МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Символьные вычисления в системе компьютерной математики Maxima

Учебное пособие для вузов

пособие для студентов, обучающихся по направлениям 01.03.01 Математика, 02.03.01 Математика и компьютерные науки, 01.03.04 Прикладная математика и по специальности 01.05.01 Фундаментальная математика и механика

Воронеж
2015

Что такое символьные вычисления

Что представляют эти самые символьные или, как их еще называют, аналитические вычисления, в отличие от численных расчетов. Компьютеры, как известно, оперируют с числами (целыми и с плавающей запятой). К примеру, решения уравнения $x^2 = 2x + 1$ можно получить как −0.41421356 и 2.41421356, а $3x = 1$ — как 0.33333333. А ведь хотелось бы увидеть не приближенную цифровую запись, а точную величину, т. е. $1\pm\sqrt{2}$ в первом случае и 1/3 во втором. С этого простейшего примера и начинается разница между численными и символьными вычислениями. Но кроме этого, есть еще задачи, которые вообще невозможно решить численно. Например, параметрические уравнения, где в виде решения нужно выразить неизвестное через параметр; или нахождение производной от функции; да практически любую достаточно общую задачу можно решить только в символьном виде. Поэтому неудивительно, что и для такого класса задач появились компьютерные программы, оперирующие не только числами, а почти любыми математическими объектами, от векторов до тензоров, от функций до дифференциальных уравнений и т.д.

Среди математического ПО для аналитических (символьных) вычислений наиболее широко известно коммерческое (*Maple*, *Mathematica*); это очень мощный инструмент для ученого или преподавателя, аспиранта или студента, позволяющий автоматизировать наиболее рутинную и требующую повышенного внимания часть работы, оперирующий при этом аналитической записью данных, т. е. фактически математическими формулами. Такую программу можно назвать средой программирования, с той разницей, что в качестве элементов языка программирования выступают привычные человеку математические обозначения. История проекта, известного ныне под именем Maxima, началась еще в конце 60-х годов в легендарном MIT (Massachusetts Institute of Technology — Массачусетский Технологический институт), когда в рамках существовавшего в те годы большого проекта MAC началась работа над программой символьных вычислений, которая получила имя Macsyma (от MAC Symbolic MAnipulation). Архитектура системы была разработана к июлю 1968 г., непосредственно программирование началось в июле 1969. в качестве языка для разработки системы был выбран Lisp, и история показала, насколько это был правильный выбор: из существующих в то время языков программирования он единственный продолжает развиваться и сейчас — спустя почти полвека после старта проекта. Принципы, положенные в основу проекта, позднее были заимствованы наиболее активно развивающимися ныне коммерческими программами — Mathematica и Maple; таким образом, Macsyma фактически стала родоначальником всего направления программ символьной математики. Естественно, Macsyma была закрытым коммерческим проектом; его финансировали государственные и частные организации, среди которых были вошедшее в историю ARPA (Advanced Research Projects Agency; помните ARPAnet — предок интернета?), Энергетический и Оборонный Департаменты США (Departments of Energy & Defence, DOE and DOD). Проект активно развивался, а организации, контролирующие его, менялись не раз, как это всегда бывает с долгоживущими закрытыми проектами. в 1982 году профессор Уильям Шелтер (William Schelter) начал разрабатывать свою версию на основе этого же кода, под названием Maxima. в 1998 году Шелтеру удалось получить от DOE права на публикацию кода по лицензии GPL. Первоначальный проект Macsyma прекратил свое существование в 1999 году. Уильям Шелтер продолжал заниматься разработкой Maxima вплоть до своей смерти в 2001 году. Но, что характерно для открытого ПО, проект не умер вместе со своим автором и куратором. Сейчас проект продолжает активно развиваться, и участие в нем является лучшей визитной карточкой для математиков и программистов всего мира. На данный момент Maxima выпускается под две платформы: Unix-совместимые системы, т. е. Linux и *BSD, и MS Windows.

команды. Команда в Максиме — это любая комбинация математических выражений и встроенных функций, завершенная, в простейшем случае, точкой с запятой. После ввода команды и нажатия «Enter» (в более поздних версиях, например 5.31.2, ввод данных осуществляется с помощью комбинации клавиш «Shift»+«Enter») Maxima выведет результат и будет ожидать следующей команды:

```
(%i1)  (1/2+1/3+1/4)/(1/5+1/6+1/8);

(%o1)  130/59
```

Как видите, каждая ячейка имеет свою метку; эта метка — заключенное в скобки имя ячейки. Каждый ввод и вывод помечаются системой и затем могут быть использованы снова. Символ (%i1) используется для обозначения команд, введенных пользователем, а (%o1) - при выводе результатов вычислений. Ячейки ввода именуются как %i с номером (i от *input* — ввод), ячейки вывода — как %o с соответствующим номером (o от *output* — вывод). Со знака % начинаются все встроенные служебные имена: чтобы, с одной стороны сделать их достаточно короткими и удобными в использовании, а с другой — избежать возможных накладок с пользовательскими именами, которые тоже часто удобно делать короткими. Благодаря такому единообразию вам не придется запоминать, как часто бывает в других системах, какие из таких коротких и удобных имен зарезервированы программой, а какие вы можете использовать для своих нужд. К примеру, внутренними именами **%e** и **%pi** обозначены общеизвестные математические постоянные; а через %c с номером обозначаются константы, используемые при интегрировании, для которых использование буквы «c» традиционно в математике. При вводе мы можем обращаться к любой из предыдущих ячеек по ее имени, подставляя его в любые выражения. Кроме того последняя ячейка вывода обозначается через %, а последняя ячейка ввода — через _. Это позволяет обращаться к последнему результату, не отвлекаясь на то, каков его номер.

```
(%i2)  %+47/59;

(%o2)  3
```

Здесь %+47/59 — то же самое, что %o1+47/59. Вывод результата вычисления не всегда нужен на экране; его можно заглушить, завершив команду символом $ вместо ;. Заглушенный результат при этом все равно вычисляется; как видите, в этом примере ячейки %o1 и %o2 доступны, хотя и не показаны (к ячейке %o2 обращение идет через символ %, смысл которого расшифрован выше):

```
(%i1)  sqrt(2)+3$
(%i2)  2*sqrt(2)+1$
(%i3)  %-%o1;

(%o3)  sqrt(2)-2
```

Каждую следующую команду не обязательно писать с новой строки; если ввести несколько команд в одну строчку, каждой из них все равно будет соответствовать свое имя ячейки. К примеру, здесь в строке после метки %i1 введены ячейки от %i1 до %i4; в ячейке %i3 используются %i1 и %i2 (обозначенная как _ — предыдущий ввод):

```
(%i3)  asin(1/2)$acos(1/2);%i1+_;%o1+%;

(%o4)  %pi/3

(%o5)  %pi/3+130/59

(%o6)  %pi/3+260/59
```

В wxMaxima и TeXmacs последнюю или единственную команду в строке можно не снабжать завершающим символом **;** — это сработает так же, как если бы она была **завершена ;** т. е. вывод заглушен не будет. Если вы выберете другой интерфейс, не забывайте ее добавлять.

Помимо использования имен ячеек, мы, естественно, можем и сами давать имена любым выражениям. По-другому можно сказать, что мы присваиваем значения переменным, с той разницей, что в виде значения такой переменной может выступать любое математическое выражение. Делается это с помощью двоеточия — знак равенства оставлен уравнениям, которые, учитывая общий математический контекст записи, проще и привычнее так читаются. И к тому же, так как основной конек Максимы — символьная запись и аналитические вычисления, уравнения достаточно часто используются. Например:

```
(%i1)  equation:x^3-x=0$

(%i2)  solve(equation);
(%o2)  [x=-1,x=1,x=0]
```

В каком-то смысле двоеточие даже нагляднее в таком контексте, чем знак равенства: это можно понимать так, что мы задаем некое обозначение, а затем через двоеточие расшифровываем, что именно оно обозначает. После того, как выражение поименовано, мы в любой момент можем вызвать его по имени:

```
(%i3)  diff(equation,x);
(%o3)  3x^2-1=0

(%i4) solve([x^2+6*x+9], [x]);

(%o4) [x=-3]
```

Таким образом: *для инициализации процесса вычислений следует ввести команду, затем символ ; (точка с запятой) и нажать клавишу* ***Enter (или Shift + Enter).*** *Если не требуется вывод полученной информации на экран, то вместо точки с запятой используется символ* ***$.*** *Обратиться к результату последней команды можно с помощью*

символа `%`*. Для повтора ранее введенной команды, скажем* `(%i2)`*, достаточно ввести два апострофа и затем метку требуемой команды, например,* `'(%i2) '`

Система Maxima обращает внимание на регистр введенных символов в именах встроенных констант и функций. Регистр букв важен при использовании переменных, например, Maxima считает x и X разными *переменными.*

Для стандартных математических констант используются следующие обозначения: `%e` для основания натуральных логарифмов, **%i** для мнимой единицы (квадратный корень из числа -1) и **%pi** для числа π.

Присваивание значения какой-либо переменной осуществляется с помощью знака **:** (двоеточие), а символ **=** (равно) используется при задании уравнений или подстановок.

```
(%i1) x:2;
(%o1)                                 2
(%i2) y:3;
(%o2)                                 3
(%i3) x + y;
(%o3)                                 5
```

Любое имя можно очистить от присвоенного ему выражения функцией kill(), и освободить занимаемую этим выражением память. Для этого нужно просто набрать kill(name), где name — имя уничтожаемого выражения; причем это может быть как имя, назначенное вами, так и любая ячейка ввода или вывода. Точно так же можно очистить разом всю память и освободить все имена, введя kill(all). В этом случае очистятся в том числе и все ячейки ввода-вывода, и их нумерация опять начнется с единицы. В дальнейшем, если по контексту будет иметься в виду логическое продолжение предыдущих строк ввода-вывода, я буду продолжать нумерацию (этим приемом я уже воспользовался выше). Когда же новый «сеанс» будет никак не связан с предыдущим, буду начинать нумерацию заново; это будет косвенным указанием сделать «kill(all)», если вы будете набирать примеры в Maxima, так как имена переменных и ячеек в таких «сеансах» могут повторяться. Для этого можно использовать основное меню. Кликаем ЛКМ по кнопке **Maxima** основного меню, переходим на строку **Очистить память** и еще раз кликнем ЛКМ. В этом меню можно также **Прервать** вычисления, **Перезапустить Maxima** и др. операции.

Функция **`kill`** *аннулирует присвоенные ранее значения переменных. Параметр* `all` *этой функции приводит к удалению значения всех переменных, включая метки* `(%i)` *и* `(%o)`.

```
(%i4) kill(x);
(%o4)                                done
(%i5) x + y;
(%o5)                              x + 3
(%i6) kill(all);
(%o0)                                done
(%i2) x + y;
(%o2)                              y + x
```

Для завершения работы с системой применяется функция **quit();**, а прерывание процесса вычислений осуществляется путем нажатия комбинации клавиш Ctrl- (после чего следует ввести :q для возврата в обычный режим работы), или команды в меню: **Maxima ->Прервать.** Можно использовать также команды: **Очистить память,** а также **Перезапустить Maxima.**

Работа с выражениями

При записи математических выражений могут использоваться четыре стандартные арифметрические операции (+, -, *, /) и операция возведения в степень (^, ^^ или **). Приоритет этих операций традиционен, для изменения порядка вычислений следует использовать круглые скобки. Кроме чисел выражения могут содержать результаты вычислений математических функций. Аргументы функций указываются в круглых скобках, например, запись sqrt(5) означает корень квадратный из числа 5. Если в результате расчета получается дробное выражение, то оно и выводится в виде обыкновенной дроби. Иррациональные числа, входящие в выражение, представляются в символьном виде.

Примеры работы программы с арифметическими операциями:

Сложение и умножение коммутирующие операции. Вычитание a- b представлено в Максима как сложение a+ (-b). Деление a/ b представлено в Максима как умножение, a * b ^ (-1).

```
(%i1) c + g + d + a + b + e + f;
(%o1)                g + f + e + d + c + b + a
(%i2) [op (%), args (%)];
(%o2)              [+, [g, f, e, d, c, b, a]]
(%i3) c * g * d * a * b * e * f;
(%o3)                       a b c d e f g
(%i4) [op (%), args (%)];
(%o4)                [*, [a, b, c, d, e, f, g]]
(%i5) apply ("+", [a, 8, x, 2, 9, x, x, a]);
(%o5)                       3 x + 2 a + 19
(%i6) apply ("*", [a, 8, x, 2, 9, x, x, a]);
                                    2  3
(%o6)                         144 a  x
```

op (операция), args(аргументы), apply(применить).

Деление и возведение в степень не коммутирующие операции. Операции обозначаются как "/" и "^".

```
(%i1) [a / b, a ^ b];
                                 a   b
(%o1)                           [-, a ]
                                 b
(%i2) [map (op, %), map (args, %)];
(%o2)                [[/, ^], [[a, b], [a, b]]]
(%i3) [apply ("/", [a, b]), apply ("^", [a, b])];
                                 a   b
(%o3)                           [-, a ]
                                 b
```

```
(%i1) 17 + b - (1/2)*29 + 11^(2/4);
                                          5
(%o1)                    b + sqrt(11) + -
                                          2
(%i2) [17 + 29, 17 + 29.0, 17 + 29b0];
(%o2)                    [46, 46.0, 4.6b1]
```

Для раскрытия скобок используется функция **expand**. Команда **ev** позволит получить численное значение выражения. Ее первый аргумент есть вычисляемое выражение, а второй - опция **numer**. Напомним, что символ **%** означает результат предыдущего вычисления.

```
(%i3) 1/100+1/101;

(%o3) 201/10100

(%i4) (1+sqrt(2))^5;

(%o4) (sqrt(2)+1)^5

(%i5) expand(%i14);

(%o5) 29*sqrt(2)+41

(%i6) ev(%i15,numer);

(%o6) 82.01219330881975
```

Допускается более удобная форма функции **ev,** требующая указания только ее аргументов:

```
(%i7) 29*sqrt(2) + 41, numer;
(%o7)                        82.01219330881976
```

По умолчанию результат содержит 16 значащих цифр. Для вывода числа в экспоненциальной форме используется функция **bfloat**:

```
(%i8) bfloat(%i7);
(%o8)                        ev(8.201219330881976b1,numer)
```

Запись mbn есть сокращенная форма выражения $m*10^n$.

Количество значащих цифр в представлении числа определяется специальной переменной **FPPREC.** Увеличение ее значения приводит к возрастанию точности результата, например,

```
(%i9) fpprec;
(%o9)                        16

(%i10) fpprec:100;
(%o10)                        100

(%i11) ''(%i7);
(%o11)                        8.20121933088197607657604923686248525#
03077530516721861648463104778270702443495 4#
83506838511144226151 55B1
```

Символ # в конце выводимой строки означает, что число не уместилось на одной строке и его оставшаяся часть переносится на следующую. В последнем примере мы использовали повторение ранее введенной команды "`(%i7)`;

Система Maxima может работать с числами произвольной длины и точности:

```
(%i12) 100!;
(%o12)                        933262154439441526816992388562667004#
907159682643816214685929638952175999932299 1#
560894146397615651828625369792082722375825 1#
185210916864000000000000000000000000
```