УДК 519.63+004.94

ОБЗОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ ПОСТРОЕНИЯ И ИССЛЕДОВАНИЯ МОДЕЛЕЙ СЛОЖНЫХ СИСТЕМ

Головинский М.И., Григорьев А.В.
Донецкий национальный технический университет
кафедра прикладной математики и информатики
E-mail: shinigami503@gmail.com, grigorie@pmi.dgtu.donetsk.ua

Аннотация

Головинский М.И., Григорьев А.В. Обзор инструментальных средств для построения и исследования моделей сложных систем. *Рассмотрены некоторые типы дифференциальных уравнений и методики их решений. Сделан обзор существующих математических и программных средств для решения поставленной задачи. Определен подход формирования инструмента для решения задач моделирования сложных систем.*

Общая постановка проблемы

В наше время стало возможно проверять качество работы системы сложного объекта, даже тогда когда существует только проект объекта, и определены его только самые общие физико-динамические характеристики. Для построения и исследования таких сложных систем используются различные математические и моделирующие пакеты. Отличие моделей заключается в их предназначении, они подразделяются на стационарные и динамические. Стационарные, описывают модель как нечто постоянное, и может быть описана при помощи системы обычных уравнений, динамические же, описывают модель как сложную изменяющуюся по времени системы взаимосвязанных показателей такую систему лучше описывать в форме системы дифференциальных уравнений.

Для решений задач по моделированию объектов аэродинамики и проектирования нагрузки и вентиляции в шахтах, в большей степени подходят уравнения описывающие динамику поведения, а соответственно системы дифференциальных уравнений.

Исследования

Метод конечных элементов (МКЭ) [1,2] — численный метод решения задач прикладной физики. Метод широко используется для решения задач механики деформируемого твёрдого тела, теплообмена, гидродинамики и электродинамики.

С точки зрения вычислительной математики, идея метода конечных элементов заключается в том, что минимизация функционала вариационной задачи осуществляется на совокупности функций, каждая из которых определена на своей подобласти, для численного анализа системы позволяет рассматривать его как одну из конкретных ветвей диакоптики — общего метода исследования систем путём их расчленения.

Программирование в ограничениях [3] — максимально декларативно и основано на описании модели задачи, а не алгоритма ее решения. Модель специфицируется в виде неупорядоченной совокупности отношений, которые соответствуют связям, существующим между параметрами задачи. Эти отношения, называемые общим термином «ограничения» могут иметь вид уравнений, неравенств, логических выражений и т. п.

Одним из наиболее развитых и практически значимых подходов, относящихся к программированию в ограничениях, являются недоопределенные модели. Технология H-моделей выделяется среди других подходов вычислительной мощностью, универсальностью и эффективностью.

Обобщенная вычислительная модель M = (V, W, C, R) состоит из следующих четырех множеств [4]:

V – множество объектов из заданной предметной области,

R – множество ограничений на значениях объектов из V,

W – множество функций присваивания,

C – множество функций проверки корректности.

Каждому объекту v ∈ V сопоставлены: универсум Xv; начальное значение из универсума (точное, недоопределенное, или полная неопределенность); функция присваивания Wv; функция проверки корректности Cv.

Функция присваивания [5] — это двухместная функция, работающая при каждой попытке присваивания очередного значения объекту v ∈ V и определяющая новое значение объекта как функцию от текущего и присваиваемого значений.

Функция проверки корректности — это унарный предикат, который исполняется в случае, если значение объекта x изменилось, и проверяет правильность этого нового значения. Ограничения из R должны быть функционально интерпретируемыми.

На уровне интерпретации ОВМ представляется двудольным ориентированным графом, в котором выделены два типа вершин: объекты и функции. Дуги связывают функциональные и объектные вершины. Входящие в вершину-функцию дуги соотносят с ней объекты, значения которых выступают в качестве входных аргументов для функции, исходящие — указывают на объекты, в которые должна производиться запись вырабатываемых функцией результатов.

Имитационная модель [6] — логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

Simulink [7] – это инструмент для моделирования и анализа многодоменных динамических систем. Его основной интерфейс – графический. Он представлен в виде графического блока диаграмм инструментов и настраиваемого набора библиотек различных блоков. Simulink широко используется в теории управления и цифровой обработке сигнала для моделирования многодоменного и модельно-ориентированного проектирования.

AnyLogik [8] — программное обеспечение для имитационного моделирования сложных систем и процессов. Модели AnyLogik могут быть основаны на любой из основных парадигм имитационного моделирования: моделирование, системная, и агентное моделирование.

Графическая среда моделирования AnyLogic включает в себя следующие элементы:

- Stock & Flow Diagrams (диаграмма потоков и накопителей) применяется при разработке моделей, используя метод системной динамики;

- Statecharts (карты состояний) в основном используется в агентных моделях для определения поведения агентов. Но также часто используется в дискретно-событийном моделировании, например для симуляции машинных сбоев;

- Action charts (блок-схемы) используется для построения алгоритмов. Применяется в дискретно-событийном моделировании (маршрутизация звонков) и агентном моделировании (для логики решений агента).

Process flowcharts (процессные диаграммы) основная конструкция, используемая для определения процессов в дискретно-событийном моделировании.

Ниже в таблице (табл. 1) приведены сравнительные характеристики сред моделирования.

Таблица 1 – Сравнительные характеристики

Среда	Графический интерфейс	Библиотеки расширений	Язык моделирования	Моделируемые системы	Режим работы
Simulink	+	+	MATLAB	Все	с MATLAB
AnyLogik	+	+	Java	Все	Самост.

Система программирования Mozart [9] – это профессиональная платформа для разработки интеллектуальных и распределенных приложений. Эта система – результат длительных исследований в дизайне и реализации языков программирования, основанных на программировании в ограничениях, распределенного программирования и интерфейсов типа человек-компьютер.

Mozart включает в себя большинство концепций популярных парадигм программирования, в том числе логического, функционального, императивного, объектно-ориентированного, программирования с ограничениями, распределённого и параллельного программирования [10]. С одной стороны, Mozart обладает простой формальной семантикой, а с другой — для него создана эффективная программная реализация. Так, на уровне языковых конструкций поддерживается простое и естественное распараллеливание и распределение вычислений по сети, что позволяет легко создавать отказоустойчивые приложения. Для реализации программирования в ограничениях в языке Mozart вводится концепция вычислительных пространств, в которой происходит поиск решения.

Структуры данных. Ядро языка образовано несколькими основными структурами данных, но может быть расширено.

Основные структуры данных:

- Числа (целые или с плавающей запятой);
- Записи (для группировки элементов данных);
- Списки (простые линейные структуры).

Эти значения являются константными сущностями первого рода, типизация при этом динамическая. Если программа обнаруживает несвязанную переменную, она ожидает до тех пор, пока переменной будет присвоено значение.

Функции являются сущностями первого рода, что позволяет применять парадигму функционального программирования.

Для полноценного использования многоядерной архитектуры компьютеров, компьютерных сетей и вычислительных кластеров используются асинхронные агенты.

Существуют и другие языки, которые в той или иной степени поддержвают парадигму программирования в ограничениях, как то UniCalc [11], Prolog III [12], CHIP [13], Mathematica [14]. Ниже приведена таблица (табл. 2) сравнения характеристик этих языков.

Таблица 2 – Сравнительная характеристика языков программирования

	UniCalc	Prolog III	CHIP	Mozart	Mathematica
GUI	+	-	+	+	-
ООП	-	+	+	+	-
Distributed Programming Support	-	-	-	+	-
CPS	+	-	+	+	-
CPL	+	+	+	+	-
Решение ОДУ УЧП	-	-	-	+	+
Решение СЛУ/СЛН	+	+	+	+	+
GUI constructor	-	-	-	-	-
Интеллектуальный поиск оптимального решения	-	-	-	-	-

Выводы

При анализе математических средств решения задач построения для динамических сложных моделей выявлены оптимальные для поставленной задачи математические методы. В ходе анализа существующих решений в сфере программных средств были выявлены недостатки и достоинства существующих продуктов, которые были учтены и проанализированы.

Для обеспечения поставленных целей моделирования сложных систем, система моделирования должна обладать:

- Способность модели быстро обучаться и переобучаться, для этого целесообразно использовать технологию ИНС основанную на методе программирования в ограничениях;
- Способность системы распараллеливать процесс моделирования и обучения;
- В качестве базисных отношений между элементами, использовать отношения, описываемые дифференциальными уравнениями;
- В качестве метода решения (аппроксимации решения) использовать метод конечных элементов;
- Способность системы доопределять пропущенные параметры и отношения в процессе обучения;
- Система текстового задания исходных параметров и отношений между элементами моделируемой системы;
- Графический редактор.

Литература

1. Митчелл Э. Метод конечных элементов для уравнений с частными производными. – М. Мир, 1981;
2. Метод конечных элементов — Википедия http://ru.wikipedia.org/wiki/Метод_конечных_элементов;
3. А.С.Нариньяни, В.В. Телерман, Д.М. Ушаков, И.Е. Швецов - Программирование в ограничениях и недоопределенные модели. //Информационные технологии №7, 1998. М., Издательство “Машиностроение”. -С. 13-22;
4. Григорьев А.В. Программирование в ограничениях как метод численного решения дифференциальных уравнений в инструментальной оболочке для создания интеллектуальных САПР // Наукові праці Донецького національного технічного університету. Випуск 70. Серія: «Інформатика, кібернетика та обчислювальна техніка».– Донецьк: ДонНТУ, 2003 – С. 108-116.;
5. Лю Б. Теория и практика неопределенного программирования / Б. Лю; Пер. с англ. М.: БИНОМ. Лаборатория знаний, 2005. 416 С.: ил.
6. Имитационное моделирование — Википедия http://ru.wikipedia.org/wiki/Имитационное_моделирование;
7. Simulink - Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Simulink;
8. AnyLogic — Википедия http://ru.wikipedia.org/wiki/AnyLogic;
9. The Mozart Programming System http://www.mozart-oz.org/;
10. Oz (язык программирования) — Википедия http://ru.wikipedia.org/wiki/Oz_(Язык_программирования);
11. Е.Ю. Ботоева 2- и 3-мерная визуализация множества решений в системе UniCalc.– Новосибирск, 2006;
12. Логическое программирование в ограничениях - Искусственный интеллект http://rriai.org.ru/logicheskoe-programmirovanie-v-ogranicheniyah-3.html;
13. CHIP V5 from COSYTEC, Second Generation of Constraint Programming. http://www.cosytec.com/production_scheduling/chip/chip_technology.htm;
14. Mathematica — Википедия http://ru.wikipedia.org/wiki/Mathematica