Федеральное агентство морского и речного транспорта
Федеральное государственное образовательное учреждение
высшего профессионального образования
«Волжская государственная академия водного транспорта»

Кафедра информатики, систем управления и телекоммуникаций

РЕФЕРАТ

СИСТЕМА КОМПЬЮТЕРНОЙ АЛГЕБРЫ MATHEMATICA

Выполнил: А.С. Пудов

Проверил: Ю.С. Федосенко

Нижний Новгород
2016

Содержание

Введение

Система компьютерной алгебры (**СКА**, англ. *computer algebra system, CAS*) — это прикладная программа для символьных вычислений, то есть выполнения преобразований и работы с математическими выражениями в аналитической (символьной) форме.

СКА появились в начале 1960-х и поэтапно развивались, в основном, в двух направлениях: теоретическая физика и создание искусственного интеллекта.

Первым успешным примером была новаторская работа Мартина Велтмана (позднее удостоенная Нобелевской премии по физике), который в 1963 создал программу для символьных вычислений (для нужд физики высоких энергий), которая была названа Schoonschip.

Используя LISP, Карл Энгельман в 1964 создал MATHLAB в рамках проекта MITRE (по исследованию искусственного интеллекта). Позже MATHLAB стал доступным в университетах для пользователей мейнфреймов PDP-6 и PDP-10 с такими ОС как TOPS-10 или TENEX. Сейчас он может быть всё ещё запущен на SIMH эмуляциях PDP-10. MATHLAB («mathematical laboratory») не стоит путать с MATLAB («matrix laboratory»), системой для численных расчётов, созданной 15 лет спустя в университете Нью-Мехико.

Начиная с конца 1960-х первое поколение СКА включало в себя системы:

- MACSYMA (Джоэл Мозес),
- MATHLAB (Массачусетский технологический институт),
- SCRATCHPAD (Ричард Дженкс, IBM),
- REDUCE (Тони Хирн),
- SAC-I, позже SACLIB (Джорж Коллинз),
- MUMATH для микропроцессоров (Дэвид Стоутмайер) и его продолжатель
- DERIVE.

Эти системы были способны выполнять символьные вычисления: интегрирование, дифференцирование, факторизация.

Ко второму поколению, в котором стал применяться более современный графический интерфейс пользователя, относятся Maple (Кейт Геддес и Гастон Гоннет, университет Уотерлу, 1985 год) и Mathematica (Стивен Вольфрам), которые широко используются математиками, учёными и инженерами. Бесплатные альтернативы — Sage, Maxima, Reduce.

В 1987 Hewlett-Packard представила первый карманный аналитический калькулятор (HP-28), и в нём впервые для калькуляторов были реализованы организация алгебраических выражений, дифференциирование, ограниченное аналитическое интегрирование, разложение в ряд Тейлора и поиск решений алгебраических уравнений.
Компания Texas Instruments в 1995 году выпустила калькулятор TI-92 с революционными на тот момент расширениями CAS на основе программного обеспечения Derive. Этот калькулятор и последовавшие за ним, в том числе TI-89 и серии TI-Nspire CAS, выпущенный в 2007 году, продемонстрировали возможность создания сравнительно компактных и недорогих систем компьютерной алгебры.

В третьем поколении стал применяться категориальный подход и операторные вычисления:

- AXIOM, последователь SCRATCHPAD (NAG),
- MAGMA (Джон Кэннон, Сиднейский университет),
- MUPAD (Бенно Фуксштейнер, университет города Падерборн).

На 2012 год исследования в области систем компьютерной алгебры продолжаются в трёх направлениях: возможности по решению всё более широких задач, простота использования и скорость работы.

Введение в Mathematica

Mathematica — система компьютерной алгебры, широко используемая в научных, инженерных, математических и компьютерных областях. Изначально система была разработана Стивеном Вольфрамом, впоследствии — компанией Wolfram Research.

Аналитические возможности

Основные аналитические возможности:

- решение систем полиномиальных и тригонометрических уравнений и неравенств, а также трансцендентных уравнений, сводящихся к ним;
- решение рекуррентных уравнений;
- упрощение выражений;
- нахождение пределов;
- интегрирование и дифференцирование функций;
- нахождение конечных и бесконечных сумм и произведений;
- решение дифференциальных уравнений и уравнений в частных производных;
- преобразования Фурье и Лапласа, а также Z-преобразование;
- преобразование функции в ряд Тейлора, операции с рядами Тейлора: сложение, умножение, композиция, получение обратной функции;
- вейвлет-анализ.

Система также осуществляет численные расчёты: определяет значения функций (в том числе специальных)) с произвольной точностью, осуществляет полиномиальную интерполяцию функции от произвольного числа аргументов по набору известных значений, рассчитывает вероятности.

Теоретико-числовые возможности — определение простого числа по его порядковому номеру, определение количества простых чисел, не превосходящих данное; дискретное

преобразование Фурье; разложение числа на простые множители, нахождение НОД и НОК.

Также в систему заложены линейно-алгебраические возможности — работа с матрицами (сложение, умножение, нахождение обратной матрицы, умножение на вектор, вычисление экспоненты, взятие определителя), поиск собственных значений и собственных векторов.

Система результаты представляет как в алфавитно-цифровой форме, так и в виде графиков. В частности, реализовано построение графиков функций, в том числе параметрических кривых и поверхностей; построение геометрических фигур (ломаных, кругов, прямоугольников и других); построение и манипулирование графами. Кроме того, реализовано воспроизведение звука, график которого задаётся аналитической функцией или набором точек.

Язык программирования Wolfram

Wolfram — это интерпретируемый язык функционального программирования. Можно сказать, что система Mathematica написана на языке Wolfram , хотя некоторые функции, особенно относящиеся к линейной алгебре, в целях оптимизации были написаны на языке Си.

Wolfram поддерживает и процедурное программирование с применением стандартных операторов управления выполнением программы (циклы и условные переходы), и объектно-ориентированный подход. Wolfram допускает отложенные вычисления. Также в системе Mathematica можно задавать правила работы с теми или иными выражениями.

Одним из базовых принципов встроенного языка Mathematica является представление любых сущностей в виде списков. Например, сумма чисел — это список с головным элементом `Plus`. Операции `Map` и `Apply`позволяют заменять головные элементы списков и применять заданное выражение к каждому элементу списка.

Элементы синтаксиса

Комментарии, которые не могут быть вложенными	(* ... *)
Регистрозависимость	да
Регулярное выражение идентификатора переменной	[_a-zA-Z][_a-zA-Z0-9]*
Регулярное выражение идентификатора функции	[_a-zA-Z][_a-zA-Z0-9]*
Присваивание значения переменной	a=b, a:=b(отложенное или ленивое присвоение)
Объявление переменной	(происходит в месте ее первого использования)

Объявление переменной с присваиванием значения	<varname> = <value>;
Группировка выражений	(...)
Блок	[...]
Равенство	==
Неравенство	!=
Сравнение	< > >= <=
Определение функции	f[x_, y_, z_] := (x + y + z)
Вызов функции	f[x, y, z]
Вызов функции без параметров	f[]
Последовательность	;
Если - то	If[condition, trueBlock]
Если - то - иначе	If[condition, trueBlock, falseBlock]
Бесконечный цикл	While[1<2, loopBody]
Цикл с предусловием	While[condition, loopBody]
Цикл for - next для диапазона целых чисел с инкрементом на 1	For[i = 0, i < 10, i++, loopBody]
Цикл for - next для диапазона целых чисел с декрементом на 1	For[i = 10, i > 0, i--, loopBody]

Методы программирования

Такие мощные системы, как Mathematica, предназначены, в основном, для решения математических задач без их программирования большинством пользователей. Однако это вовсе не означает, что Mathematica не является языком (или системой) программирования и не позволяет при необходимости программировать решение простых или сложных задач, для которых имеющихся встроенных функций и даже пакетов расширений оказывается недостаточно или которые требуют для реализации своих алгоритмов применения типовых программных средств, присущих обычным языкам программирования. Все обстоит совсем иначе.

Фактически, основой системы Mathematica является проблемно-ориентированный на математические расчеты язык программирования сверхвысокого уровня. По своим возможностям этот язык намного превосходит обычные универсальные языки программирования, такие как Фортран, Бейсик, Паскаль или С. Важно подчеркнуть, что здесь речь идет о языке программирования системы Mathematica, а не о языке реализации самой системы. Языком реализации является универсальный язык программирования С++, показавший свою высокую эффективность в качестве языка системного программирования.

Как и всякий язык программирования, входной язык системы Mathematica содержит операторы, функции и управляющие стриктуры. Основные операторы и функции этого языка и относящиеся к ним опции мы фактически уже рассмотрели. Набор описанных ранее типовых операторов и функций характерен для большинства современных языков программирования. Мощь системы Mathematica как средства программирования решения математических задач обусловлена необычно большим (в сравнении с обычными языками программирования) набором функций, среди которых немало таких, которые реализуют сложные и практически полезные математические преобразования и современные вычислительные методы (как численные, так и аналитические).

Число этих функций только в ядре и библиотеках приближается к тысяче. Среди них такие операции, как символьное и численное дифференцирование и интегрирование, вычисление пределов функций, вычисление специальных математических функций и т. д. — словом, реализации именно тех средств, для создания которых на обычных языках программирования приходится составлять отдельные, подчас довольно сложные программы. Почти столько же новых функций (или модернизированных старых) содержат пакеты расширения (Add-on Packages).

Язык программирования системы Mathematica трудно отнести к какому-либо конкретному типу. Можно разве что сказать, что он является типичным интерпретатором и не предназначен для создания исполняемых файлов. Впрочем, для отдельных выражений этот язык может осуществлять компиляцию с помощью функции Compile, что полезно при необходимости увеличения скорости счета.

Этот язык вобрал в себя лучшие средства ряда поколений языков программирования, таких как Бейсик, Фортран, Паскаль и С. Благодаря этому он позволяет легко реализовывать все известные типы (концепции) программирования: функциональное, структурное, объектно-ориентированное, математическое, логическое, рекурсивное и т. д. К примеру, вычисление таких функций, как факториал, в Mathematica можно запрограммировать в виде функции пользователя целым рядом способов:

```
f[n_] =n!
f[n_] =Gamma[n-l]
f [n_] =n*f [n-1] ;f [0]=l;f [1]=1;
f[n_] =Product[i/i,n]
f [n_] =Module[t=l,Do[t=t*i,i,n] ;t]
f [n_] =Module [ { t=l } , For [ i=l , i<=n , i++ ,
t*=i ] ; t]
f[n_] =Fold [Times,1, Range [n] ]
```

Все их можно проверить с помощью следующего теста:

```
{f[0],f[1],f[5],f[10]}
```

```
{1, 1, 120, 3628800}
```

Как отмечалось, внутреннее представление всех вычислений базируется на применении полных форм выражений, представленных функциями. И вообще, функциям в системе Mathematica принадлежит решающая роль. Таким образом, Mathematica. фактически, изначально реализует функциональный метод программирования — один из самых эффективных и надежных. А обилие логических операторов и функций позволяет полноценно реализовать и логический метод программирования. Множество операций преобразования выражений и функций позволяют осуществлять программирование на основе правил преобразования.

Надо также отметить, что язык системы позволяет разбивать программы на отдельные модули (блоки) и хранить эти модули в тексте документа или на диске Возможно создание полностью самостоятельных блоков — именованных процедур и функций с локальными переменными. Все это наряду с типовыми управляющими структурами позволяет реализовать структурное и модульное программирование.

Столь же естественно язык системы реализует объектно-ориентированное программирование. Оно базируется прежде всего на обобщенном понятии объекта и возможности создания множества связанных друг с другом объектов. В системе Mathematica каждая ячейка документа является объектом и порождается другими, предшествующими объектами. При этом содержанием объектов могут быть математические выражения, входные и выходные данные, графики и рисунки, звуки и т. д.

С понятием объекта тесно связаны три основных свойства, перечисленные ниже:

- инкапсуляция — объединение в одном объекте как данных, так и методов их обработки;
- наследование — означает, что каждый объект, производный от других объектов, наследует их свойства;
- полиформизм — свойство, позволяющее передать ряду объектов сообщение, которое будет обрабатываться каждым

объектом в соответствии с его индивидуальными особенностями.

Приведенный ниже пример объектно-ориентированного программирования дает три определения, ассоциированные с объектом h:

```
h/ : h [x_] +h [y_] : =hplus [x , y]
h/:p[h[x_],x]:=hp[x]
h/:f_[h[x_]] :=fh[f,x]
```

В принципе, язык программирования системы Mathematica специально создан для реализации любого из перечисленных подходов к программированию, а также ряда других — например, рекуррентного программирования, при котором очередной шаг вычислений базируется на данных, полученных на предыдущих шагах. Наглядным примером этого может служить вычисление факториала рекуррентным методом. Возможно также создание рекурсивных функций (с обращением к самим себе) и, соответственно, использование рекурсивного программирования. Оно, кстати, играет большую роль в осуществлении символьных преобразований.

Средства языка Mathematica позволяют осуществить и визуально-ориентированное программирование. Его смысл заключается в автоматической генерации программных модулей путем визуального выбора интуитивно понятного объекта — чаще всего путем щелчка на кнопке. Mathematica позволяет создавать палитры и панели с различными кнопками, позволяющими управлять программой или вводить новые программные объекты. Однако визуально-ориентированное программирование не является основным. В основном оно ориентировано на создание палитр пользователя с нужными ему функциями.
Поскольку алфавит языка программирования системы и набор операторов и функций уже были рассмотрены ранее, в этой главе нам остается рассмотреть лишь специфические средства языка и его управляющие структуры.

Примеры вычислений

После ввода коэффициентов определяем переменную y — квадратное уравнение с заданными коэффициентами. Так как x не определено, в уравнении оно останется обычной переменной (например, Print[y] выведет запись полученного уравнений c + b x + a x^2 с подставленными коэффициентами a, b и c). Функция Reduce вычисляет значения переменных, при котором указанное условие будет истинным. Условие в данном случае — квадратное уравнение, а переменная — x.

```
a = Input["Input a", 0];
b = Input["Input b", 0];
c = Input["Input c", 0];
y = a*x^2 + b*x + c;
Print[Reduce[y == 0]];
```

Факториал

Используется встроенная функция вычисления факториала !. Do — один из способов реализации циклов; выполняет первый аргумент для всех значений, заданных вторым аргументом, а именно: для всех i от 0 до 16 с шагом 1.

```
Do[Print[i, "! = ", i!] , {i, 0, 16, 1}]
```

Числа Фибоначчи

Print обязательно завершает вывод переносом строки, поэтому для того, чтобы вывести все числа Фибоначчи в одной строке, их нужно накопить в переменной msg и вывести ее. <> — оператор конкатенации; он работает только с явными строками, поэтому результат вызова Fibonacci нужно явно перевести в строку функцией ToString.

```
msg = "";
Do[msg = msg <> ToString[Fibonacci[i]] <> ", " ,
{i, 16} ];
Print[msg, "..."];
```

```
Hello, World!:
```

Функция Print выводит свои аргументы в основной выходной поток. Потоки могут быть вложенными, и для удобства копирования результатов лучше пользоваться для вывода одним потоком.

```
Print["Hello, World!"];
Hello, World!:
```

В этом случае создается временная строковая переменная. В конце строки нет символа “;”, и значение этой переменной выводится отдельным Out, что не всегда удобно.

```
"Hello, World!”
```

Факториал

Используется рекурсивное определение факториала. Обратите внимание на то, что при определении функции ее аргумент дополняется символом _.

```
Fact[n_] := If[n == 0, 1, Fact[n - 1]*n];
For[i = 0, i <= 16, i++, Print[i, "! = ",
Fact[i]]];
```

Числа Фибоначчи

Этот пример использует функцию Riffle, которая в данном случае перемежает элементы массива чисел Фибоначчи копиями строки “,”.

```
StringJoin[Riffle[Map[ToString, Table[Fibonacci[i],
{i,16}]], ", "]] <> “..."
```

Факториал

Символу f[x] ставится в соответствие список (List[…]) первых x натуральных чисел, генерируемый функцией Range[x], в которой

головная часть List заменяется на Times при помощи функции Apply[head, expr], записанной здесь как head@@expr.

Здесь использованы две парадигмы программирования, реализованные в Wolfram: во-первых, с помощью шаблонного выражения x_ число, вписанное в квадратные скобки, подставляется в соответствующее место справа от знака отсроченного присваивания := . Во-вторых, функциональная парадигма позволила избавить код от громоздких процедурных конструкций и придти к похожей на математическую записи.

```
fact[x_]:=Times@@Range[x];
```

Список литературы

1. Дьяконов В. П. Компьютерная математика. Теория и практика. — М., СПб: «Нолидж», «Питер», 1999,2001. — С. 1296.
2. Чарльз Генри Эдвардс , Дэвид Э. Пенни. Дифференциальные уравнения и проблема собственных значений: моделирование и вычисление с помощью Mathematica, Maple и MATLAB = Differential Equations and Boundary Value Problems: Computing and Modeling. — 3-е изд. — М.: «Вильямс», 2007.
3. Шмидский Яков Константинович. Mathematica 5. Самоучитель. Система символьных, графических и численных вычислений. — М.: «Диалектика», 2004. — С. 592.