Алгебра и начала

ЧАСТЬ 4: Maxima способна прорешать половину Демидовича за десять минут. Не верите? Спросите у **Тихона Тарнавского**!

Из встроенного функционала *Maxima* в первую очередь стоит обратить внимание на несколько групп функций: работу с пределами, дифференцирование, интегрирование, поиск решений уравнений – как «просто», так и дифференциальных.

«Предельничаем»...

Собственно полноценных функций для нахождения предела существует в *Maxima* аж одна. Но зато какая! Она может принимать три различных варианта списка аргументов, и кроме того, на ее действие влияют еще и три флага. Но давайте по порядку. Зовут эту функции вполне соответственно ее действию: limit; и в самом стандартном варианте ее вызов выглядит как limit(выражение, переменная, точка), то есть то, что в математической записи выглядит как $\lim_{x\to a} f(x)$, в контексте *Maxima* запишется как limit(f(x), x, a):

```
(%i1) limit((x^2-1)/(2*x^2-x-1), x, 1);
(%o1) 2/3
(%i2) limit(((1+m*x)^n-(1+n*x)^m)/x^2, x, 0);
(%o2) -m*(m-n)*n/2
(%i3) limit(((x^n-a^n)-n*a^(n-1)*(x-a))/(x-a)^2, x, a);
(%o3) n*(a^n*n-a^n)/(2*a^2)
```

Maxima может искать пределы не только в конечных точках, но и на бесконечности. Среди стандартных обозначений программы существуют универсальные названия для разных бесконечностей: плюс-бесконечность записывается через inf (от слова «infinity», как нетрудно догадаться), минус-бесконечность – через minf (от «minus infinity»); для комплексных чисел бесконечность, как известно, одна, и она (комплексная бесконечность) обозначается полным словом infinity. При работе с пределами все три обозначения могут как использоваться при вводе, так и возникать в виде найденного значения предела; отдельно здесь надо отметить один момент касательно работы с интерфейсом к *Maxima* в редакторе *TeXmacs*: символы inf и minf при выводе здесь отображаются в своей традиционной математической нотации, то есть как ∞ и $-\infty$; символ ∞ вместо inf можно, кроме того, использовать еще и при вводе.

```
(%i1) limit((sqrt(x)+x^(1/3)+x^(1/4))/sqrt(2*x+1), x, inf);
(%o1) 1/sqrt(2)
(%i2) limit(sqrt(x+sqrt(x+sqrt(x)))-x, x, inf);
(%o2) -inf
```

Второй вариант вызова функции limit() – это расширенная версия первого: limit(выражение, переменная, точка, направление), для поиска односторонних пределов. Для предела справа в качестве «направления» указывается plus, для предела слева – minus:

```
(%i1) limit(abs(sin(x))/x, x, 0, plus);
(%o1) 1
(%i2) limit(abs(sin(x))/x, x, 0, minus);
(%o2) -1
```

Месяц назад Мы занимались упрощением выражений.

Пределы справа и слева еще иногда называют соответственно пределами сверху и снизу. Хотя правильнее в таком случае говорить полностью: «предел при **x**, стремящемся к **a** сверху», в том числе чтобы не создавать путаницы с верхним и нижним пределами, которые суть совершенно другое.

Кроме упомянутых выше бесконечностей, на выходе возможно появление и еще двух обозначений, на случай, если заданный предел не существует: ind (от слова indefinite – «неопределенный») и und (от слова undefined – опять же «неопределенный»). В документации первое из этих обозначений описано как «indefinite but bounded» (не определен, но ограничен), что дает предположить, что функция, не имеющая предела, при этом ограничена либо в окрестности предельной точки, либо на всей прямой. Какое из этих предположений имелось в виду, мне так и не удалось понять, потому как на практике ни одно из них не соответствует действительности. Вывода «ind» мне не удалось добиться ни на одной функции, радикально отличающейся от «канонической» (в том смысле, что фигурирующей в стандартном примере из комплекта) функции sin(1/x) [у нас аналогичный вывод получился и для (–1)^x – прим. ред.].

```
(%i1) limit(sin(1/x), x, 0); limit(tan(1/x), x, 0);
(%o1) ind
(%o2) und
```

Здесь все правильно, tan(1/x) не ограничена в окрестности нуля. А вот дальше начинаются чудеса:

```
(%i3) limit(a: 1/(1-exp(1/x)), x, 0);
(%o3) und
(%i4) limit(a, x, 0, plus); limit(a, x, 0, minus); limit(a, x, inf); limit(a, x, minf);
(%o4) 0
(%o5) 1
(%o6) -inf
(%o7) inf
(%i8) limit(a: atan(exp(1/x)), x, 0);
(%o8) und
(%i9) limit(a, x, 0, plus); limit(a, x, 0, minus); limit(a, x, inf); limit(a, x, minf);
(%o9) %pi/2
(%o10) 0
(%o11) %pi/4
(%o12) %pi/4
```

Как видите, первая функция имеет конечные односторонние пределы в нуле, а вторая ограничена вообще на всей оси – и тем не менее... Но это, думаю, не столь критично: главное, что наличие любого из этих символов в качестве вывода дает нам понять, что искомого предела не существует.

Функция limit() в третьем варианте – limit(выражение) – предназначена уже не для поиска собственно пределов, а для упрощения выражений, содержащих символы inf и minf:

```
(%i1) limit(1/inf);
(%o1) 0
```

Выражения такого рода могут возникать, к примеру, при подстановках в формулы результатов вычисления каких-то других пределов или интегралов.

Такая способность – принимать различные списки аргументов – не является в *Maxima* чем-то особенным; она свойственна очень многим встроенным функциям, как и различное действие в зависимости от значений разнообразных переключателей. Это достаточно удобно: не нужно запоминать много разных имен функций (для поиска пределов, к примеру, используется исключительно функция limit); для вычисления производных, в том числе и частных, – функция diff (с которой мы уже бегло ознакомились в первой статье и сейчас продолжим это знакомство); для нахождения интегралов, как определенных, так и неопределенных – функция integrate (с которой мы тоже сегодня познакомимся). Имена наиболее часто используемых функций запомнить несложно, а о дополнительных ключах или флагах, в случае чего, можно прочитать во встроенной справке, набрав **? имя-функции**.

Об этих самых ключах к функции limit и осталось рассказать. Первый ключ называется lhospitallim и задает максимальное количество применений правила Лопиталя; название ключа и происходит от фамилии ученого, давшей название самому правилу, которая в оригинале пишется как L'Hospital. Напомню, правило это гласит, что в случае неопределенности вида 0/0 или ∞/∞ можно продифференцировать числитель и знаменатель – и предел от этого не изменится. Ограничитель количества применений этого правила нужен для того, чтобы избежать зацикливаний, которые могут случиться для бесконечно дифференцируемых функций, у которых в данной точке равны нулю либо бесконечности все производные. По умолчанию значение lhospitallim равно четырем, и мне не удалось сходу придумать пример, когда этого не хватает – ведь функция поиска предела использует не только правило Лопиталя, но и другие соотношения; и для всех задан- »

ных мною соотношений двух функций с корнями выше четвертого порядка в искомой точке предел был успешно найден и при умолчательном значении.

Второй ключ к функции limit – это флаг limsubst, который, будучи выставлен в true, позволяет этой функции производить подстановки внутрь неизвестных выражений. По умолчанию этот флаг равен false, что исключает ошибки вроде такой:

```
(%i1) limit(f(x)/f(x+a),x,inf),limsubst:true
(%o1) 1
(%i2) f(x):=sin(x)$"%i1
(%o3) und
```

И, наконец, последний дополнительный параметр – еще один флаг, по имени tlimswitch. По умолчанию он тоже выключен, а если его включить, функция limit будет, при невозможности найти предел другими способами, пытаться его найти путем разложения подпредельной функции в ряд Тейлора в окрестности заданной точки:

```
(%i1) limit(sqrt(x+sqrt(x+sqrt(x)))/sqrt(x+1),x,inf)
Quotient by a polynomial of higher degree
 -- an error.  Quitting.  To debug this try debugmode(true);
(%i2) limit(sqrt(x+sqrt(x+sqrt(x)))/sqrt(x+1),x,inf),tlimswitch:true
(%o2) 1
```

Но в случае поиска односторонних пределов, в тех точках, где они не равны между собой, то есть полного предела не существует, этим флагом нужно пользоваться с осторожностью: при его включении функция limit может вернуть в качестве полного предела один из односторонних:

```
(%i1) limit(sqrt(1/x+sqrt(1/x+sqrt(1/x)))-sqrt(1/x-sqrt(1/x+sqrt(1/x))),x,0),tlimswitch:true
(%o1) 1
```

Реально у этой функции в точке ноль только предел справа равен единице; а предел слева – нулю.

И последнее: почему я употребил в начале по отношению к функции limit() слово «полноценная». Потому что кроме нее существует еще одна «недофункция» – tlimit(); она представляет собой фактически просто-напросто вызов самой функции limit() с поднятым флагом tlimswitch, то есть пытается при необходимости разложить «подпредельную» функцию в ряд Тейлора вне зависимости от реального значения этого флага. Другими словами вызов tlimit(аргументы) полностью аналогичен записи limit(аргументы), tlimswitch:true; только чуть короче. И аргументы она может принимать точно такие же.

...дифференцируем и интегрируем.

О функции diff я кое-что уже рассказывал в первой статье (*см. LXF81*), и здесь это «кое-что» только напомню. В двух упомянутых тогда вариантах вызова эта функция принимала один либо два аргумента. С двумя, diff(выражение, переменная), она возвращает производную от «выражения» по заданной переменной; с одним, diff(выражение) – полный дифференциал заданного выражения. Другими словами, запись diff(f, x) равнозначна математическому обозначению df/dx, а diff(f) – df.

Но это еще не все. Кроме одного либо двух, эта функция может также принимать любое нечетное число агрументов вида diff(выражение, переменная, порядок, переменная, порядок, ...) и возвращает при этом производную либо смешанную частную производную от выражения заданных порядков по заданным переменным. К примеру, diff(f, x, 3) означает d^3f/dx^3, а diff(f, x, 1, y, 2, z, 1) – $d^4f/dxdy^2dz$. Единственный флаг, имеющий прямое отношение к самой функции diff – это флаг derivabbrev, который влияет на отображение производных в ячейках вывода *Maxima*. По умолчанию он равен ***false***, и производные обозначаются в виде дробей с буквой d; если же его выставить в ***true***, производные будут отображаться в сокращенном виде, с переменными дифференцирования записанными в виде индексов:

```
(%i1) diff(f(x)^2,x,4)
(%o1) 2*f(x)*('diff(f(x),x,4))+8*('diff(f(x),x))*('diff(f(x),x,3))+6*('diff(f(x),x,2))^2
(%i2) derivabbrev:true$"%i1
(%o3) 2*f(x)*f[x,4](x)+8*f[x](x)*f[x,3](x)+6*(f[x,2](x))^2
```

Кроме того, функция diff используется еще и для обозначения производных в дифференциальных уравнениях. Но об этом чуть позже, а сейчас перейдем к интегрированию.

Основная функция интегрирования называется, как я уже говорил, integrate и имеет два варианта вызова: для нахождения неопределенного и определенного интегралов. Первый выглядит как integrate(выражение, переменная), второй – как integrate(выражение, переменная, нижний-предел, верхний-предел):

```
(%i1) integrate(sqrt((exp(x)-1)/(exp(x)+1)),x)
(%o1) log(2*sqrt(exp(2*x)-1)+2*exp(x))+asin(exp(-x))
(%i2) integrate(cos(x)*cos(2*x)*cos(3*x),x)
(%o2) sin(6*x)/24+sin(4*x)/16+sin(2*x)/8+x/4
(%i3) integrate(x^(2*n-1)/(x^n+1),x)
(%o3) exp(n*log(x))/n-log(exp(n*log(x))+1)/n
(%i4) integrate(x^2*sqrt(a^2-x^2),x,0,a)
Is a positive, negative, or zero?p
(%o4) %pi*a^4/16
```

Вы, наверное, обратили внимание еще на один момент в ячейках %i4–%o4. Когда в выражении используется какой-либо независимый символ, результат, вообще говоря, может зависеть от значения этого символа. Если при этом о возможных значениях символа ничего не известно, то *Maxima* задаст вам один или несколько вопросов об этом значении, и решение будет искать в зависимости от ваших ответов на них. Так, в этом примере значение определенного интеграла напрямую зависит от знака параметра a:

```
(%i5) integrate(x^2*sqrt(a^2-x^2),x,0,a)
Is a positive, negative, or zero?n
(%o5) -%pi*a^4/16
```

Кроме обычных определенных интегралов *Maxima* умеет искать также и несобственные интегралы, то есть такие, у которых неограни-

чена либо область интегрирования, либо подынтегральная функция; и делается это все той же функцией integrate:

(%i1) $\text{integrate}\left(\frac{1}{1+e^x}, x, 1, \infty\right)$

(%o1) $\log(e+1)-1$

(%i2) $\text{integrate}\left(\frac{1}{\sqrt{x}}, x, 0, 1\right)$

(%o2) 2

В случае, если искомый интеграл не сходится, будет выдано сообщение об ошибке, говорящее о том, что интеграл расходящийся:

(%i3) $\text{integrate}\left(\frac{1}{x^2}, x, 0, 1\right)$

```
Integral is divergent
 -- an error.  Quitting.  To debug this try debugmode(true);
```

В случае, если интеграл не может быть найден, он либо целиком возвращается в несовершенном виде, либо упрощается частично и на выходе получается некоторая формула, включающая в несовершенном виде интеграл той части подынтегрального выражения, которую проинтегрировать не удалось:

(%i1) $\text{integrate}\left(\frac{1}{x^4-4x^3+2x^2-7x-4}, x\right)$

(%o1) $\frac{\log(x-4)}{73}-\frac{\int \frac{x^2+4x+18}{x^3+2x+1}\,\mathrm{d}x}{73}$

Кроме функций diff и integrate, в *Maxima* есть еще много разнообразных возможностей, связанных с производными и интегралами, в частности, функции для численного расчета значений определенных интегралов, а также инструменты, применимые при работе с дифференциальными и интегральными уравнениями. И разнообразны они настолько, что для того, чтобы рассмотреть подробно и с примерами их все, не хватило бы всего этого цикла. А более тезисно, хотя и на английском языке, они описаны в документации. Посему с этой темой на этом прекращаем и движемся дальше.

Решайте, сударь!

А дальше мы рассмотрим уравнения и их системы, и даже не столько рассмотрим, сколько порешим... то есть, конечно, порешаем. Уравнения и системы уравнений решаются в *Maxima* тоже одной и той же функцией, и тоже уже вам слегка знакомой по первой статье: это функция solve. Но прежде чем рассмотреть ее подробнее, нужно сказать пару слов о списках, или векторах, в *Maxima*; поскольку именно в виде списков solve возвращает корни, да и принимает параметры в случае решения системы уравнений, а не одного уравнения.

Синтаксис списков в *Maxima* весьма прост; это перечисление элементов в квадратных скобках: [элемент1, элемент2, ..., элементN]. Особенность – не в синтаксисе. Основное достоинство *Maxima*льных списков в том, что их элементами могут быть совершенно любые выражения: символы, арифметические выражения, вызовы функций, присвоения, уравнения, другие списки... Додумать можете сами. Поэтому списки и во встроенных функциях применяются достаточно широко. Функция solve в своем простейшем варианте, для решения одиночного уравнения, в качестве аргументов никаких списков, напомню, не принимает (а принимает либо уравнение и символ, относительно которого его надо решать, либо только уравнение, если символ в нем всего один). А вот в качестве результата она уже и в таком варианте возвращает список, состоящий из всех корней заданного уравнения:

(%i1) $\text{eq}: \frac{x+1}{x^2+1}=\frac{x^2}{x+2}\$\ \text{solve}(\text{eq})$

(%o2) $\left[x=-\frac{\sqrt{5}-1}{2}, x=\frac{\sqrt{5}+1}{2}, x=-\frac{\sqrt{7}\,i+1}{2}, x=\frac{\sqrt{7}\,i-1}{2}\right]$

Как видите, функция solve находит все комплексные корни уравнения, а не только действительные.

К элементу списка можно обратиться с помощью тех же квадратных скобок, указав в них номер элемента после имени списка. Напомню, что равенство, переданное в качестве дополнительного параметра функции ev, означает подстановку переменной в вычисляемое выражение. Вот так мы можем осуществить проверку решения, подставив корень из выданного списка в исходное уравнение:

(%i3) $\text{eq}, \%[1]$

(%o3) $\frac{1-\frac{\sqrt{5}-1}{2}}{\frac{\left(\sqrt{5}-1\right)^2}{4}+1}=\frac{\left(\sqrt{5}-1\right)^2}{4\left(2-\frac{\sqrt{5}-1}{2}\right)}$

(%i4) $\text{ratsimp}(\%)$

(%o4) $\frac{\sqrt{5}-3}{\sqrt{5}-5}=\frac{\sqrt{5}-3}{\sqrt{5}-5}$

Точно таким же образом можно обратиться и к любому другому элементу списка:

(%i5) $\text{eq}, \%o2[2], \text{ratsimp}; \text{eq}, \%o2[3], \text{ratsimp}; \text{eq}, \%o2[4], \text{ratsimp}$

(%o5) $\frac{\sqrt{5}+3}{\sqrt{5}+5}=\frac{\sqrt{5}+3}{\sqrt{5}+5}$
(%o6) $-1=-1$
(%o7) $-1=-1$

Вообще говоря, в качестве первого аргумента функции solve можно задавать не только уравнение, а вообще любое выражение. При этом «корни выражения» (не являющегося уравнением) ищутся в том самом смысле, в каком эта фраза понимается в математике: корни выражения – это те значения переменной, на которых выражение обращается в ноль. Возможность такой записи позволяет, к примеру, легко найти критические точки любой непрерывной функции (а заодно и вычислить значения функции в этих точках):

(%i1) $f: \frac{1+x-x^2}{1+x+x^2}\$$

(%i2) $\text{solve}(\text{diff}(f, x))$

(%o2) $[x=-2, x=0]$

(%i3) $f, \%[1]; f, \%\text{th}(2)[2]$

(%o3) $-\frac{5}{3}$

(%o4) 1

В этом примере есть еще два важных момента. Первый – функция %th(). Она, как видно из контекста, вызывается как %th(n) и возвращает n-ю с конца ячейку вывода. Это, так же как и обозначения % и _, удобно, чтобы не обращать внимания на номера ячеек, и кроме того, применимо в командных файлах *Maxima*, которые могут загружаться в том числе и прямо из интерактивной сессии (с помощью функции load) – и тогда просто заранее неизвестно, начиная с какой ячейки данный файл загружен. И второй момент: здесь проиллюстрировано, что в *Maxima* операция индексирования списка доступна не только по отношению к именам переменных, но и к вызовам функций; другими словами, если функция возвращает список значений, мы можем выбрать одно конкретное из них, написав его номер в квадратных скобках прямо после вызова функции.

Вернемся к функции solve. А именно, перейдем теперь к решению систем уравнений. Для этого существует такой вариант записи: solve([уравнение1, уравнение2, ...], [переменная1, переменная2, ...]); либо сокращенный, аналогично варианту для одиночного уравнения: если количество уравнений и количество неизвестных равны, список неизвестных можно не писать: solve([уравнение1, уравнение2, ...]) (не забудьте квадратные скобки, иначе *Maxima* примет его за вариант с одним уравнением).

```
(%i1) solve([x^2+y^2=2, x+y=1])
```

$$(\%o1)\quad \left[\left[y=-\frac{\sqrt{3}-1}{2}, x=\frac{\sqrt{3}+1}{2}\right], \left[y=\frac{\sqrt{3}+1}{2}, x=-\frac{\sqrt{3}-1}{2}\right]\right]$$

Здесь возвращается список из нескольких списков, каждый из которых соответствует одному решению системы. В качестве подстановок можно использовать как такие списки целиком (например, в данном контексте, %o1[1]), так и отдельные их элементы (например, %o1[1][1]).

В случае, когда уравнений меньше, чем неизвестных, solve поступит точно так же, как и в случае одного уравнения с несколькими символами: все неуказанные будет воспринимать как параметры:

```
(%i1) solve([x^2+y^2=a^2, x+y=2*a+1], [x, y])
```

$$(\%o1)\quad \left[\left[x=-\frac{\sqrt{-2a^2-4a-1}-2a-1}{2}, y=\frac{\sqrt{-2a^2-4a-1}+2a+1}{2}\right], \left[x=\frac{\sqrt{-2a^2-4a-1}+2a+1}{2}, y=-\frac{\sqrt{-2a^2-4a-1}-2a-1}{2}\right]\right]$$

Если solve не находит точных решений, она может, как и integrate, вернуть уравнение или систему уравнений в некотором упрощенном виде, а может и самостоятельно попытаться решить систему численно:

```
(%i1) eqs: [4*x^2-y^2=12, x*y-x=2]$
(%i2) solve(eqs)
```

(%o2) [[y = 2, x = 2], [y = −0.15356757100197, x = −1.733751846381093], [y = 3.60800322187\
0287 i + 0.076783785237878, x = −0.5202594388652 i − 0.13312403573587], [y = 0.07678378523\
7878 − 3.608003221870287 i, x = 0.5202594388652 i − 0.13312403573587]]

В таком случае, если вам все же нужны точные значения корней (в аналитической записи), либо если они не найдены даже в числах, можно попробовать решить уравнения по очереди, выражая одно неизвестное через другое:

```
(%i3) solve(eqs[2], y)
```

$$(\%o3)\quad \left[y=\frac{x+2}{x}\right]$$

```
(%i4) eqs[1], %
```

$$(\%o4)\quad 4x^2-\frac{(x+2)^2}{x^2}=12$$

```
(%i5) solve(%)
```

И подставляя в оставшиеся уравнения:

```
(%i5) solve(%)
```

$$(\%o5)\quad \left[x=\frac{7\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}}+\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)-\frac{2}{3}, x=\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)+\frac{7\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}}-\frac{2}{3}, x=\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}+\frac{7}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}}-\frac{2}{3}, x=2\right]$$

Теперь можем подставить обратно – и найти значения второй неизвестной, например, для первого и последнего корней из последнего списка:

```
(%i6) %o3[1], %[1]; %o3[1], %th(2)[4]
```

$$(\%o6)\quad y=\frac{\frac{7\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}}+\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)+\frac{4}{3}}{\frac{7\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)}{36\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}}+\left(\frac{\sqrt{139}}{24\sqrt{3}}-\frac{8}{27}\right)^{\frac{1}{3}}\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)-\frac{2}{3}}$$

$$(\%o7)\quad y=2$$

Функция solve имеет довольно большое количество различных переключателей, из которых может пригодиться в своем не-умолчательном значении в первую очередь один: solveradcan. Умолчание здесь равно ***false***, а выставив этот флаг в ***true***, мы заставим solve, помимо его умолчательного поведения, применять radcan – функцию по упрощению показательных, логарифмических и степенных (с рациональными степенями) функций. Это делает работу функции solve более медленной (потому по умолчанию этот режим и выключен), но в некоторых случаях может помочь разрешить проблемы, которые без этого ключа приведут к невозможности найти точное решение.

И снова дифференцируем и интегрируем.

Помимо «просто» уравнений, *Maxima* позволяет также решать и обыкновенные дифференциальные уравнения первого и второго порядка. Функций, непосредственно занимающихся решением таких уравнений, существует две. Первая из них занимается поиском частных решений линейных дифференциальных уравнений и систем таких уравнений; зовут ее desolve, от слов «differential equation solve». Эта функция принимает два аргумента, первый из которых – уравнение либо список уравнений, а второй – соответственно одна переменная или список переменных. Если не заданы значения функций и/или их производных в нуле, то в найденном решении они просто отображаются в виде f(0) или

$$\left.\frac{d}{dx}f(x)\right|_{x=0}$$

Задать эти значения позволяет функция atvalue(выражение, переменная = точка, значение); то есть, в данном случае atvalue(f(x), x=0, значение) или atvalue('diff(f(x)), x=0, значение). Производные в уравнениях и системах, решаемых с помощью этой функции, должны быть записаны непременно в виде 'diff(f(x), x), а не просто 'diff(f, x), а сами функции, соответственно, тоже в виде f(x), а не f – нужно продемонстрировать зависимость функции от ее аргумента.

```
(%i1) ['diff(f(x), x) = 'diff(g(x), x) + sin(x),
       'diff(g(x), x, 2) = 'diff(f(x), x) - cos(x)]
```

$$(\%o1)\quad \left[\frac{d}{dx}f(x)=\frac{d}{dx}g(x)+\sin x, \frac{d^2}{dx^2}g(x)=\frac{d}{dx}f(x)-\cos x\right]$$

```
(%i2) atvalue('diff(g(x), x), x=0, a)$ atvalue(f(x), x=0, 1)$
(%i4) desolve(%o1, [f(x), g(x)])
```

$$(\%o4)\quad [f(x)=a\,e^x-a+1, g(x)=\cos x+a\,e^x-a+g(0)-1]$$

И конечно же, точно так же как для обычных уравнений и систем, здесь мы тоже можем проверить решение с помощью подстановки, но

только надо еще дополнительно задать принудительное вычисление производных, так как в уравнениях они фигурируют в несовершенной форме:

(%i5) %o1, %, diff

(%o5) $[a\,e^x = a\,e^x,\ a\,e^x - \cos x = a\,e^x - \cos x]$

Вторая функция из этой группы называется ode2 и предназначена она для решения обыкновенных дифференциальных уравнений первого и второго порядка; ее название происходит от фразы «ordinary differential equations of 1st or 2nd order». Пишется она так: ode2(уравнение, зависимая-переменная, независимая-переменная). Здесь уже независимая переменная указывается в списке параметров функции явно, и потому обозначения вида y(x) не нужны: и функция, и переменная обозначаются просто одиночными буквами. Также в отличие от предыдущей функции, ode2 ищет не частное, а общее решение. Произвольная константа в решении уравнения первого порядка обозначена через %c; в решении уравнения второго порядка таких констант, естественно, две, и обозначаются они как %k1 и %k2.

(%i1) $x^2\,'\mathrm{diff}(y,x) + 3\,y\,x = \dfrac{\sin(x)}{x}$

(%o1) $x^2\left(\dfrac{d}{d\,x}\,y\right) + 3\,x\,y = \dfrac{\sin x}{x}$

(%i2) $\mathrm{ode2}(\%, y, x)$

(%o2) $y = \dfrac{\%c - \cos x}{x^3}$

(%i3) $'\mathrm{diff}(y,x,2) + y\,'\mathrm{diff}(y,x)^3 = 0$

(%o3) $\dfrac{d^2}{d\,x^2}\,y + y\left(\dfrac{d}{d\,x}\,y\right)^3 = 0$

(%i4) $\mathrm{ode2}(\%, y, x)$

(%o4) $\dfrac{y^3 + 6\,\%k1\,y}{6} = x + \%k2$

В дополнение к функции ode2 существуют три функции для поиска частных решений на основе полученных общих. Иначе говоря, эти функции, получая конкретные условия относительно значения функции-решения в заданной точке, находят исходя из этих значений соответствующие им величины интегральных констант. Одна из этих функций предназначена для обработки решения дифференциального уравнения первого порядка. Она называется ic1 (i от «initial value» – начальное значение; c от «constant» – константа; 1 от «1st order» – первого порядка) и принимает три аргумента: первый – само решение, в том виде, в котором его находит функция ode2; второй – значение независимой переменной (x-координаты), третий – значение функции (зависимой переменной, y) при этом значении x и возвращает частное решение, проходящее через точку с заданными координатами (x, y):

(%i5) $\mathrm{ic1}(\%o2, x = \pi, y = 0)$

(%o5) $y = -\dfrac{\cos x + 1}{x^3}$

И две функции работают с решениями уравнений второго порядка. Так как в общем решении уравнения второго порядка фигурируют две независимые константы, то эти функции задают уже по два условия для поиска частного решения. Первая функция выглядит как ic2(общее решение, x, функция-в-точке-x, производная-в-точке-x). Расшифровка названия аналогична предыдущей функции. Действует тоже аналогично ей, а в качестве второго условия задает значение производной в той же заданной точке:

(%i6) $\mathrm{ic2}(\%o4, x = 0, y = 0, '\mathrm{diff}(y,x) = 2)$

(%o6) $\dfrac{y^3 - 3\,y\,(y^2 - 1)}{6} = x$

(%i7) $\mathrm{ratsimp}(\%)$

(%o7) $-\dfrac{2\,y^3 - 3\,y}{6} = x$

И последняя функция называется bc2 (b от «boundary value» – граничное значение; а дальше все как в предыдущей). Ее аргументы: первым, как и в двух остальных вариантах, идет само общее решение, возвращенное функцией ode2; после него идут две пары значений: x0, y0, x1, y1, задающие две точки, через которые должен проходить график функции-решения:

(%i8) $\mathrm{bc2}(\%o4, x = 0, y = 1, x = 1, y = 3)$

(%o8) $\dfrac{y^3 - 10\,y}{6} = x - \dfrac{3}{2}$

Касательно дифференциальных уравнений все этими функциями и заканчивается, никаких дополнительных ключей к ним не предназначено.

В следующий раз мы поговорим о средствах построения графиков функций и о средствах автоматизации: условных операторах, циклах и управляющих условиях, налагаемых на различные символы и выражения. LXF

Через месяц Графики, управляющие структуры и циклы.