# ANYDYNAMICS – НОВАЯ ВЕРСИЯ СРЕДЫ RAND MODEL DESIGNER

**Ю.Б. Колесов (Москва), Ю.Б. Сениченков (Санкт-Петербург)**

**Введение**

Современные среды моделирования, такие как Matlab-Simulunk, OpenModelica AnyDynamics, AnyLogic, – это сложно организованные программные продукты, помогающие пользователю строить компьютерную модель, проводить с ней вычислительные эксперименты, встраивать ее в программные и программно-аппаратные комплексы, использовать как прототип будущего устройства на этапе проектирования, а затем и как учебный тренажер на этапе эксплуатации сложных технических систем, учить компьютерному моделированию студентов.

Перечисленные среды моделирования относятся к универсальным, способным строить компьютерные модели, используя различные математические модели: {(сосредоточенные -- распределённые); (статические -- динамические); (дискретные -- непрерывные); (линейные – нелинейные); (детерминированные -- стохастические)}.

Всеядность универсальных сред, с одной стороны, недостаток, специализированные среды каждую свою модель строят обычно лучше, так и огромное преимущество, если речь заходит о многокомпонентных системах с компонентами различной природы, системах управлении сложными системами, проведении масштабных экспериментов, требующих разнообразных сложных компьютерных инструментов обработки и визуализации результатов эксперимента, кибер-физических системах.

Универсальные среды постоянно модифицируются и совершенствуются, чтобы удовлетворять постоянно возрастающим потребностям разработчиков традиционных наукоемких областей: кораблестроительной, авиационной, космической техники и других, но и запросам создателей бытовой техники, которая становятся все сложнее и сложнее. Универсальность среды связана не только с использованием различных математических моделей, но и с разнообразием применяемых технологий создания компьютерных моделей. В этой статье будет обсуждаться только универсальные среды с объектно-ориентированными языками моделирования высокого уровня.

**Универсальные языки моделирования**

**Однокомпонентные системы**

Самые простые модели универсальных сред это – однокомпонентные модели: непрерывные, дискретные и гибридные (событийно-управляемые) динамические системы. Здесь естественными конкурентами универсальных сред выступают математические пакеты Matlab, Maple, Mathematica и другие, оснащенные как символьными, так и численными решателями алгебраических, дифференциальных и алгебро-дифференциальных уравнений, и встроенными процедурами, помогающими пользователю анализировать и преобразовывать исходную модель к желаемому виду. В универсальных средах модель преобразуется в решаемую итоговую систему автоматически.

Графические языки для однокомпонентных систем – это языки уравнений. Формы записи дифференциальных и разностных уравнений (непрерывные, дискретные системы) в универсальных средах обычно просты и интуитивно понятны, поэтому многие среды часто не используют графические редакторы уравнений и обходятся конструкциями, свойственными алгоритмическим языкам. Общепринятым графическим языком представления гибридных систем является стандартизованный UML [1] язык машин состояний [2], модифицированный таким образом, что активности, приписанные состояниям, описывают поведение объекта не только в дискретном, но и в непрерывном (гибридном) времени (карты поведения).

Альтернатива языку машин состояний – это язык условных уравнений, в основе которого лежит конструкция y= **if** (Predicate) **then** y1 **else** y2. Сложные условные уравнения трудночитаемые и служат источником труднонаходимых ошибок. На практике используются оба подхода: в среде Matlab+Simulink – компонент StateFlow [3], в средах AnyLogic [23] – карты поведения, AnyDynamics [11,13] – карты поведения и условные уравнения, в среде OpenModelica [6] – условные уравнения.

**Многокомпонентные системы**

Многокомпонентные системы умеют строить многие среды, например, Matlab+Simulink, а также группа сред: OpenModelica, Dymola, MapleSim [4], SystemModeler [5], использующих язык Modelica [6]. Упоминание сред MapleSim, SystemModeler не случайно, и говорит о появлении новых подходов к построению универсальных сред.

Среды Simulink, Scilab [7], SimInTech [8], ISMA [9], и многие другие строят итоговую систему для устройств, состоящих только из компонент с входами-выходами. Среды, использующие язык Modelica, и AnyDynamics уже умеют дополнительно строить итоговые системы для схем, содержащих компоненты с контактами-потоками, и интересны разработчикам моделей с компонентами различной физической природы («физическое» моделирование, кибер-физические системы [10]). Среды AnyLogic, AnyDynamics добавляют к числу допустимых компонентов компоненты-агенты. Агентный подход применяют для исследования систем, которые не удается описать единой итоговой системой уравнений, но можно использовать локальные законы поведения агентов и правила их взаимодействия, и изучать статистическими методами, многократно разыгрывая различные сценарии взаимодействия между агентами.

Таким образом можно выделить следующие компоненты систем универсальных сред: {Изолированные компоненты, Компоненты с входами-выходами, Компоненты с контактами-потоками, Агенты}.

Все перечисленные среды разрешают использовать в качестве компонентных уравнений событийно-управляемые уравнения (синонимы: условные уравнения, кусочно-непрерывные уравнения). Стандартом машин состояний для дискретных систем можно считать машины состояний языка UML. Однако в различных универсальных средах машины состояний для работы в непрерывном времени реализованы по-разному. Компонента StateFlow среды Simulink наиболее близка к машине состояний языка UML. Карты поведения среды AnyDynamics соответствуют стандарту UML, но не содержат исторических и параллельных состояний. Среды, использующие язык Modelica, допускают только условные уравнения, которые можно условно назвать примитивными машинами состояний.

Для средств описания поведения можно предложить такую классификацию: {машины состояний, условные уравнения}.

Одной из важнейших задач для многокомпонентных систем является задача построения итоговой системы уравнений.

Для однокомпонентных моделей с машинами состояний это задача решается достаточно просто. Анализируя локальные классы машины состояния, или структуру условных уравнений, можно выделить независимые системы уравнений, и уже на этапе компиляции построить нужный код. Какие из них будут решаться, и как часто, зависит уже от последовательности реализовавшихся событий. Уравнения могут сильно отличаться и задача автоматического выбора решателя может быть чрезвычайно важна, так как от выбора решателя зависит время проведения вычислительного эксперимента и точность вычислений. Публикаций, посвященных решению этой проблемы, чрезвычайно мало.

Модели из блоков с входами-выходами, с внутренними машинами состояний, отличаются от однокомпонентных моделей только «размерностью» композиции внутренних машин состояний: число независимых систем может оказаться чрезвычайно большим, но уравнения для них можно построить на этапе компиляции.

Построение итоговой системы в случае, если используются компоненты с контактами-потоками с внутренними машинами состояний, уже не может быть выполнено на основе базовой системы для независимых блоков. Однако и в этом случае удается осуществить достаточно быстрое формирование системы на этапе исполнения при возникновении событий, приводящих к изменению траектории. Таким образом строятся только те системы, что возникают при реализации конкретной последовательности событий.

Наконец, обратимся к используемым в универсальных средах языкам моделирования и языкам проведения вычислительного эксперимента. Пользователи среды обычно используют графические формы языков или создают модель прямо на языке моделирования. Успех математического пакета Matlab и среды моделирования Simulink с пользовательскими языками высокого уровня вдохновил и разработчиков других математических пакетов. Появились среды моделирования MapleSim, SystemModeler. Эти среды задуманы как среды с высокоуровневыми языками моделирования, позволяющие различным категориям пользователей использовать всю вычислительную мощь математических пакетов применительно к иерархическим, событийно-управляемым многокомпонентным системам.

Все перечисленные среды для построения сложных моделей большими коллективами разработчиков используют объектно-ориентированный подход, разница только в механизмах создания, повторного использования, модификации модулей, организации работы нескольких групп над одним проектом, доступных пользователю, то есть о полноте использования возможностей объектно-ориентированного подхода.

Среду Matlab+Simulink можно формально назвать средой, поддерживающей объектно-ориентированное моделирование, так как, создавая новую модель, пользователь использует экземпляры базовых классов, и может строить с их помощью новые пользовательские классы, соответствующие новым структурным схемам.

Языки сред OpenModelica, Anydynamics используют многие из возможностей объектно-ориентированного подхода, но опять-таки по-разному. Например, в языке Modelica разрешается множественное наследование, а в языке Model Vision Language среды Anydynamics – нет. В следующих разделах статьи будет сделана попытка сравнительного анализа сред OpenModelica и Anydynamics.

Идея стандартизации языков моделирования обсуждается давно, примерами могут служить языки UML, Modelica, ставшими стандартами de-facto, однако реальные технологии моделирования и результаты моделирования в различных средах могут существенно различаться.

**Системы с переменной структурой**

Модели переменной структуры имеют большое практическое значение [12]. К наиболее известным моделям переменной структуры относятся системы массового обслуживания. Универсальные среды, использующие машины состояний и имеющие создавать экземпляры классов и связи между объектами по ходу выполнения модели, способны создавать модели переменной структуры. Операторы AnyDynamics: {new <имя_класса> [(<действительные_значения_переменных>)] [name <строковое_выражение>] – возвращает ссылку на новый экземпляр указанного класса с указанными значениями переменных и указанным именем (если имя не указано, оно будет формироваться автоматически); destroy(Obj) – уничтожает динамический объект

Obj; connect(v1,v2) – создает динамическую связь между внешними переменными v1 и v2; disconnect(v1,v2) – уничтожает динамическую связь между внешними переменными v1 и v2;} обеспечивают создание новой структуры, а машины состояний помогают ее менять, если это необходимо.

Возможность динамически менять структуру модели резко усложняет процесс формирования итоговой системы уравнений. Например, в языке Modelica даже для более простого случая, когда меняются только уравнения в условных уравнениях, накладываются серьезные ограничения – число уравнений должно быть постоянным. В среде AnyDynamics итоговые уравнения при изменении структуры и уравнений удается строить на этапе исполнения, без существенных временных потерь.

**AnyDynamics. История создания**

AnyDynamics – это универсальная среда объектно-ориентированного моделирования, первые версии которой (Model Vision Studium) появились в конце 20-го столетия [13]. Среда обладала своим объектно-ориентированным языком моделирования – Model Vision Language (MVL). Первоначально среда позволяла создавать только многокомпонентные системы из блоков с входами-выходами, но уже отличилась от Simulink, Scilab наличием компонентных машин состояний, реализованных в форме карт поведения [14] (Behavior-Charts: B-Charts), прототипом которых послужили карты состояний (State-Charts) Харрелла [2], при этом пользователь мог создавать новые и модифицировать существующие компоненты самостоятельно, используя механизм наследования.

С самого начала созданные модели могли работать как под управлением среды, так и как независимые приложения в реальном времени. С 2006 по 2020 год в среду добавлялись новые типы компонентов: компоненты с контактами-потоками, агенты; совершенствовался визуализатор поведения: появился новый отладчик, разнообразные виды диаграмм, библиотека стандартных вычислительных экспериментов. Создается промышленная версия среды MvStudiumProfessional [15] для разработки морских тренажеров. Среда меняет свое название на Rand Model Designer, и признается лучшим отечественным продуктом по версии журнала PC_Magazine [16]. Многочисленные последовательные изменения привели к необходимости создать новую версию с новой архитектурой и новыми компонентами. AnyDynamics – это новая версия среды Rand Model Designer (2008-2020), ставшая доступной пользователям в конце 2020 года.

**AnyDynamics. Редактор моделей**

Редактор моделей позволяет создавать модели для отладки; исполняемые, оптимизированные модели; редактировать модели непосредственно на языке моделирования или используя графические редакторы. Для записи алгоритмических операторов можно использовать языки Ada или C#.

**Непрерывные системы**

Простейшими моделями среды являются «Непрерывные системы», к которым относятся как классические динамические системы (системы обыкновенных дифференциальных уравнений, разрешенные относительно производных, с заданными начальными условиями, и гладкой правой частью, гарантирующей существование и единственность решения), так и приводимые к ним системы, неразрешенные относительно производных (M(x)*dx/dt=f(x) =>

{M(x)*y=f(x);dx/dt=y}, и системы алгебро-дифференциальных уравнений. Учитывая, что в инженерной практике встречаются и модели, заданные явными зависимостями от времени, системами алгебраических уравнений, с коэффициентами, зависящими от времени, и при вычислении правых частей любых уравнений могут использоваться алгоритмические функции, этот тип моделей также включается в список непрерывных систем, «инженерных динамических систем».

Гарантировать существование и единственность решения для инженерных динамических систем априори уже трудно, и это должен делать сам пользователь. В случае динамических систем гладкость правых частей можно анализировать автоматически (используется процедура, аналогичная процедуре **smooth** (p, expr) языка Modelica).

Пользовательские непрерывные системы упорядочиваются и превращаются в итоговые системы алгебро-дифференциальных уравнений, передающиеся решателям. Подстановки упорядочиваются с помощью алгоритма Холла, найденные алгебраические циклы превращаются в уравнения. Далее следует назначение искомых переменных. В алгебро-дифференциальных уравнениях, понижается индекс, если это необходимо. Понижение индекса производится с помощью символьного дифференцирования. Существует также возможность понижать индекс, строя эквивалентную систему с помощью дополнительных дифференциальных уравнений, численно вычисляющих нужные производные [17], но это – эвристический прием, который может приводить к ошибкам в случае сложных событийно-управляемых систем.

Непрерывные системы можно рассматривать как изолированные однокомпонентные модели с тривиальным интерфейсом (нет внешних переменных). В таблице 1 приведены примеры непрерывных систем в среде OpenModelica и AnyDynamics.

**Гибридные системы**

Гибридные или событийно-управляемые динамические системы, - это второй класс простых однокомпонентных систем, который можно считать наследником класса «Непрерывные системы», с новыv атрибутом «Карта поведения».

Карта поведения соответствует стандарту машин состояний языка UML, но не имеет исторических и параллельных состояний. Do-активности задаются с помощью локальных классов двух типов, соответствующих непрерывным системам и гибридным, со своими, возможно иерархическими картами поведения. Локальные классы могут модифицироваться с помощью механизма наследования, с добавлением и переопределением уравнений класса-родителя.

Помимо создания событийно управляемых систем с непрерывными активностями и гибридным временем (таблица 2), можно создавать дискретные событийно-управляемые системы, а также с помощью карт поведения решать разностные уравнения (таблица 3). Язык Modelica поддерживает машины состояний, но Do-активности должны быть дискретными (таблица 2, 1а), для непрерывных активностей используются условные уравнения (таблица 2, 1б). В среде AnyDynamics расширения UML машин состояний с непрерывными активностями (таблица 2, 1в).

Таблица 1. Непрерывные системы

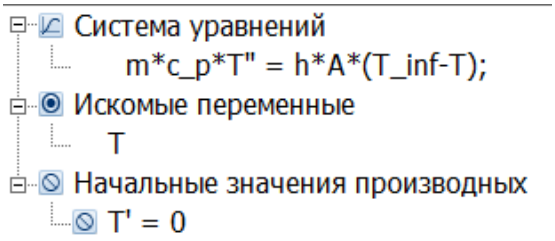
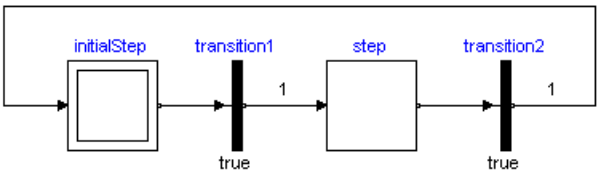
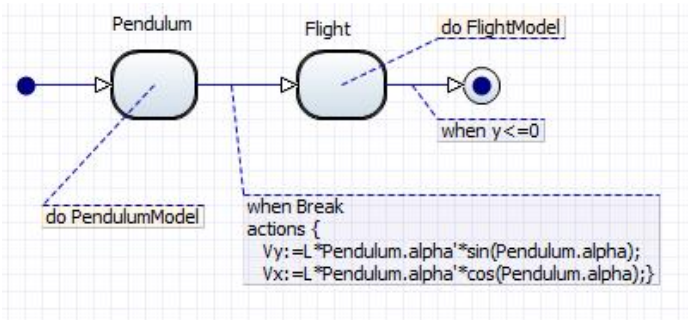
| № | OpenModelica | AnyDynamics |
|---|---|---|
| 1 | equation<br>der(x) = A*x + B*u;<br>y = C*x + D*u;<br>end; | $\frac{dx}{dt} = A \cdot x + B \cdot u$<br>$y = C \cdot x + D \cdot u$ |
| 2 | **initial equation**<br>T = T0 "Specify initial value for T";<br>**equation**<br>m*c_p***der**(T) = h*A*(T_inf-T) "Newton's law of cooling";<br>**end** | $m \cdot c\_p \cdot \frac{dT}{dt} = h \cdot A \cdot (T\_inf - T)$ |
| 3 | **initial equation**<br>T = T0 "Specify initial value for T";<br>V = V0;<br>**Equation**<br>**der**(T)=V;<br>m*c_p***der**(V) = h*A*(T_inf-T) "Newton's law of cooling";<br>**end** |  |

Таблица 2. Машины состояний

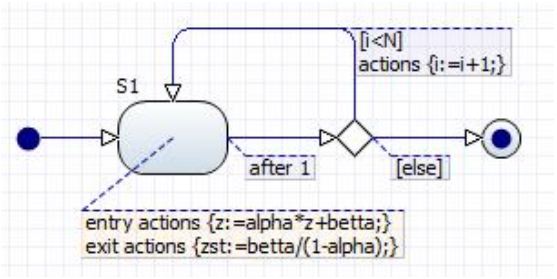
| № | OpenModelica |
|---|---|
| 1а |  |
| 1б | der(h) = v;<br>der(v) = **if** flying **then** -g **else** 0;<br>flying = not (h <= 0 and v <= 0);<br>**when** h < 0 **then**<br>reinit(v, -e * pre(v));<br>**end when** |
| | **AnyDynamics** |
| 1в |  |

**Компоненты с входами/выходами**

Компоненты с входами-выходами (I/O) можно считать наследниками изолированных гибридных систем, у которых появились новые атрибуты: внешние переменные входы-выходы, структурная схема и интерфейс.

В AnyDynamics существует библиотеки SysLib, коллекция основных компонентов пакета Simulink. В отличии от пользователей среды Simulink, пользователи AnyDynamics видят описание блоков на языке MVL и могут использовать и модифицировать их как обычные компоненты. Эту библиотеку удобно использовать в учебном процессе для иллюстрации техники создания компонентных моделей в среде Simulink. Аналогичная библиотека есть в среде OpenModelica (Modelica/Blocks).

Таблица 3. Дискретные системы

| OpenModelica | AnyDynamics |
|---|---|
| **equation**<br>**when** sample() **then**<br>x = a * pre(x) + b * pre(u);<br>**end when**;<br>y = x; |  |

**Компоненты с контактами/потоками**

Компоненты с контактами и потоками (C/F) ничем не отличаются от компонентов с входами-выходами, только теперь внешними переменными являются контакты и потоки, а связи обычно реализуются с помощью коннекторов. Внутренние компоненты, как и компоненты с входами выходами, могут иметь внутреннюю структуру и внутренние машины состояний. И это основное отличие от аналогичных компонентов языка Modelica.

**Модели общего вида**

Наиболее сложно организованной моделью является многокомпонентная модель с собственной картой поведения – картой поведения модели и иерархической структурной схемой с событийно управляемыми компонентами.

Карта поведения модели может использоваться: для управления вычислительным экспериментом; изменения структуры модели с помощью ключей, разрывающих связи, или с помощью создания новых объектов со связями (рис. 8); для проведения «мгновенных» вычислений в «ортогональном» времени.

До сих пор мы рассматривали карты поведения с do-активностями, соответствовавшим поведению непрерывных, гибридных систем, и «инженерных динамических систем», не гарантирующих существование и единственность решения уравнений. Однако и любой экземпляр класса, если он демонстрирует поведение, зависящее от времени, можно считать сложной «инженерной динамической системой», и таким образом связывать его поведение с do-активностью карты поведения.

При этом любое состояние карты поведения может иметь входные и выходные действия (аналог – секция «algorithm» языка Modelica), выполняющиеся «мгновенно», без продвижения непрерывного модельного времени, и длительное поведение (аналог - секция «equation» языка Modelica), с изменением непрерывной составляющей гибридного времени. Запрещение продвижения непрерывного модельного времени при выполнении do-активностей в карте поведения модели (и только в ней), называется

вычислениями в ортогональном времени. Это можно рассматривать как «мгновенный» прогноз, оптимизацию, предотвращение нежелательных действий до начала очередных действий.

**Агенты**

Модели общего вида позволяют создавать агентные системы. Карта повеления управляет численностью агентов и их связями (таблица 4). Экземпляры классов агентов и связи между ними могут создаваться и уничтожаться по мере необходимости в течение жизни всей модели.

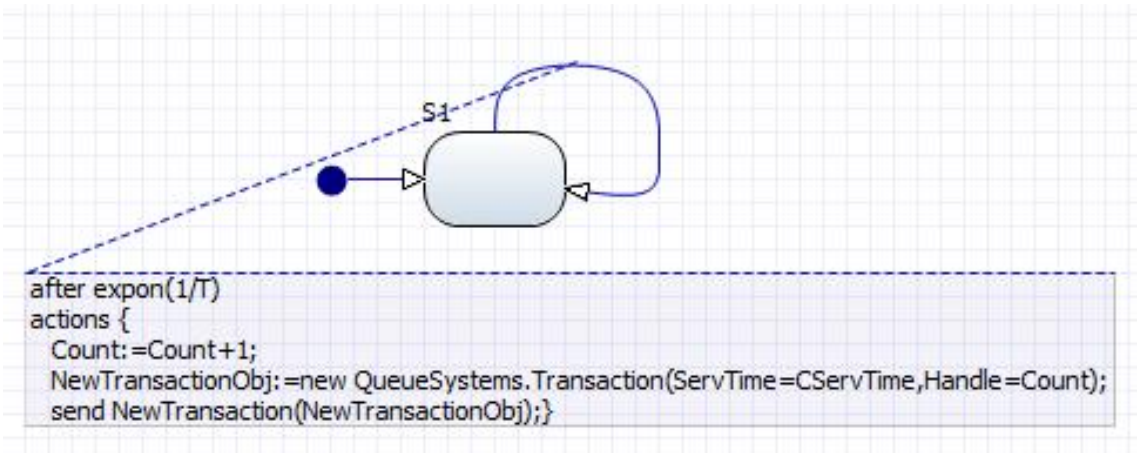
**AnyDynamics. Визуализатор поведения**

Визуализатор поведения – это компонент, который позволяет отлаживать созданную модель (ее отладочную версию – debug version); измерять временные характеристики отлаженной модели (рабочей версии кода – release version); проводить вычислительные эксперименты, в том числе, и стандартные, реализованные в виде встроенных процедур; управлять выбором численных методов и менять режимы их работы.

**Отладка**

При отладке нас интересует реализовавшаяся последовательность событий конкретного эксперимента и порожденный ею путь в композиции машин состояний модели. Переход от состояния к состоянию сопровождается выполнением мгновенных действий на переходах, входных и выходных действий состояний, вычислением функций и выполнением процедур, написанных на языке моделирования. Состоянию могут быть приписаны либо уравнения, либо код и уравнения, определяемые экземпляром приписанного конкретного класса. Все это осложняет реализацию отладчика.

Таблица 4. Агенты

| № | AnyDynamics |
|---|---|
| 1 |  |

Наиболее простым по реализации и наиболее трудоемким для использования является трассировка различной степени детализации, фиксирующая, как «логику» поведения модели, так и ход численных расчетов. Существующая трассировка работает в гибридном времени, то есть отслеживает вычисления во временных «щелях» и на участках непрерывного времени. При трассировке численных методов, программная реализация численных методов рассматривается как «черный ящик», который по запросу управляющей программы (времени и текущему значению переменных)

возвращает решение в затребованной временной точке. Последовательность этих запросов и ответов численного метода и размешается в выходном файле.

Доступна трассировка компонентных машин состояния. Отладчик фиксирует события, приводящие к смене состояний, указывает путь, ведущий к новому текущему состоянию, и может по шагам отслеживать выполнение операторов языка MVL, связанных с дискретными действиями. Приостановка выполнения кода может быть привязана ко времени, выполнению условия, срабатываю перехода, входу в состояние, получению сигнала. Приостановка может сопровождаться сообщениями, присваиванием конкретным переменным нужных, «отладочных» значений. Такая последовательность отладочных действий может быть зафиксирована в виде Плана отладки.

Пользователь также использовать пошаговую отладку последовательностей операторов в визуальной модели, аналогичную инструментам визуального программирования.

**Визуализация поведения**

Визуализировать поведения можно с помощью временных и фазовых диаграмм, 2-D и 3-D анимации, гистограмм.

Временные диаграммы (2-D и 3-D) и фазовые диаграммы обладают типичными для этого вида диаграмм возможностями, но в отличие, например, от среды OpenModelica, строятся по ходу выполнения. 2-D анимация позволяет создавать окна с измерительными и управляющими приборами и связывать их с интересующими переменными. 3-D анимации позволяет составлять образы наблюдаемых объектов, как композицию базовых элементов – шаров, цилиндров, пружин, шестеренок, и перемещать ее как единое целое в трехмерном пространстве по заданной траектории.

**Стандартные вычислительные эксперименты**

Визуальная модель позволяет проводить следующие типовые вычислительные эксперименты:

· получение параметрической зависимости;

· определение вероятности события при заданных законах распределения параметров модели;

· определение математического ожидания и среднеквадратического отклонения переменной модели при заданных законах распределения параметров модели;

· оптимизация – нахождение совокупности оптимальных значений выбранных параметров модели, минимизирующих или максимизирующих значение заданной целевой функции с учетом ограничений;

· расчет вариантов**;**

· анализ глобальной чувствительности – ранжирование параметров модели по степени их влияния на выходную переменную модели.

**Численные методы**

Пользователи могут описывать непрерывное поведение в виде систем алгебраических, обыкновенных дифференциальных, или алгебро-дифференциальных систем [17]. В качестве программных реализаций численных методов использовался либо оригинальный код, либо адаптированные в соответствии с требованиями среды коды пакетов BLAS, LINPACK, EISPACK, MINPACK, ODEPACK, а также из книг с открытым программным обеспечением, таких как [18-22].

Алгебраические уравнения сортируются на линейные и нелинейные. Линейные решаются с помощью методов из коллекций LINPACK, SPARSRPAK. Нелинейные – методом Ньютона или Пауэлла. Дифференциальные уравнения решаются методами Рунге-Кутты (явными и неявными), линейными многошаговыми методами (ODEPACK,

DDASSL, (semi-explicit index-1 form LSODA). Дифференциальные уравнения, не разрешенные относительно производных, приводятся к алгебро-дифференциальным уравнениям в полу-явной форме. В случае алгебро-дифференциальных уравнений индекс (semi-explicit index-1 form) понижается с помощью символьного дифференцирования.

**Заключение**

Среду AdyDynamics можно найти на сайте [www.mvstudium.com]. Доступны бесплатная (студенческая) версия и профессиональная, позволяющая создавать модели, встроенные в приложения. Тем самым преподаватели могут выбирать для учебного процесса либо OpenModelica, либо AnyDynamics. Учебников, посвященных среде AnyDynamics на русском языке, больше, чем среде OpenModelica. Учебные пособия [23-25] предназначены для курсов «Математическое моделирование» и «Технологии моделирования».

**Литература**


1. **Rumbaugh D., Jacobson I., Booch G.** Unified Software Development Process. AddisonFWesley, 1999.
2. **Harel D.**: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming (8), 231-274 (1987).
3. MathWorks,https: //www.mathworks.com/products/stateflow.html, last ac. 2021/05/05.
4. MapleSim, https://www.maplesoft.com/products/maplesim/, last accessed 2021/05/05.
5. SystemModeler, https://www.wolfram.com/system-modeler/, last accessed 2021/05/05.
6. Modelica association, https://modelica.org/, last accessed 2021/05/05.
7. Scilab, https://www.scilab.org/, last accessed 2021/05/05.
8. SimInTech, https://simintech.ru/, last accessed 2021/05/05.
9. **Shornikov Yu., Dostovalov D.** Fundamentals of event-continuous systems simulation theory. Novosibirsk, 2018.
10. **Fradkov A.** Cybernetic physics. Science, St. Petersburg, 2003.
11. **Kolesov Yu., Isakov A., Senichenkov Yu.** A new tool for visual modeling – Rand Model Designer 7 // In: 8th Vienna Conference on Mathematical Modeling, pp. 345-347, IFAC-PapersOnLine 28 (1) (2015).
12. **Kolesov Yu., Urqua A., Martin-Villaba C., Senichenkov Yu.** Simulation of variable structure models using Rand Model Designer. In: Proceedings - 8th EUROSIM Congress on Modelling and Simulation. Invited paper, EUROSIM (2016).
13. **Kolesov Yu., Senichenkov Yu.** Model Vision 3.0 for Windows 95/NT. The graphical environment for complex dynamic system design. In: ICI&C'97. International conference on Informatics and Control. St. Petersburg, pp. 764-768 (1997).
14. **Kolesov Yu., Senichenkov Yu.** Visual Specification language intended for event-driven hierarchical dynamical systems with variable structure. In: ICI&C'97. International conference on Informatics and Control. St. Petersburg, pp. 704-711 (1997).
15. **Tarasov S.** Application Experience of Component Modeling in Transas Group's Training System Development for Cargo-Ballast and Technological Operations. Automation and Remote Control, 77( 6), pp. 1106–1114. (2016).
16. Russian software 2015: innovations and advancement. PC Magazine: Russian Edition. 52-59, 11 (2015).
17. **Isakov A., Senichenkov Yu.** Rand Model Designer' numerical library. In: Linköping Electronic Conference Proceedings 142:140, s. 953-958 (2016).
18. **Hairer E., Norsett S.P., Wanner G.** Solving ordinary differential equations I. Nonstiff Problems. Springer-Verlag. (1987).

19. **Hairer E., Wanner G.** Solving ordinary differential equations II. Stiff and Differential-algebraic Problems. Springer-Verlag. (1991, 1996).
20. **Hindmarsh Alan C.** (1983) ODEPACK, A Systematized Collection of ODE Solvers; in p.55-64 of Stepleman, R.W. et al.[ed.] (1983) Scientific Computing, North-Holland, Amsterdam.
21. **Petzold Linda R.** (1983) Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations. Siam J. Sci. Stat. Computer, 4, 136-148.
22. **Andrei V. Borshchev, Yuri B. Kolesov and Yuri B. Senichenkov.** Java Engine for UML Based Hybrid State Machines.2000 Winter Simulation Conference (WSC'00), December 10-13, 2000, Orlando, Florida, USA.
23. **Сениченков Ю.Б.** Компонентное моделирование сложных динамических систем: сборник заданий. ISBN 978-5-7422-6328-9. СПб. : ПОЛИТЕХ-ПРЕСС, 2019.122 с.
24. **Сениченков Ю.Б., Ампилова Н.Б., Тимофеев Е.Л.** Математическое моделирование сложных динамических систем: сборник заданий. ISBN 978-5-7422-5960-2. СПб. : ПОЛИТЕХ-ПРЕСС, 2018. 108 с.
25. **Колесов Ю.Б., Сениченков Ю.Б.** Компонентное моделирование сложных динамических систем: учебное пособие. ISBN 978-5-7422-6685-2 СПб.: ПОЛИТЕХ-ПРЕСС, 2019.
26. **Колесов Ю.Б., Сениченков Ю.Б.** Математическое моделирование сложных динамических систем: учебное пособие. ISBN 978-5-7422-6684-3. СПб.: ПОЛИТЕХ-ПРЕСС, 2019.