Тема №6 «Алгебраические уравнения и оптимизация»
Лекция 1 «Алгебраические уравнения. Основы и методы решения»

---

1. Одно уравнение с одним неизвестным

Рассмотрим одно алгебраическое уравнение с одним неизвестным *x*.

$$f(x) = 0, \ (1)$$

Например, sin(x)=0.

Для решения таких уравнений Mathcad имеет встроенную функцию **root**, которая, в зависимости от типа задачи, может включать либо два, либо четыре аргумента и, соответственно, работает несколько по-разному.

root(f(x),x);

root(f(x),x,a,b);

f (x) – скалярная функция, определяющая уравнение (1);

x – скалярная переменная, относительно которой решается уравнение;

a,b – границы интервала, внутри которого происходит поиск корня.

Первый тип функции **root** требует дополнительного задания начального значения (**guess value**) переменной *x*. Для этого нужно просто предварительно присвоить *x* некоторое число. Поиск корня будет производиться вблизи этого числа. Таким образом, присвоение начального значения требует априорной информации о примерной локализации корня.

Приведем пример решения очень простого уравнения sin(x)=0, корни которого известны заранее.

Листинг 1.1. Поиск корня нелинейного алгебраического уравнения:

$$x := 0.5$$

$$f(x) := \sin(x)$$

$$solution := root(f(x), x)$$

$$solution = -6.2 \times 10^{-7}$$

График функции f (x)=sin(x) и положение найденного корня показаны на рис. 1.1.

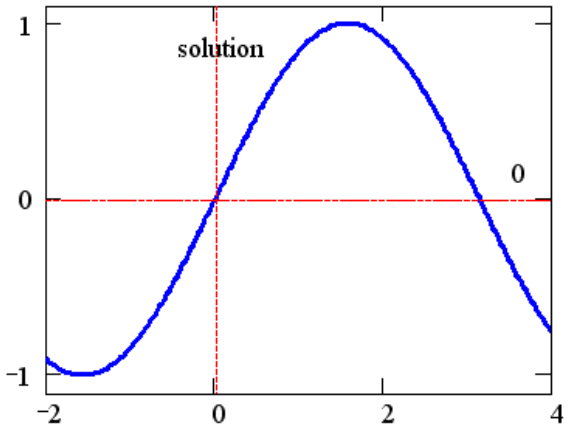

Рис. 1.1. Графическое решение уравнения sin(x)=0

Обратите внимание, что, хотя уравнение имеет бесконечное количество корней $x_n = np_i$ ($n=0,\pm1,\pm2,\ldots$), Mathcad находит (с заданной точностью) только один из них, x_0,

лежащий наиболее близко к *x=0.5.* Если задать другое начальное значение, например x=3, то решением будет другой корень уравнения $x_1=p_i$ и т. д. Таким образом, для поиска корня средствами Mathcad требуется его предварительная локализация. Это связано с особенностями выбранного численного метода, который называется методом секущих и состоит в следующем (рис. 1.2):

Начальное приближение принимается за 0-е приближение к корню: $x_0=x$.

Выбирается шаг h=TOLx и определяется первое приближение к корню $x_1=x_0+h$. Если x=0, то принимается h=TOL.

Через эти две точки проводится секущая – прямая линия, которая пересекает ось *x* в некоторой точке x_2. Эта точка принимается за второе приближение.

Новая секущая проводится через первую и вторую точки, тем самым определяя третье приближение, и т. д.

Если на каком-либо шаге оказывается, что уравнение выполнено, т. е. $|If(x)|<TOL$, то итерационный процесс прерывается, и *x* выдается в качестве решения.

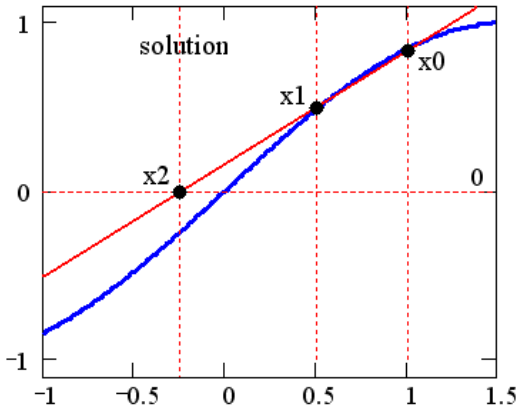

Рис. 1.2. Иллюстрация метода секущих

Результат, показанный на рис. 1.2, получен для погрешности вычислений, которой в целях иллюстративности предварительно присвоено значение TOL=0.5. Поэтому для поиска корня с такой невысокой точностью оказалось достаточно одной итерации. В вычислениях, приведенных в листинге 1.1, погрешность TOL=0.001 была установлена по умолчанию, и решение, выданное численным методом, лежало намного ближе к истинному положению корня *x=0.* Иными словами, чем меньше константа TOL, тем ближе к нулю будет значение *f (x)* в найденном корне, но тем больше времени будет затрачено вычислительным процессором Mathcad на его поиск.

Соответствующий пример можно найти в Быстрых шпаргалках, на странице Ресурсов Mathcad. Он расположен в разделе "**Solving Equations**" (Решение уравнений) и называется "**Effects of TOL on Solving Equations**" (Влияние константы TOL на решение уравнений).

Если уравнение неразрешимо, то при попытке найти его корень будет выдано сообщение об ошибке. Кроме того, к ошибке или выдаче неправильного корня может привести и попытка применить метод секущих в области локального максимума или минимума *f (x).* В этом случае секущая может иметь направление, близкое к горизонтальному, выводя точку следующего приближения далеко от предполагаемого положения корня. Для решения таких уравнений лучше применять другую встроенную функцию **Minerr**. Аналогичные проблемы могут возникнуть, если начальное приближение выбрано слишком далеко от настоящего решения или *f (x)* имеет особенности типа бесконечности.

Для решения уравнения с одним неизвестным применимы и градиентные методы, относящиеся в Mathcad к системам уравнений. Иногда удобнее задавать не начальное приближение к корню, а интервал *[a,b],* внутри которого корень заведомо находится. В этом случае следует использовать функцию **root** с четырьмя аргументами, а присваивать

начальное значение *x* не нужно, как показано в листинге 1.2. Поиск корня будет осуществлен в промежутке между *a* и *b* альтернативным численным методом (Риддера или Брента).

Листинг 1.2. Поиск корни алгебраического уравнения заданном интервале:

$$\text{solution} := \text{root}(\sin(x), x, -1, 1)$$

$$\text{solution} = 0$$

Обратите внимание, что явный вид функции *f (x)* может быть определен непосредственно в теле функции root.

Когда **root** имеет четыре аргумента, следует помнить о двух ее особенностях:

- внутри интервала *[a,b]* не должно находиться более одного корня, иначе будет найден один из них, заранее неизвестно, какой именно;
- значения *f (a)* и *f (b)* должны иметь разный знак, иначе будет выдано сообщение об ошибке.

Если уравнение не имеет действительных корней, но имеет мнимые, то их также можно найти. В листинге 1.3 приведен пример, в котором уравнение $x^2+1=0$, имеющее два чисто мнимых корня, решается два раза с разными начальными значениями. При задании начального значения 0.5 (первая строка листинга) численный метод отыскивает первый корень (отрицательную мнимую единицу – i), а при начальном значении – 0.5 (третья строка листинга) находится и второй корень (i).

Листинг 1.3. Поиск мнимого корня:

$$x := 0.5$$

$$\text{root}\left(x^2 + 1, x\right) = -i$$

$$x := -0.5$$

$$\text{root}\left(x^2 + 1, x\right) = i$$

Для решения этого уравнения второй вид функции **root** (с четырьмя, а не с двумя аргументами) неприменим, поскольку *f (x)* является положительноопределенной, и указать интервал, на границах которого она имела бы разный знак, невозможно.

Остается добавить, что *f (x)* может быть функцией не только *x*, а любого количества аргументов. Именно поэтому в самой функции **root** необходимо определить, относительно какого из аргументов следует решить уравнение. Эта возможность проиллюстрирована листингом 1.4 на примере функции двух переменных $f(x,y)=x^2-y^2+3$. В нем сначала решается уравнение *f(x,0)=0* относительно переменной x, а потом – другое уравнение *f (1,y) =0* относительно переменной *y*.

Листинг 1.4. Поиск корня уравнения, заданного функцией двух переменных:

$$f(x,y) := x^2 - y^2 + 3$$

$$x := 1$$

$$y := 0$$

$$root(f(x,y),x) = -1.732i$$

$$root(f(x,y),y) = 2$$

В первой строке листинга определяется функция *f (x,y),* во второй и третьей – значения, для которых будет производиться решение уравнения по y и x, соответственно.

В четвертой строке решено уравнение *f (x,0)=0*, а в последней – уравнение *f (1,y)=0*. Не забывайте при численном решении уравнений относительно одной из переменных предварительно определить значения остальных переменных. Иначе попытка вычислить уравнения приведет к появлению ошибки "This variable or function is not defined above", в данном случае говорящей о том, что другая переменная ранее не определена. Конечно, можно указать значение других переменных непосредственно внутри функции **root**, беспрепятственно удалив, например, вторую и третью строки листинга 8.4 и введя его последние строки в виде *root(f (x,0),x)= и root(f (1,y),y)=*, соответственно.

Корни полинома

Если функция *f (x)* является полиномом, то все его корни можно определить, используя встроенную функцию **polyroots(v)**, где **v** – вектор, составленный из коэффициентов полинома.

Поскольку полином *N*-й степени имеет ровно *N* корней (некоторые из них могут быть кратными), вектор v должен состоять из *N+I* элемента. Результатом действия функции poiyroots является вектор, составленный из *N* корней рассматриваемого полинома. При этом нет надобности вводить какое-либо начальное приближение, как для функции **root**. Пример поиска корней полинома четвертой степени иллюстрируется листингом 1.5.

Листинг 1.5. Поиск корня полинома:

$$v := (3 \quad -10 \quad 12 \quad -6 \quad 1)^T$$

$$polyroots(v) = \begin{pmatrix} 1 \\ 1 - 5.113i \times 10^{-6} \\ 1 + (5.113i \times 10)^{-6} \\ 3 \end{pmatrix}$$

Коэффициенты рассматриваемого в примере полинома

f (x) = (x-3) (x-1)^3 =x^4 -6x^3 + 12x^2 -10x+3

записаны в виде вектора в первой строке листинга. Первым в векторе должен идти свободный член полинома, вторым – коэффициент при x^1 и т. д. Соответственно, последним *n+1* элементом вектора должен быть коэффициент при старшей степени X^n.

Иногда исходный полином имеется не в развернутом виде, а, например, как произведение нескольких полиномов. В этом случае определить все его коэффициенты можно, выделив его и выбрав в меню **Symbolics** (Символика) пункт **Expand** (Разложить). В результате символьный процессор Mathcad сам преобразует полином в нужную форму, пользователю надо будет только корректно ввести ее в аргументы функции polyroots.

Во второй строке листинга 1.5 показано действие функции **poiyroots**. Обратите внимание, что численный метод вместо двух из трех действительных единичных корней (иными словами, кратного корня 1) выдает два мнимых числа. Однако малая мнимая часть этих корней находится в пределах погрешности, определяемой константой TOL, и не должна вводить пользователей в заблуждение. Просто нужно помнить, что корни полинома могут быть комплексными, и ошибка вычислений может сказываться как на действительной, так и на комплексной части искомого корня.

Для функции **poiyroots** можно выбрать один из двух численных методов – метод полиномов Лаггера (он установлен по умолчанию) или метод парной матрицы.

Для смены метода:

Вызовите контекстное меню, щелкнув правой кнопкой мыши на слове **polyroots.**

В верхней части контекстного меню выберите либо пункт **LaGuerre** (Лаггера), либо **Companion Matrix** (Парная матрица).

Щелкните вне действия функции **polyroots** – если включен режим автоматических вычислений, будет произведен пересчет корней полинома в соответствии с вновь выбранным методом.

Для того чтобы оставить за Mathcad выбор метода решения, установите флажок **AutoSelect** (Автоматический выбор), выбрав одноименный пункт в том же самом контекстном меню.

2. Системы уравнений

Рассмотрим решение системы *N* нелинейных уравнений с *M* неизвестными:

$$f_1(x_1,\ldots,x_м) = 0,$$
$$\ldots \qquad (2.1)$$
$$f_n(x_1,\ldots,x_м) = 0,$$

Здесь $f_1(x_1,\ldots,x_м),\ldots f_n(x_1,\ldots,x_м)$, – некоторые скалярные функции от скалярных переменных $x_м x_2/ \ldots /x_м$ и, возможно, от еще каких-либо переменных. Уравнений может быть как больше, так и меньше числа переменных. Заметим, что систему (2.1) можно формально переписать в виде:

$$f(x) = 0, \qquad (2.2)$$

где *x* – вектор, составленный из переменных x1,x2,…,xм, а f (x) – соответствующая векторная функция.

Для решения систем имеется специальный вычислительный блок, состоящий из трех частей, идущих последовательно друг за другом:

Given – ключевое слово;

система, записанная логическими операторами в виде равенств и, возможно, неравенств;

Find($x_1\ldots,x_м$) – встроенная функция для решения системы относительно переменных $x_x,\ldots, x_м$.

Вставлять логические операторы следует, пользуясь панелью инструментов **Boolean** (Булевы операторы). Если Вы предпочитаете ввод с клавиатуры, помните, что логический знак равенства вводится сочетанием клавиш **CTRL + =**. Блок **Given/Find** использует для поиска решения итерационные методы, поэтому, как и для функции **root**, требуется задать начальные значения для всех x1,…, xм. Сделать это необходимо до ключевого слова **Given**. Значение функции **Find** есть вектор, составленный из решения пс каждой переменной. Таким образом, число элементов вектора равно число аргументов **Find**.

В листинге 1.6. приведен пример решения системы двух уравнений.

Листинг 1.6. Решение системы уравнений:

$$f(x,y) := x^4 + y^2 - 3$$

$$g(x,y) := x + 2\cdot y$$

$$x := 1 \qquad y := 1$$

Given

$$f(x,y) = 0$$

$$g(x,y) = 0$$

$$v := \text{Find}(x,y)$$

$$v = \begin{pmatrix} 1.269 \\ -0.635 \end{pmatrix}$$

$$f(v_0, v_1) = -1.954 \times 10^{-7}$$

$$g(v_0, v_1) = 0$$

В первых двух строках листинга вводятся функции, которые определяют систему уравнений. Затем переменным x и y, относительно которых она будет решаться, присваиваются начальные значения. После этого следует ключевое слово **Given** и два логических оператора, выражающих рассматриваемую систему уравнений. Завершает вычислительный блок функция **Find**, значение которой присваивается вектору v. Следующая строка показывает содержание вектора v, т. е. решение системы. Первый элемент вектора есть первый аргумент функции **Find**, второй элемент – ее второй аргумент. В последних двух строках осуществлена проверка правильности решения уравнений.

Часто бывает очень полезно проверить точность решения уравнений, вычислив значения образующих их функций в найденных вычислительным процессором корнях, как это сделано в конце листинга 1.6.

Отметим, что уравнения можно определить непосредственно внутри вычислительного блока. Таким образом, можно не определять заранее функции *f (x,y)* и *g(x,y)*, как это сделано в первых двух строках листинга 1.6, а сразу написать:

Given

$x^4 + y^2 = 3$

$x + 2\,y = 0$

Такая форма представляет уравнения в более привычной и наглядной форме, особенно подходящей для документирования работы.

Графическая интерпретация рассмотренной системы представлена на рис. 1.3. Каждое из уравнений показано на плоскости XY графиком. Первое – сплошной кривой, второе – пунктиром. Поскольку второе уравнение линейное, то оно определяет на плоскости XY

прямую. Две точки пересечения кривых соответствуют одновременному выполнению обоих уравнений, т. е. искомым действительным корням системы. Как нетрудно убедиться, в листинге найдено только одно из двух решений – находящееся в правой нижней части графика. Чтобы отыскать и второе решение, следует повторить вычисления, изменив начальные значения так, чтобы они лежали ближе к другой точке пересечения графиков, например x=-1, y=-1.

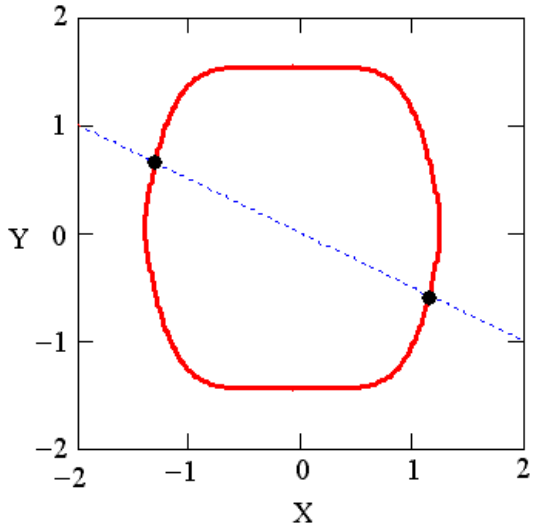

Рис. 1.3. Графическое решение системы двух уравнений

Пока мы рассмотрели пример системы из двух уравнений и таким же числом неизвестных, что встречается наиболее часто. Но число уравнений и неизвестных может и не совпадать. Более того, в вычислительный блок можно добавить дополнительные условия в виде неравенств. Например, введение ограничения на поиск только отрицательных значений x в рассмотренный выше листинг 1.6 приведет к нахождению другого решения, как это показано в листинге 1.7.

Листинг 1.7. Решение системы уравнений и неравенств:

$$x := 1 \qquad y := 1$$

Given

$$x^4 + y^2 = 3$$

$$x + 2 \cdot y^2 = 0$$

$$x < 0$$

$$\text{Find}(x, y) = \begin{pmatrix} -1.269 \\ 0.635 \end{pmatrix}$$

Обратите внимание, что, несмотря на те же начальные значения, что и в листинге 8.6, мы получили в листинге 1.7 другой корень. Это произошло именно благодаря введению дополнительного неравенства, которое определено в блоке **Given** в предпоследней строке листинга 1.7.

Если предпринять попытку решить несовместную систему, Mathcad выдаст сообщение об ошибке, гласящее, что ни одного решения не найдено, и предложение попробовать поменять начальные значения или значение погрешности.

Вычислительный блок использует константу **CTOL** в качестве погрешности выполнения уравнений, введенных после ключевого слова **Given**. Например, если CTOL=0.001, то уравнение *x=10* будет считаться выполненным и при x=10.001, и при

x=9.999. Другая константа **TOL** определяет условие прекращения итераций численным алгоритмом. Значение **CTOL** может быть задано пользователем так же как и TOL, например, CTOL: = 0.01. По умолчанию принято, что CTOL=TOL=0.001, но Вы по желанию можете переопределить их.

Особенную осторожность следует соблюдать при решении систем с числом неизвестных большим, чем число уравнений. Например, можно удалить одно из двух уравнений из рассмотренного нами листинга 1.6, попытавшись решить единственное уравнение *g(x,y)=0* с двумя неизвестными *x* и *y*. В такой постановке задача имеет бесконечное множество корней: для любого *x* и, соответственно, *y=-x/2* условие, определяющее единственное уравнение, выполнено. Однако, даже если корней бесконечно много, численный метод будет производить расчеты только до тех пор, пока логические выражения в вычислительном блоке не будут выполнены (в пределах погрешности). После этого итерации будут остановлены и выдано решение. В результате будет найдена всего одна пара значений *(x,y),* обнаруженная первой.

Вычислительным блоком с функцией **Find** можно найти и корень уравнения с одним неизвестным. Действие **Find** в этом случае совершенно аналогично уже рассмотренным в данном разделе примерам. Задача поиска корня рассматривается как решение системы, состоящей из одного уравнения. Единственным отличием будет скалярный, а не векторный тип числа, возвращаемого функцией **Find**. Пример решения уравнения из предыдущего раздела приведен в листинге 1.8.

Листинг 1.8. Поиск корня уравнения с одним неизвестным с помощью функции Find:

$$x := 0.5$$

Given

$$\sin(x) = 0$$

$$\text{Find}(x) = -3.814 \times 10^{-7}$$

В чем же отличие приведенного решения от листинга 1.1 с функцией **root**? Оно состоит в том, что одна и та же задача решена различными численными методами. В данном случае выбор метода не влияет на окончательный результат, но бывают ситуации, когда применение того или иного метода имеет решающее значение.

3. О численных методах решения систем уравнений

Если Вы решаете "хорошие" уравнения, как все те, которые были приведены в предыдущих разделах, то можете никогда не задумываться, как именно Mathcad ищет их корни. Однако даже в этом случае полезно представлять, что происходит "за кадром", т. е. какие действия совершаются в промежутке между введением необходимых условий после ключевого слова **Given** и получением результата после применения функции **Find**. Это важно хотя бы с позиций выбора начальных значений переменных перед вычислительным блоком. Рассмотрим в данном разделе некоторые особенности численных методов и возможности установки их различных параметров, которые предоставляет Mathcad.

Функция **Find** реализует градиентные численные методы. Покажем их основную идею на примере уравнения с одним неизвестным *f(x)=0* для функции *f (x)=x2 +5x+2,* график которой показан на рис. 1.4. Основная идея градиентных методов состоит в последовательных приближениях к истинному решению уравнения, которые вычисляются с помощью производной от *f (x).* Приведем наиболее простую форму алгоритма, называемого методом Ньютона:

За нулевую итерацию принимается введенное пользователем начальное значение x_0=x.

В точке xo методом конечных разностей вычисляется производная $f'(x_0)$.

Пользуясь разложением Тейлора, можно заменить *f (x)* в окрестности xo касательной – прямой линией $f(x)=f(x_0)+f(x_0)(x-x_0)$.

Определяется точка x^1, в которой прямая пересекает ось *x* (см. рис. 1.4).

Если f (x_1)<TOL, то итерации прерываются, и значение x1 выдается в качестве решения. В противном случае x_1 принимается за новую итерацию, и цикл повторяется: строится касательная к *f (x)* в точке x_1, определяется x_2 – точка ее пересечения с осью *x* и т. д.

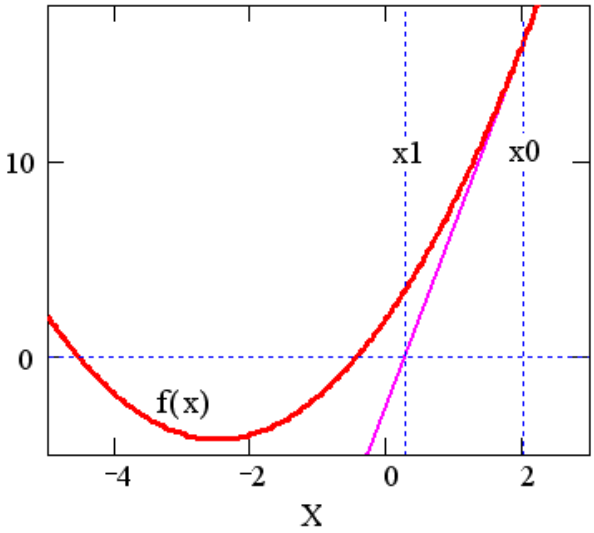

Рис. 1.4. Иллюстрация метода Ньютона

Модификация алгоритма Ньютона для решения системы нескольких уравнений заключается в линеаризации соответствующих функций многих переменных, т. е. аппроксимации их линейной зависимостью с помощью частных производных. Например, для нулевой итерации в случае системы двух уравнений используются выражения.

Чтобы отыскать точку, соответствующую каждой новой итерации, требуется приравнять оба равенства нулю, т. е. решить на каждом шаге полученную систему линейных уравнений.

Mathcad предлагает три различных вида градиентных методов. Чтобы поменять численный метод:

Щелкните правой кнопкой мыши на названии функции **Find**.

Наведите указатель мыши на пункт **Nonlinear** (Нелинейный) в контекстном меню.

В появившемся подменю (рис. 1.5) выберите один из трех методов: **Conjugate Gradient** (Сопряженных градиентов), **Quasi-Newton** (Квази-Ньютоновский) или **Levenberg-Marquardt** (Левенберга).

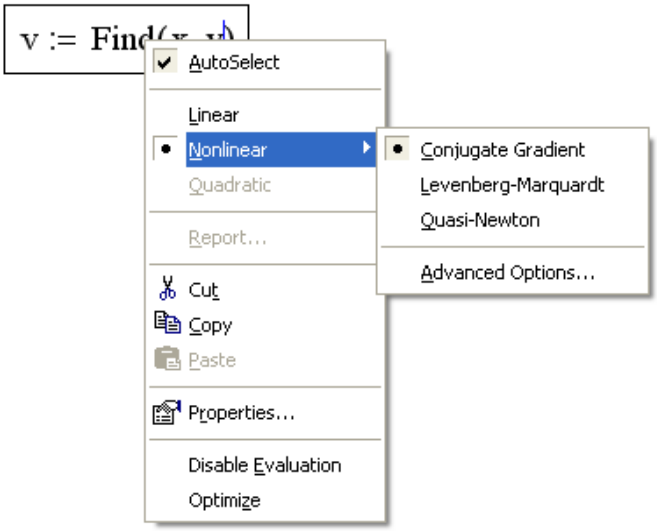

Рис. 1.5. Смена численного метода

Чтобы вернуть автоматический выбор типа численного метода, в контекстном меню надо выбрать пункт **AutoSelect** (Автоматический выбор). Если установлена опция автоматического выбора (о чем говорит флажок, установленный в пункте **AutoSelect**), то текущий тип численного метода можно узнать, вызвав то же самое подменю и посмотрев, который из них отмечен точкой. Два последних метода являются квази-Ньютоновскими, основная идея которых была рассмотрена выше. Первый из них, метод сопряженных градиентов, является двухшаговым – для поиска очередной итерации он использует как текущую, так и предыдущую итерации. Алгоритм Левенберга подробно описан в справочной системе Mathcad, а детальную информацию о методах Ньютона и сопряженных градиентов можно найти в большинстве книг по численным методам.

Помимо выбора самою метода, имеется возможность устанавливать их некоторые параметры. Для этого нужно вызвать с помощью того же контекстного меню диалоговое окно **Advanced Options** (Дополнительные параметры), выбрав в контекстном меню пункты **Nonlinear › Advanced options** (Нелинейный › Дополнительные параметры). В этом диалоговом окне (рис. 8.6.) имеется пять групп переключателей, по два в каждой.

В первой строке **Derivative estimation** (Аппроксимация производной) определяется метод вычисления производной **Forward** (Вперед) или **Central** (Центральная). Они соответствуют аппроксимации производной либо правой (двухточечная схема "вперед"), либо центральной (трехточечная симметричная схема) конечной разностью.

Обратите внимание, что вычисление производной в градиентных численных методах решения уравнений производится более экономичным способом, нежели при численном дифференцировании.

Во второй строке **Variable estimation** (Аппроксимация переменных) можно определить тип аппроксимации рядом Тейлора. Для рассмотренного нами в этом разделе случая аппроксимации касательной прямой линией выберите переключатель **Tangent** (Касательная), для более точной квадратичной аппроксимации (параболой) выберите **Quadratic** (Квадратичная).

Следующая группа переключателей **Linear variable check** (Проверка линейности) позволяет в специфических задачах сэкономить время вычислений. Если Вы уверены, что нелинейности всех функций, входящих в уравнение, мало сказываются на значениях всех их частных производных, то установите переключатель **Yes** (Да). В этом случае производные будут приняты равными константам и не будут вычисляться на каждом шаге.

С осторожностью изменяйте параметры численных методов. Пользуйтесь ими, когда решение не находится при выставленных по умолчанию параметрах или когда расчеты занимают очень продолжительное время.

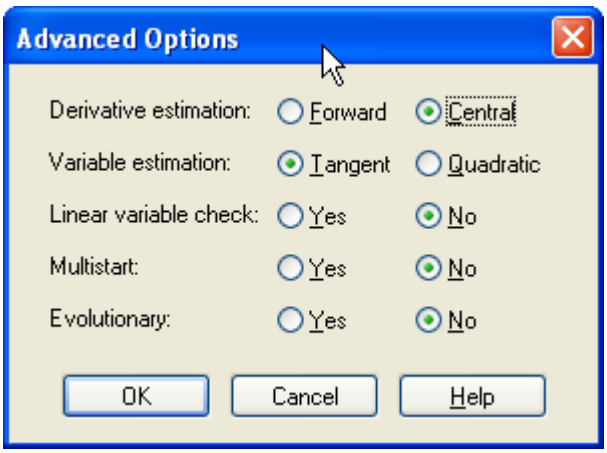

Рис. 1.6. Диалоговое окно Advanced Options

Пара переключателей **Multistart** (Сканирование) задает опцию поиска глобального или локального минимума или максимума. Если выставлен переключатель **Yes** (Да), Mathcad будет пытаться найти наиболее глубокий экстремум из области, близкой к начальному приближению. Эта опция предназначена, в основном, для настройки (тех же самых, градиентных) алгоритмов поиска экстремума, а не для решения алгебраических уравнений.

Наконец, последний переключатель **Evolutionary** (Эволюционный алгоритм), если установить его в положение **Yes** (Да), позволяет использовать модификацию численного метода для решения уравнений, определяемых не обязательно гладкими функциями Как мы убедились в этом разделе, все градиентные методы, реализованные в функции **Find**, требуют многократного вычисления производных. Если Вы работаете с достаточно гладкими функциями, то градиентные методы обеспечивают быстрый и надежный поиск корня. Для поиска корня недостаточно гладких функций одной переменной, следует либо выставить данную опцию функции **Find**, либо использовать метод секущих (функцию **root**). Помните, что правильный выбор численного метода и его параметров может помочь при решении нестандартной задачи, которая при стандартных установках может и не поддаваться решению.

4. Приближенное решение уравнений

Иногда приходится заменять задачу определения корней системы уравнений задачей поиска экстремума функции многих переменных. Например, когда невозможно найти решение с помощью функции **Find**, можно попытаться потребовать вместо точного выполнения уравнений условий минимизировать их невязку. Для этого следует в вычислительном блоке вместо функции **Find** использовать функцию **Minerr**, имеющую тот же самый набор параметров. Она также должна находиться в пределах вычислительного блока:

x_1: = C_1… $x_м$: =$C_м$ – начальные значения для неизвестных.

Given – ключевое слово.

Система алгебраических уравнений и неравенств, записанная логическими операторами.

Minerr (x_1,…,$x_м$) – приближенное решение системы относительно переменных x_1;…,$x_м$, минимизирующее невязку системы уравнений.

В функции **Minerr** реализованы те же самые алгоритмы, что и в функции **Find**, иным является только условие завершения работы численного метода. Поэтому пользователь может тем же самым образом, с помощью контекстного меню, выбирать численный алгоритм приближенного решения для функции **Minerr**.

Пример использования функции **Minerr** показан в листинге 1.9. Как видно, достаточно заменить в вычислительном блоке имя функции на **Minerr**, чтобы вместо точного (с

точностью до **TOL**) получить приближенное решение уравнения, заданного после ключевого слова **Given**.

Листинг 1.9. Приближений решение уравнения, имеющего корень (x=0, y=0):

$$x := 1 \qquad y := 1$$

$$k := 10^6$$

Given

$$k \cdot x^2 + y^2 = 0$$

$$v := \text{Minerr}(x, y)$$

$$v = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Листинг 1.9 демонстрирует приближенное решение уравнения $kx^2 + y^2 := 0$, которое при любом значении коэффициента *k* имеет единственный точный корень *(x=0,y=0)*. Тем не менее, при попытке решить его функцией **Find** для больших k, порядка принятых в листинге, происходит генерация ошибки "No solution was found" (Решение не найдено). Это связано с иным поведением функции $f(x,y)=kx^2+y^2$ вблизи ее корня, по сравнению с функциями, приводимыми в качестве примеров выше. В отличие от них, $f(x,y)$ не пересекает плоскость $f(x,y)=0$, а лишь касается ее (рис. 1.7) в точке $(x=0,y=0)$. Поэтому и найти корень изложенными в предыдущем разделе градиентными методами сложнее, поскольку вблизи корня производные $f_{(x,y)}$ близки к нулю, и итерации могут уводить предполагаемое решение далеко от корня.

Ситуация еще более ухудшается, если наряду с корнем типа касания (см. рис. 8.7) имеются (возможно, весьма удаленные) корни типа пересечения. Тогда попытка решить уравнение или систему уравнений с помощью функции **Find** может приводить к нахождению корня второго типа, даже если начальное приближение было взято очень близко к первому. Поэтому если Вы предполагаете, что система уравнений имеет корень типа касания, намного предпочтительнее использовать функцию **Minerr**, тем более, что всегда есть возможность проверить правильность решения уравнений простой подстановкой в них полученного решения (см. листинг 1.6).

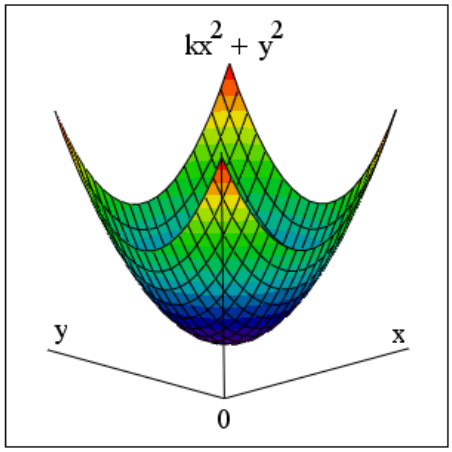

Рис. 1.7. График функции $kx^2 + y_2$

В листинге 1.9 мы рассмотрели пример нахождения существующего решения уравнения. Приведем в заключение пример нахождения функцией **Minerr** приближенного

решения несовместной системы уравнений и неравенств (листинг 1.10). Решение, выдаваемое функцией **Minerr**, минимизирует невязку данной системы.

Листинг 1.10. Приближенное решение несовместной системы уравнений и неравенств:

$x := 1 \qquad y := 1$

Given

$x^2 + y^2 = -1$

$x > 0.1$

$y \le -0.2$

$\text{Minerr}(x, y) = \begin{pmatrix} -0.042 \\ -0.085 \end{pmatrix}$

Как видно из листинга, в качестве результата выдаются значения переменных, наилучшим образом удовлетворяющие уравнению и неравенствам внутри вычислительного блока. Внимательный читатель может обнаружить, что решение, выдаваемое функцией **Minerr** в рассматриваемом примере, не является единственным, поскольку множество пар значений *(x,y)* в равной степени минимизирует невязку данной системы уравнений и неравенств. Поэтому для различных начальных значений будут получаться разные решения, подобно тому как разные решения выдаются функцией **Find** в случае бесконечного множества корней (см. обсуждение листинга 1.6)

Еще более опасен случай, когда имеются всего несколько локальных минимумов функции невязки. Тогда неудачно выбранное начальное приближение приведет к выдаче именно этого локального минимума, несмотря на то, что другой (глобальный) минимум невязки может удовлетворять системе гораздо лучше.