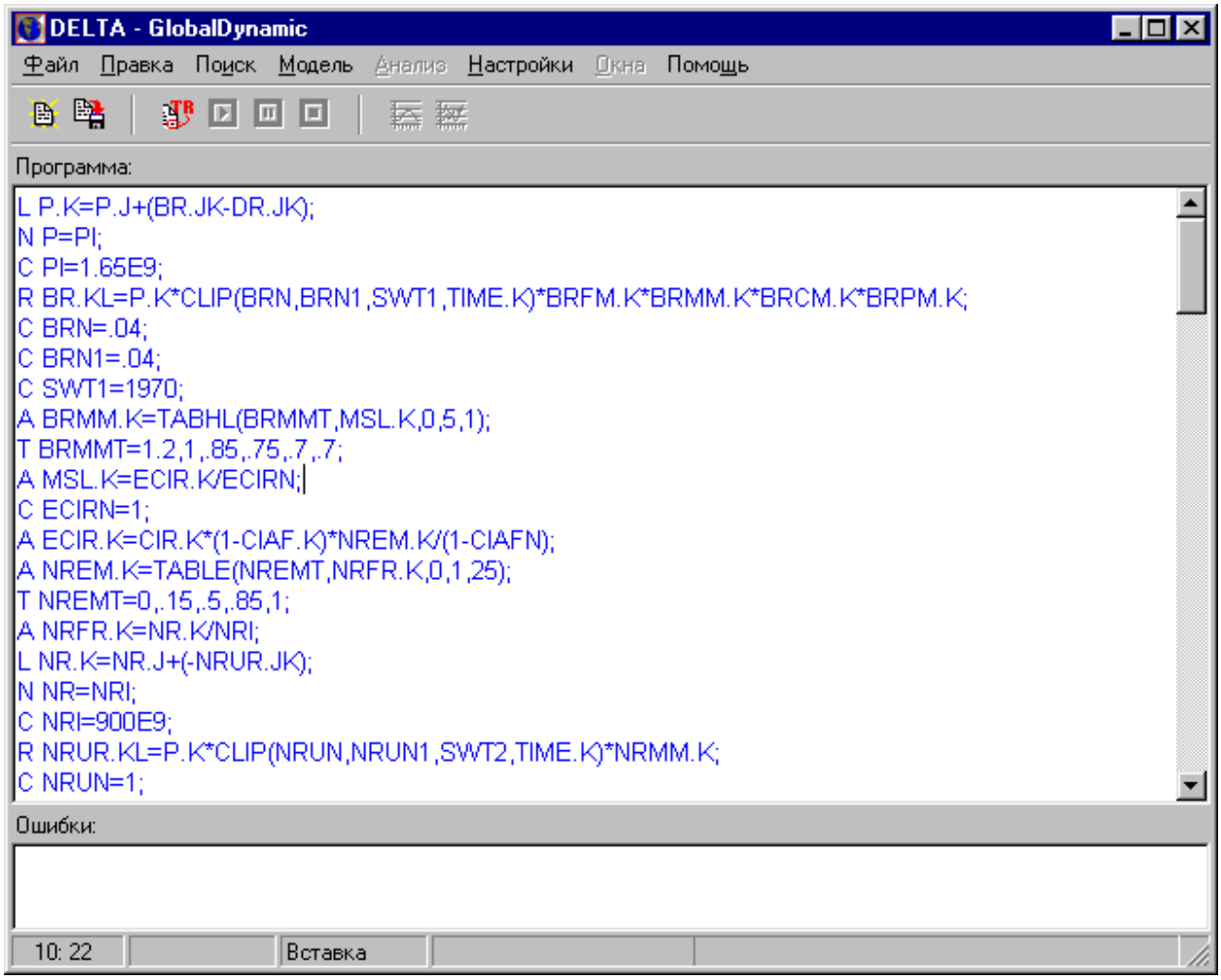


Рис. 1. Общий вид системы DELTA

Окно программы условно делится на пять частей:

- Меню программы, подробно описываемое ниже;
- Панель управления, дублирующая некоторые часто используемые команды из меню;
- Окно текста программы;
- Окно для вывода ошибок при работе с системой;

- Строка состояния, показывающая основные состояния системы в данный момент времени.

Стандартная схема последовательности шагов работы с программой DELTA представлена на рисунке 2. В блоках, очерченных прямоугольниками, приведены названия этапов работы. Пункты меню, которые необходимо использовать для осуществления текущего этапа, перечислены отдельно и связаны с соответствующим этапом работы. На любом этапе работы Вы можете выбрать пункты меню **Файл**, **Настройки** и **Помощь**.

Рассмотрим теперь более подробно всю структуру системы DELTA. Для начала разберем все пункты меню программы.

6.3. Меню Файл

Для того, чтобы начать процесс моделирования, естественно предположить, что модель предварительно должна быть формально описана. В системе DELTA средством для описания модели служит язык имитационного моделирования DYNAMO PLUS. Формальное описание модели на языке DYNAMO PLUS в дальнейшем мы будем для краткости и по аналогии с различными средами разработки, использующими понятие «язык», называть «программа».

Итак, меню **Файл** содержит следующие подпункты:

- **Новый;**
- **Открыть;**
- **Сохранить;**
- **Сохранить как...;**
- **Выход.**

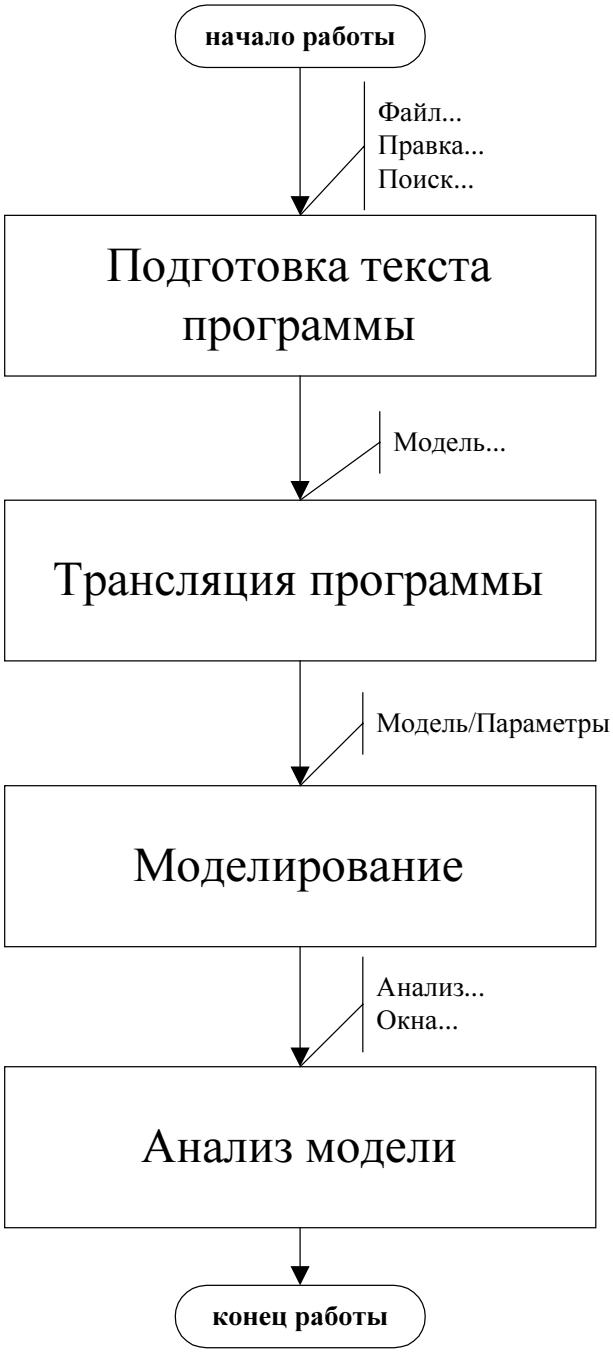


Рис. 2. Этапы работы с программой DELTA

Вы можете написать исходный текст программы полностью в системе DELTA предварительно выбрав команду **Новый**, а затем сохранить его с помощью команды **Сохранить как...** В любой момент редактирования теста программы, Вы можете быстро сохранить текст в файле под тем же именем, используя команду **Сохранить** или нажав Ctrl-S. В дальнейшем мы опустим расшиф-

ровку горячих клавиш, так как Вы всегда можете посмотреть их в меню программы. Конечно, Вы можете набрать текст программы в любом другом редакторе имеющем возможность сохранять файл в ASCII формате, и затем открыть этот файл в DELTA с помощью команды **Открыть.** Команда **Выход** завершает работу программы.

6.4. Меню Правка

Меню **Правка** содержит следующие подпункты:

- **Вырезать;**
- **Копировать;**
- **Вставить;**
- **Удалить;**
- **Выделить все.**

Описание вышеперечисленных команд опускаем, так как их функциональность вполне традиционна Windows приложениям.

6.5. Меню Поиск

Меню **Поиск** содержит следующие подпункты:

- **Найти;**
- **Заменить;**
- **Найти и далее.**

Описание вышеперечисленных команд также опускаем, так как их функциональность вполне традиционна Windows приложениям.

6.6. Меню **Модель**

Меню **Модель** содержит следующие подпункты:

- **Транслировать;**
- **Запуск;**
- **Пауза;**
- **Остановка;**
- **Параметры.**

Команда **Транслировать** запускает процесс трансляции исходного текста программы. Не вдаваясь в точное определение термина «трансляция» отметим следующие важные для пользователя действия, выполняемые при трансляции системой DELTA:

- проверка исходного текста языка на наличие в нем лексических и синтаксических ошибок;
- проверка по исходному тексту программы правильности и непротиворечивости описания структуры модели.

При трансляции никакие команды, кроме **Прервать** (трансляцию), не доступны.

На обнаруженные системой при трансляции ошибки выдаются соответствующие сообщения в окне «Ошибки», в котором двойным щелчком мыши на какой-либо ошибке курсор окна редактирования переходит в ту позицию, в которой была обнаружена ошибка. Список ошибок см. ниже.

Транслированный (без ошибок) текст программы можно запустить на выполнение собственно процесса моделирования командой **Запуск**. В любой момент моделирования Вы можете прервать этот процесс командой **Пауза** или завершить моделирование командой **Остановить**.

Необходимо помнить, что при выполнении процесса моделирования Вам недоступны следующие команды системы:

- **Модель/Трансляция**;
- **Модель/Запуск**;
- Все команды меню **Анализ**;
- Все команды меню **Окна**.

Если же Вы воспользовались командами **Пауза** или **Остановить**, то тогда недоступными остаются только все команды меню **Окна.** Соответственно в этот момент времени Вы можете, воспользовавшись командами из меню Анализ, просмотреть текущие результаты моделирования.

Последняя команда в этом меню Параметры выводит следующее окно диалога, состоящее из трех разделов, которые объединяют настройки параметров моделирования текущей, обязательно транслированной модели:

- **Общие;**
- **Абсолютные погрешности;**
- **Переменные для анализа.**

Окно диалога **Общие** параметров модели имеет следующий вид.

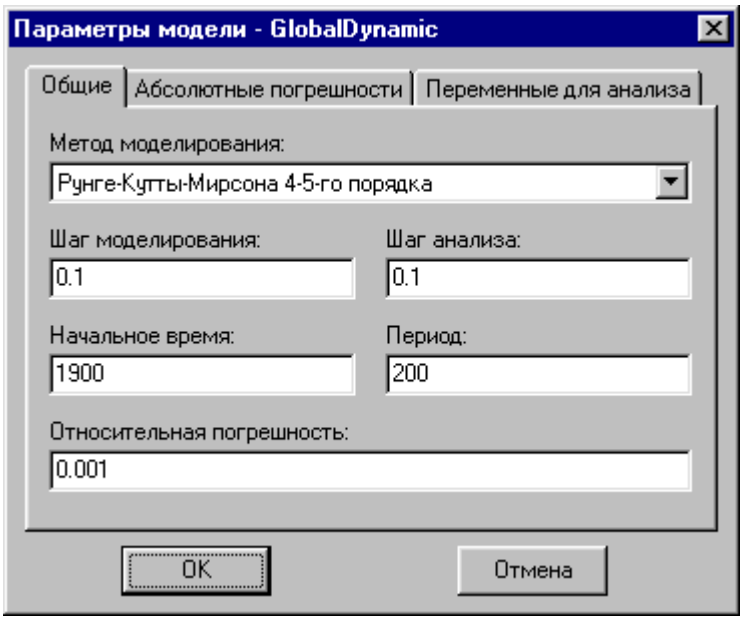


Рис. 3. Окно диалога Параметры модели/Общие

В данном окне Вы можете установить следующие параметры:

• выбрать **метод моделирования.** Более точно – метод численного решения системы обыкновенных дифференциальных уравнений (ОДУ), в терминах которых в действительности формально, т.е. математически, описывается модель. Текст программы, записанной на языке DYNAMO PLUS, на самом деле полностью соответствует некой системе ОДУ. Авторы системы DELTA советуют выбирать для моделирования метод Рунге-Кутта-Мирсона-Шампайна-Рейчелта. Необходимо отметить, что все методы используют переменный шаг моделирования (см. ниже), что существенным образом ускоряет работы программы.

• **шаг моделирования**, который используется как верхняя граница при автоматическом регулировании переменного шага моделирования алгоритмами системы. (Да простят нас уважаемые пользователи за некоторую тавтологию). Это необходимо учитывать, т.к. чем меньше шаг моделирования, тем дольше будет выполняться процесс моделирования, даже если точность вычисления (т.е. погрешности см. ниже) задана достаточно

низкой. Поэтому на больших и сложных моделях внимательно подходите к выбору этого параметра.

• **шаг анализа -** используется после моделирования при анализе. Т.е. с этим шагом по шкале времени выводятся графики анализируемых переменных или их значения экспортируются в файлы в DBF формате. **Шаг анализа** не может быть меньше **шага моделирования**.

• **начальное время**, которое присваивается начальному значению системной переменной «время».

• **период** равен временному интервалу моделирования.

• **относительная погрешность,** используется при автоматическом подборе шага моделирования. **Относительная погрешность** одинаковая для всех переменных.

Следующее окно диалога – **Абсолютные погрешности** параметров модели – имеет следующий вид.

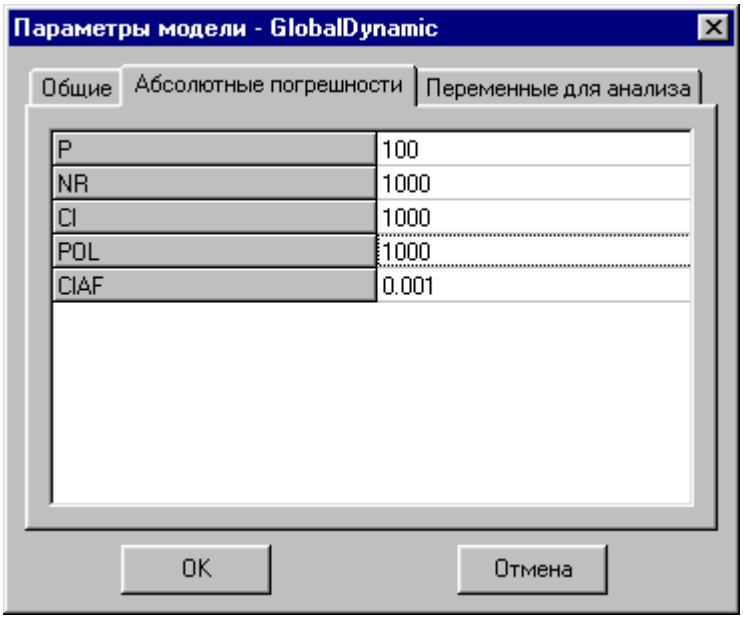


Рис. 4. Окно диалога
Параметры модели/Абсолютные погрешности

В данном окне Вы можете установить для каждой переменной абсолютную погрешность.

И последнее окно диалога – **Переменные для анализа** параметров модели - имеет следующий вид.

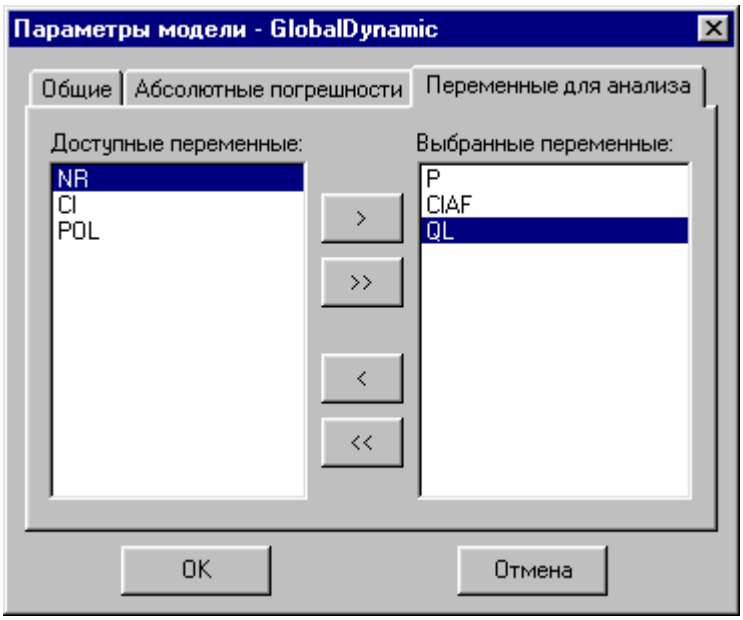


Рис. 5. Окно диалога Параметры модели/Переменные для анализа

В списке **доступных переменных** находятся все уровневые и дополнительные переменные. Из данного списка Вы можете выбрать переменные, которые потом будете использовать для анализа. Если Вы выберите слишком много переменных или шаг анализа установите слишком малым, то программа оставит список **выбранных переменных** тем же, так как система не может выделить необходимое количество памяти для хранения всех необходимых аналитических данных.

6.7. Меню Анализ

Меню **Анализ** содержит следующие подпункты:

- **Обычный график;**
- **Фазовая траектория;**
- **Экспорт в DBF файл.**

Команда **Обычный график** выводит диалоговое окно:

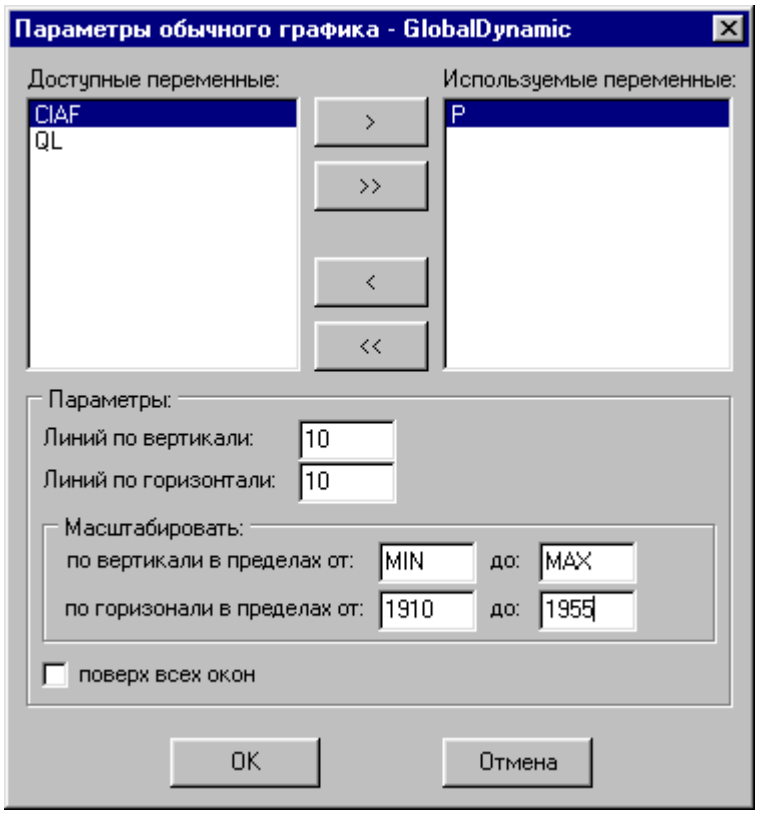


Рис. 6. Окно диалога Параметры обычного графика

В нем Вы можете выбрать переменные, для которых нужно нарисовать графики. Для более подходящего Вам отображения графиков можно изменить значения некоторых параметров, устанавливаемых системой по умолчанию:

- **линий по вертикали** – количество линий координатной сетки по оси анализируемой переменной;
- **линий по горизонтали** – количество линий координатной сетки по оси времени;
- **коэффициенты масштабирования по вертикали и горизонтали.** По умолчанию значения коэффициентов установлены таким образом, чтобы вывод графиков осуществлялся на всем диапазоне полученных в результате моделирования значений – от минимального (MIN) до максимального (MAX). Вы можете уменьшить или даже увеличить этот диапазон;

- установка параметра **поверх всех окон** – означает, что выводимое окно будет располагаться поверх всех окон.

Команда **Фазовая траектория** выводит диалоговое окно:

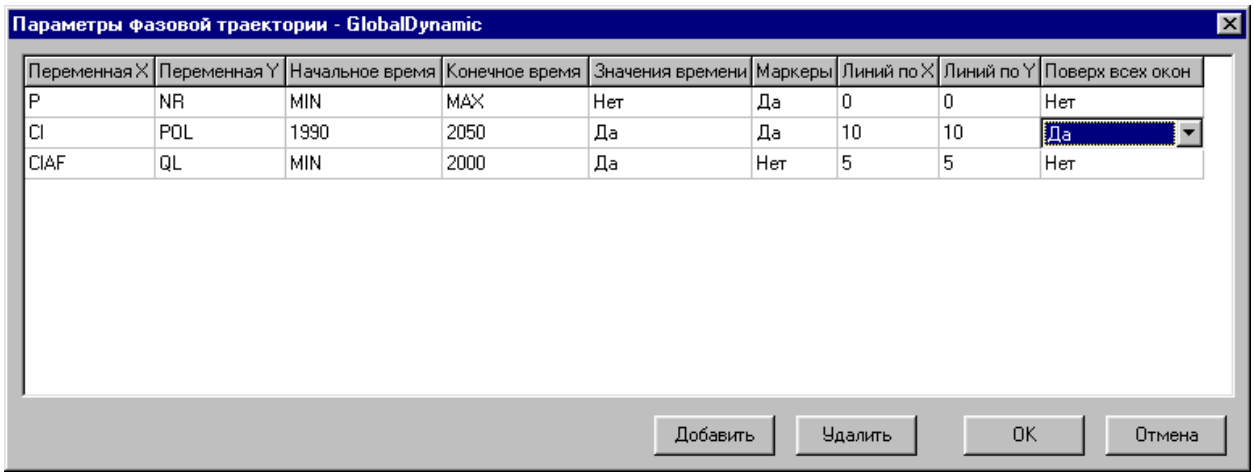


Рис. 7. Окно диалога Параметры фазовой траектории

В нем Вы можете выбрать переменные, для которых нужно нарисовать графики. Для более подходящего Вам отображения графиков можно изменить значения некоторых параметров, устанавливаемых системой по умолчанию:

- **начальное время** – начальное время вывода графика;
- **конечное время** – конечное время вывода графика;
- **значения времени** – подписывать ли значения времени на графике;
- **маркеры** – обозначать ли маркерами временные метки на графике;
- **линий по X** – количество линий координатной сетки по оси переменной X;
- **линий по Y** – количество линий координатной сетки по оси переменной Y;

- установка параметра **поверх всех окон** – означает, что выводимое окно будет располагаться поверх всех окон.

Команда **Экспорт в DBF файл** выводит диалоговое окно:

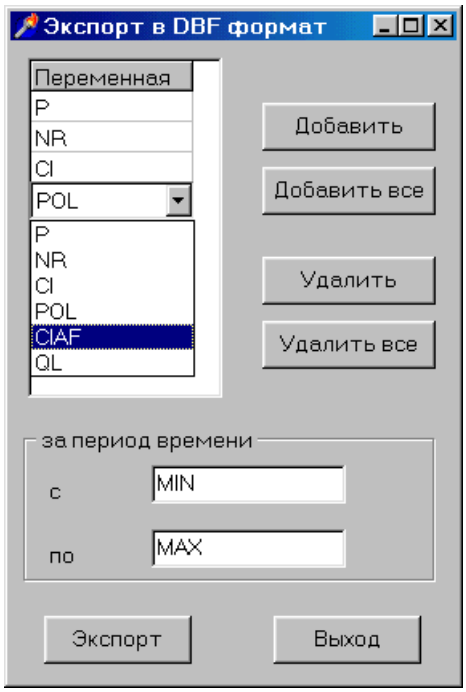


Рис. 8. Диалоговое окно экспорта в DBF формат

В нем Вы можете выбрать переменные, которые нужно записать во внешний файл. Также Вы можете указать границы временного интервала за который нужно произвести экспорт. По умолчанию осуществляется экспорт за весь период моделирования.

6.8. Меню Настройки

Меню **Настройки** содержит следующие подпункты:

- **Модель;**
- **Среда;**
- **Анализ;**

Команда **Модель** выводит диалоговое окно аналогичное **Модель/Параметры/Общие**. Параметры, задаваемые в данном окне, будут устанавливаться по умолчанию для транслированной модели.

Команда **Среда** выводит диалоговое окно, состоящее из двух разделов, которые объединяют настройки системы. В первом разделе **Общие** (рис. 9) Вы можете настроить приоритет Windows-потока, в котором выполняется моделирование.

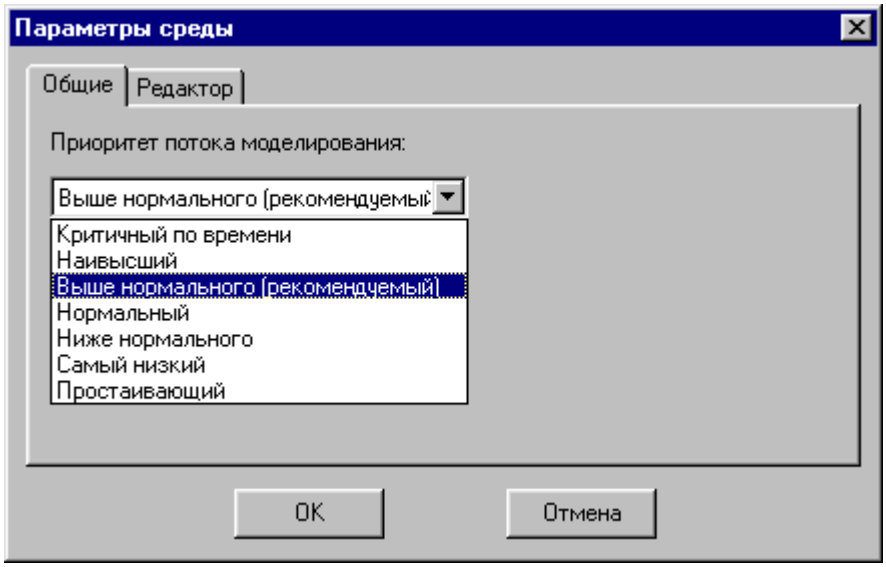


Рис. 9. Окно диалога Параметры среды/Общие

Во втором разделе **Редактор** (рис. 10) Вы можете изменить настройки шрифта, используемого в окне редактирования текста программы.

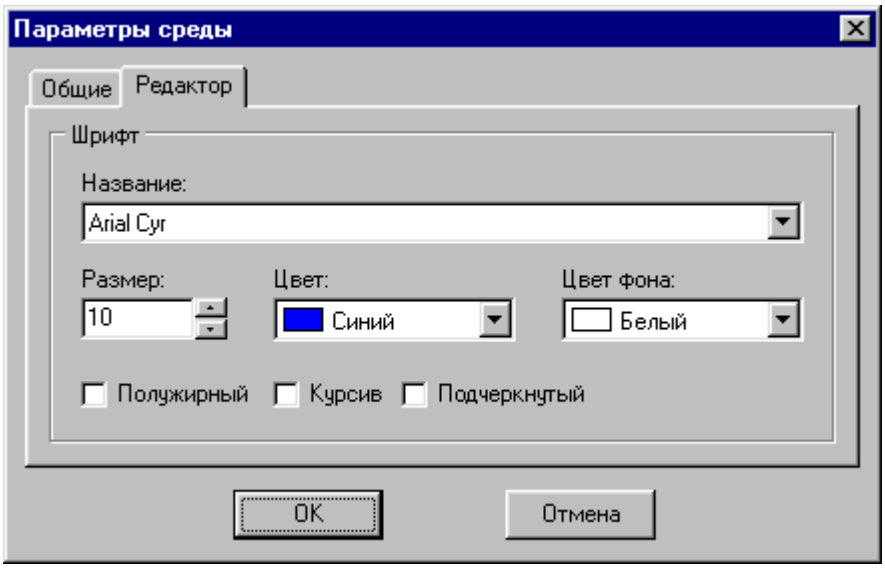


Рис. 10. Окно диалога Параметры среды/Редактор

6.9. Меню Окна

Меню **Окна** содержит следующие подпункты:

- **Выстроить все;**
- **Закрыть все;**
- **<Номер> <вид анализа> - <имя переменной>.**

Команда **Выстроить все** открывает все окна анализа.

Команда **Закрыть все** закрывает все окна анализа.

При выборе пункта меню вида **<Номер> <вид анализа> - <имя переменной>**, выводится соответствующее окно анализа. На рисунках 11, 12. приведены примеры окон анализа модели «GlobalDynamic».

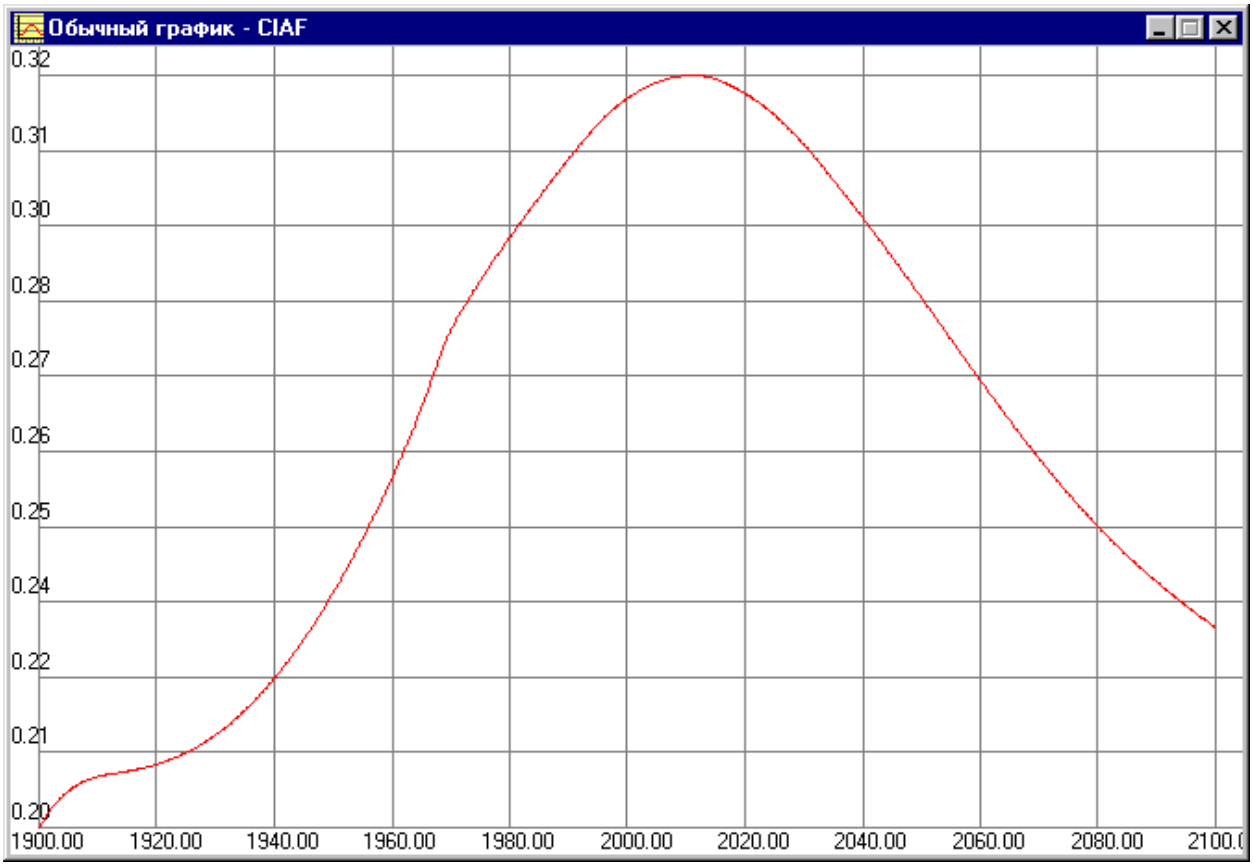


Рис. 11. Окно анализа «Обычный график» для переменной CIAF модели «GlobalDynamic»

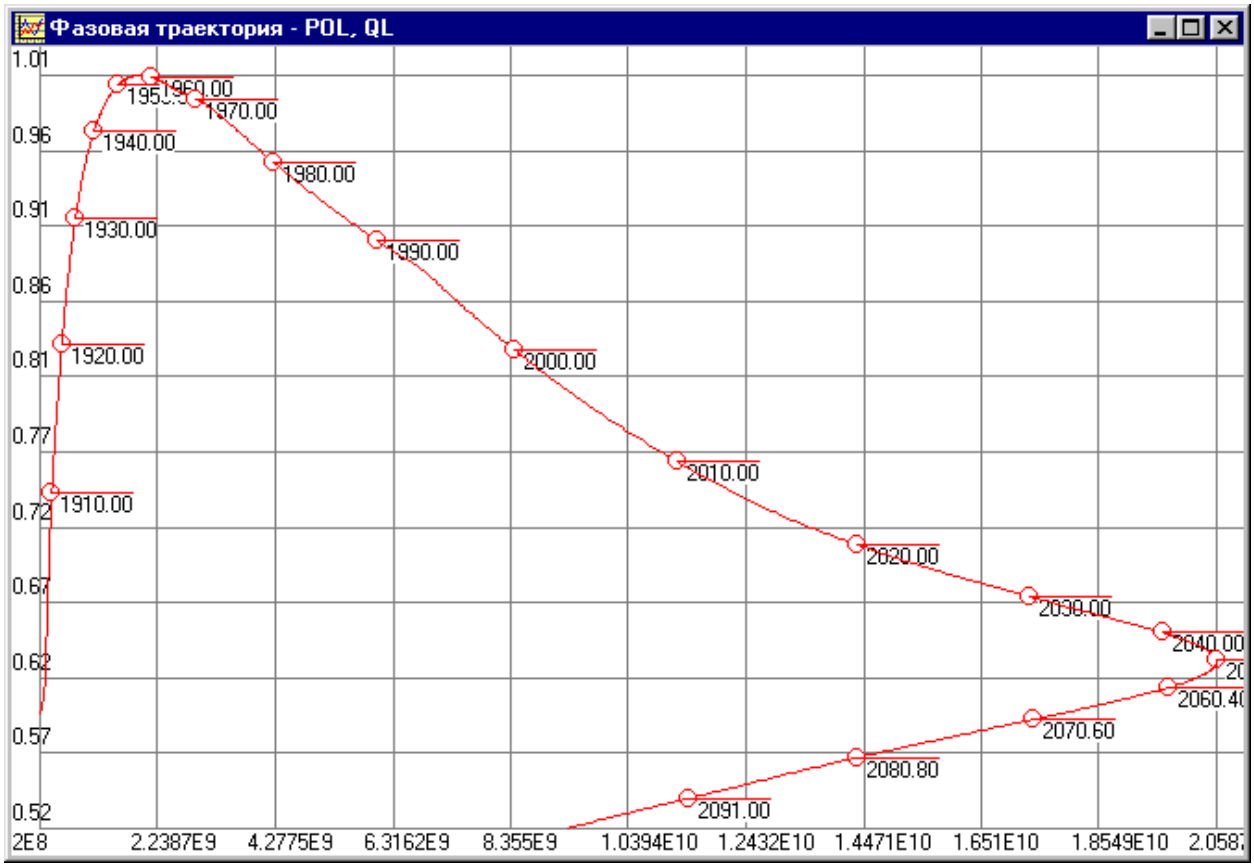


Рис. 12. Окно анализа «Фазовая траектория» для переменных POL, QL модели «GlobalDynamic»

6.10. Меню Помощь

Меню **Помощь** содержит подпункты:

- **Вызов справки;**
- **О программе…**

Команда **Вызов справки** запускает справочную систему с кратким руководством по работе с программой.

Команда **О программе…** сообщает информацию о версии системы и знакомит Вас с ее разработчиками.

6.11. Сообщения об ошибках

Ошибки на этапе лексического анализа:

1. Неправильная запись идентификатора;
2. Неправильная запись идентификатора или отсутствует операция;
3. Неправильная запись числа или отсутствует операция;
4. Неправильная запись числа;
5. Отсутствует начало комментария;
6. Неправильная запись идентификатора и отсутствует начало комментария;
7. Неправильная запись числа и отсутствует начало комментария;
8. Не закончена конструкция.

Ошибки на этапе синтаксического анализа:

1. Не хватает параметров;
2. Синтаксически неправильная конструкция;
3. Неправильное использование переменной;
4. Не определен шаг моделирования DT;
5. Не определено начальное время моделирования TIME;
6. Не определен период моделирования LENGTH;
7. Неправильное использование функции;
8. Повторное описание определяющего уравнения;
9. Не определено уравнение для переменной;
10. Не определен тип переменной.

Ошибки на этапе дополнительной проверки программы модели:

1. Нет ни одной переменной уровня;
2. Не определен тип переменной <имя>;
3. Не определено уравнение для переменной <имя>;

4. Неправильное использование переменной <имя> для уравнения переменной <имя>.

Ошибки на этапе моделирования:

1. Переменная <имя> образует циклическую зависимость с переменной <имя>;
2. Нет ни одной свободной переменной;
3. Ошибка моделирования в момент времени T. Задана слишком большая точность вычислений или проблема неопределенности.

ПРИЛОЖЕНИЕ 7. СПИСОК ФАЙЛОВ НА ДИСКЕТЕ

1. Source\DELTA.dpr – основной файл Delphi-проекта.
2. Source\About.dfm - исходный файл формы Delphi.
3. Source\TransForm.dfm - исходный файл формы Delphi.
4. Source\UCommonGraphParams.dfm - исходный файл формы Delphi.
5. Source\UPhaseTrackParams.dfm - исходный файл формы Delphi.
6. Source\UDefaltAnalysisParams.dfm - исходный файл формы Delphi.
7. Source\Unit1.dfm - исходный файл формы Delphi.
8. Source\UParamsForm.dfm - исходный файл формы Delphi.
9. Source\VisualDataF1.dfm - исходный файл формы Delphi.
10. Source\VisualDataF2.dfm - исходный файл формы Delphi.
11. Source\UDefaultParamsForm.dfm - исходный файл формы Delphi.
12. Source\UEnvironmentParams.dfm - исходный файл формы Delphi.
13. Source\UMessageBoxConfirm.dfm - исходный файл формы Delphi.
14. Source\UDBFExport.dfm - исходный файл формы Delphi.
15. Source\About.pas - исходный файл модуля Delphi.
16. Source\TransForm.pas - исходный файл модуля Delphi.
17. Source\UCommonGraphParams.pas - исходный файл модуля Delphi.

18. Source\UPhaseTrackParams.pas - исходный файл модуля Delphi.
19. Source\UDefaltAnalysisParams.pas - исходный файл модуля Delphi.
20. Source\Unit1.pas - исходный файл модуля Delphi.
21. Source\UParamsForm.pas - исходный файл модуля Delphi.
22. Source\VisualDataF1.pas - исходный файл модуля Delphi.
23. Source\VisualDataF2.pas - исходный файл модуля Delphi.
24. Source\UDefaultParamsForm.pas - исходный файл модуля Delphi.
25. Source\UEnvironmentParams.pas - исходный файл модуля Delphi.
26. Source\UMessageBoxConfirm.pas - исходный файл модуля Delphi.
27. Source\UDBFExport.pas - исходный файл модуля Delphi.
28. Source\Constants.pas - исходный файл модуля Delphi.
29. Source\NumDiff.pas - исходный файл модуля Delphi.
30. Source\SimThread.pas - исходный файл модуля Delphi.
31. Source\Translator.pas - исходный файл модуля Delphi.
32. Source\Types.pas - исходный файл модуля Delphi.
33. Setup\DELTA.exe – исполняемый модуль системы DELTA.
34. Setup\Readme.txt – текстовый файл описания.
35. Examples\GlobalDynamics.dyn – модель мировой динамики Дж. Форрестера на языке DYNAMO PLUS.

36. Examples\ШЛЯПА.dyn – демо-программа на языке DYNAMO PLUS.
37. Manual\User’s Guide.html – руководство пользователя системы DELTA.
38. Manual\Programmer’s Guide.html – руководство программиста.
39. Manual\Diplom.pdf – текст данной дипломной работы.

ПРИЛОЖЕНИЕ 8. ДИСКЕТА С СИСТЕМОЙ DELTA