Решение уравнений и систем в Mathcad

В.Очков

Многие инженерные и научно-технические задачи сводятся к *решению уравнений* и их *систем* — алгебраических, степенных, тригонометрических, т. е. к поиску значений *неизвестных*, превращающих эти уравнения в тождества строго или приближенно. Успех в решении подобных задач зависит не только от мощности соответствующих инструментов, встроенных в Mathcad, но и от знания пользователем их особенностей, нюансов, сильных и слабых сторон.

> ***Примечание***
> Иногда в Mathcad-документах можно увидеть не готовую формулу, по которой рассчитывается корень системы, а сама система с операторами ее решения. Делается это намеренно для того, чтобы, высветить тот или иной баланс сил, энергий, масс и т. д., заложенный в расчет или этап расчета. Это очень полезно и уместно в расчетах образовательной направленности.

3.1. Встроенные решатели Mathcad

Инструменты решения аналитических уравнений и их систем в Mathcad собраны в трех "ящиках с инструментами" (toolbox) — см. рис. 3.1:

- встроенные функции категории **Решение уравнений** в диалоговом окне **Вставка функции**;
- команды символьных преобразований из меню **Символьные операции**, в частности, команда **Переменная | Решить**.
- операторы символьных преобразований, в частности, оператор `solve` (решить для переменной) из панели инструментов **Символьные**;

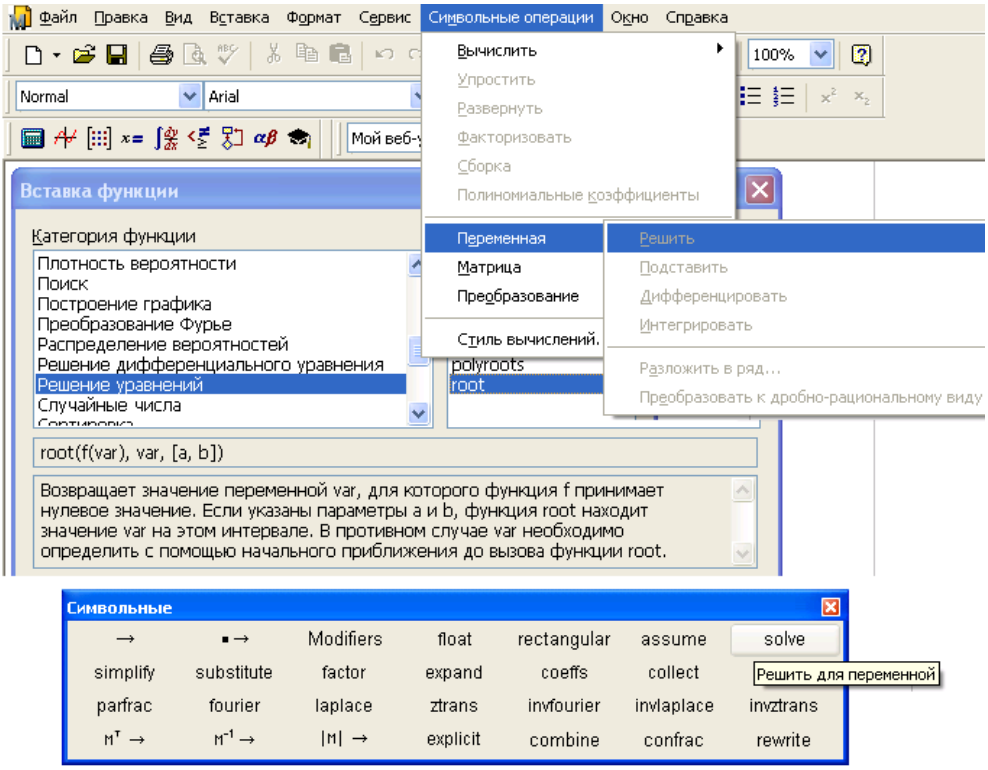


Рис. 3.1. Три инструмента решения уравнений и их систем в среде Mathcad

Как следует из названий, два последних инструмента — это символьная математика, математика компьютерных аналитических преобразований. В первом же пункте отмечены встроенные функции, и некоторые из них имеют двоякую сущность — могут возвращать численный или символьный ответ в зависимости от того, каким оператором вывода их "потревожили" — оператором → (символьный вывод) или оператором = (численный вывод).

3.2. Поиск нулей и функций и решение уравнений

Для решения одиночных уравнений предназначены встроенные функции `root` (корень) и `polyroots` (корни полинома). Если быть точным, то функции `root` и `polyroots`, возвращают не *корни* уравнений, а *нули* функций. В случае, когда функции `root` и `polyroots` необходимо применить к поиску корня *(корней)* уравнения, то его следует преобразовать в функцию: перенести одну из частей уравнения в другую, поменяв его знак, и работать с "ненулевой" частью как с функцией. Например, уравнение

$a \cdot x = b$

преобразуется в функцию

```
f(x):=a·x – b
```

Функция `root` первоначально имела только два аргумента: первый из них — это анализируемая функция (в полном ее написании `y(x)`, а не просто `y`), у которой ищется нуль, а второй отмечал аргумент (неизвестное), найденное значение которого делает функцию равной нулю. В двухаргументной функции `root` заложен *метод секущих*, требующий первого приближения. Вблизи которого определяется вторая опорная точка, равная значению первого приближения плюс значение встроенной (системной) переменной `TOL`, значение которой по умолчанию равно `0.0001`, умноженной на значение первого приближения. Через эти две точки проводится *секущая*, пересечение которой с осью *x* дает очередное (третье) приближение. Итерации к корню (к нулю функции) заканчиваются тогда, когда в очередном приближении значение функции будет отличаться от нуля менее чем на значение встроенной (системной) переменной `TOL`. Реализацию метода секущих средствами программирования Mathcad, а также пошаговое приближение к корню (нулю) средствами можно увидеть на сайте **http://twt.mpei.ac.ru/mas/worksheets/secant.mcd**. Фрагмент реализации этого метода представлен на рис. 3.2.

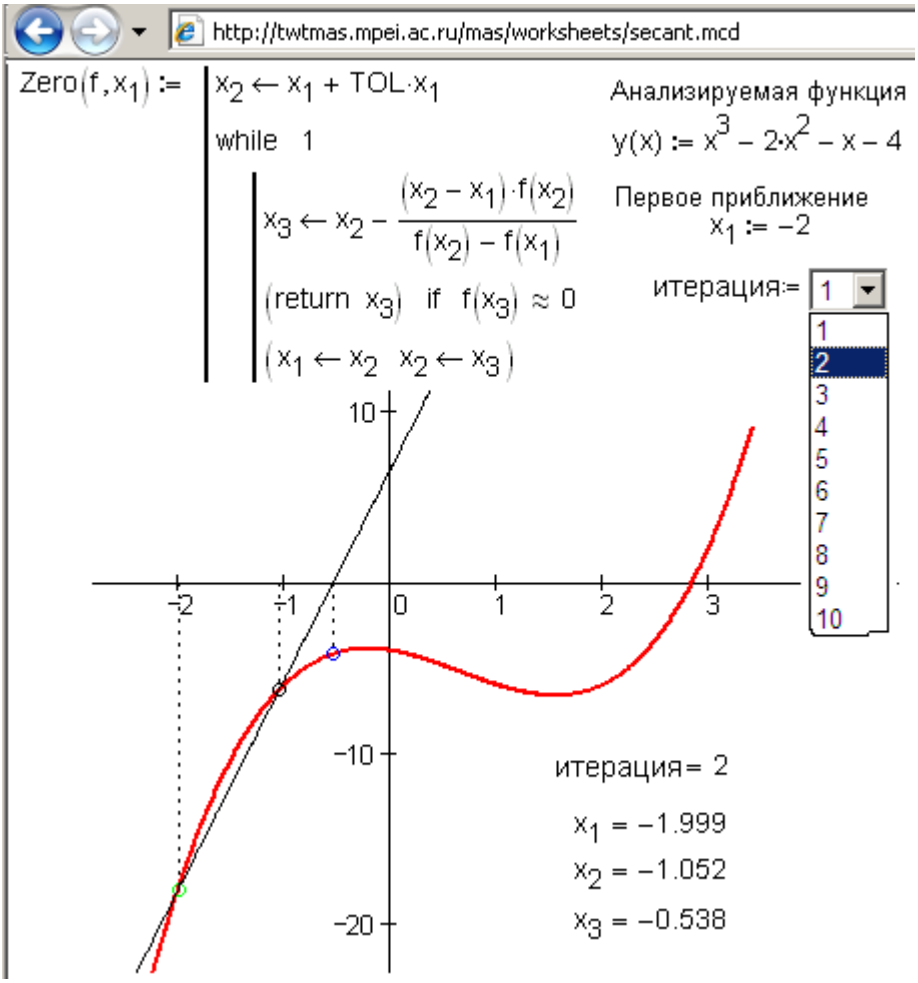


Рис. 3.2. Метод секущих

Посетитель указанного сайта указывает в списке номер итерации (1, 2, 3 и т.д.) и видит, как секущая приближается к нулю. Если затем нажимать кнопки обозревателя Интернета "вперед-назад" (см. левый верхний угол рис. 2.2), то можно ускорить перемещение секущей, вызывая картинки из буфера обозревателя интернета, и получить псевдоанимацию, о которой мы рассказывали в конце *главы 1*.

В Mathcad 13/14 появились встроенные средства отладки программ, которые, в частности, позволяют прослеживать работу встроенных инструментов поиска нуля функции. На рис. 3.3 показано, какие промежуточные значения просчитываются при поиске нуля пользовательской функции `y(x)` с помощью встроенной двухаргументной функции `root`.

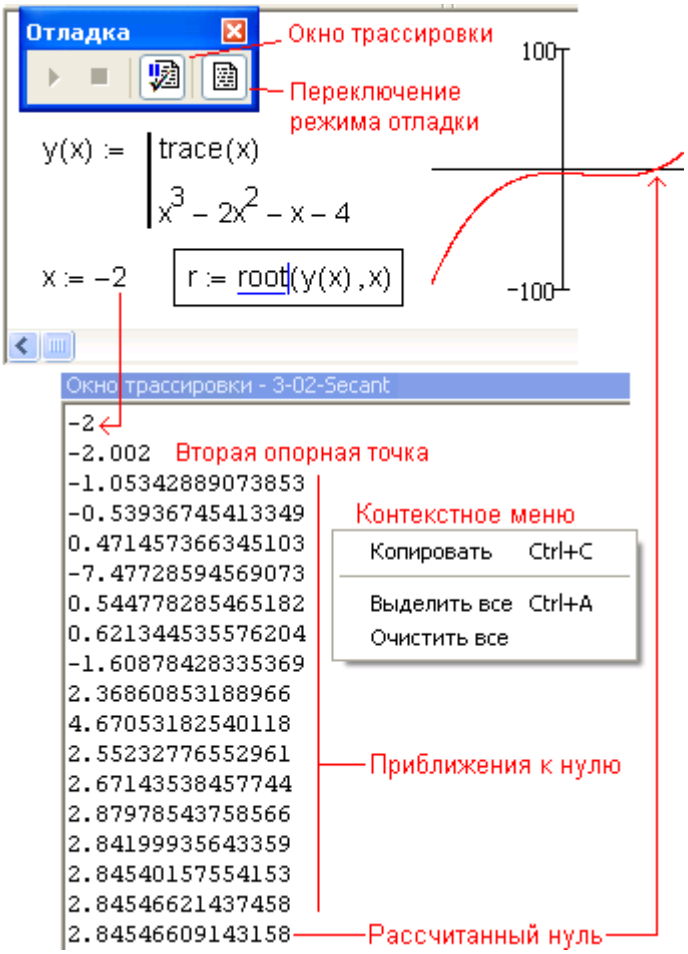


Рис. 3.3. Трассировка двухаргументной функции `root`

Чтобы проследить работу функции `root`, нужно анализируемой функции `y(x)` добавить встроенную функцию `trace(x)`, открыть окно трассировки через меню **Вид** и включить режим трассировки соответствующим переключателем окна **Отладка** (см. рис. 3.3). После этого нажатия клавиши <F9> будут приводить к тому, что в окне трассировки станут появляться числа промежуточных значений поиска нуля функции. Эту колонку чисел можно перенести в основное окно Mathcad и при необходимости использовать в расчетах.

На рис. 3.4 показано, как с помощью ползунка (см. рис. 1.21) на оси *x* отмечается точка первого приближения и какое значение нуля функции (переменная `zero`) возвращает при этом функция `root`.

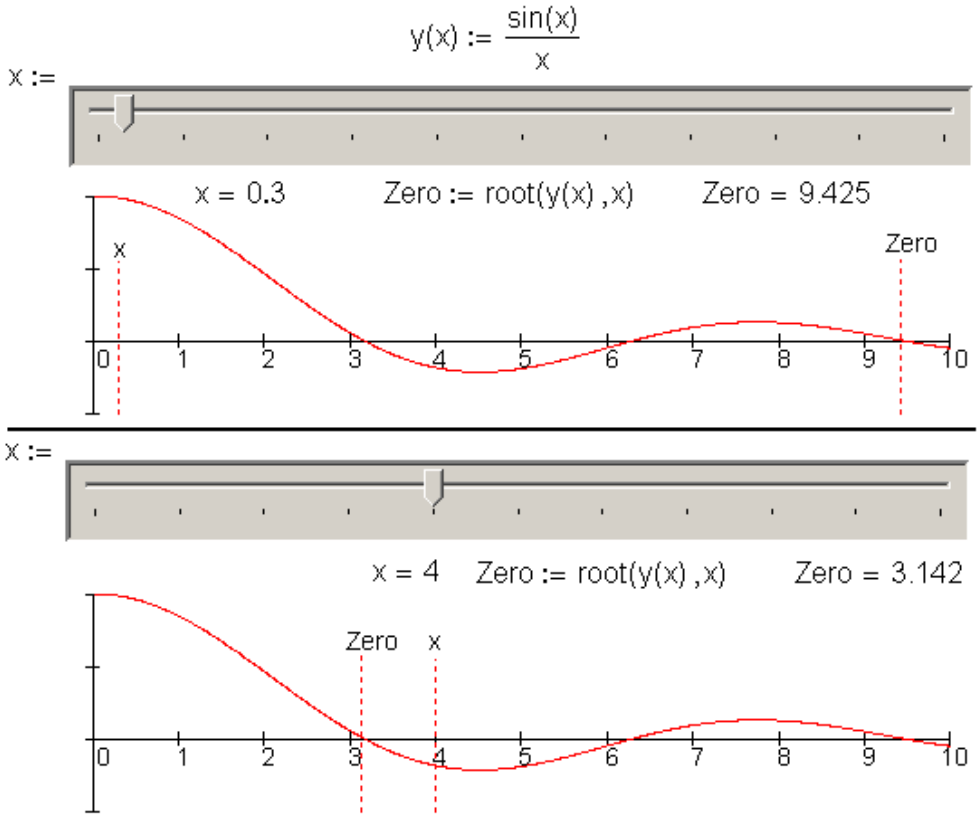


Рис. 3.4. Испытание двухаргументной функции `root`

Один из недостатков функции `root` в ее двухаргументной реализации заключается в том, что нет четкой привязки первого приближения с найденным нулем функции. Так из рис. 3.4 видно, что первое приближение `x=0.3` (верхняя часть рисунка) дает не ближайший к этой точке нуль (`3.142`), а третий нуль (`x=9.425`) — мы намеренно анализируем функцию `sin(x)/x`, у которой бесконечное число нулей. Эта особенность хорошо отображена на рис. 3.5, где показано, что место выбора первого приближения вблизи точки `x=0` очень сильно влияет на значение возвращаемого нуля (ось *y*).

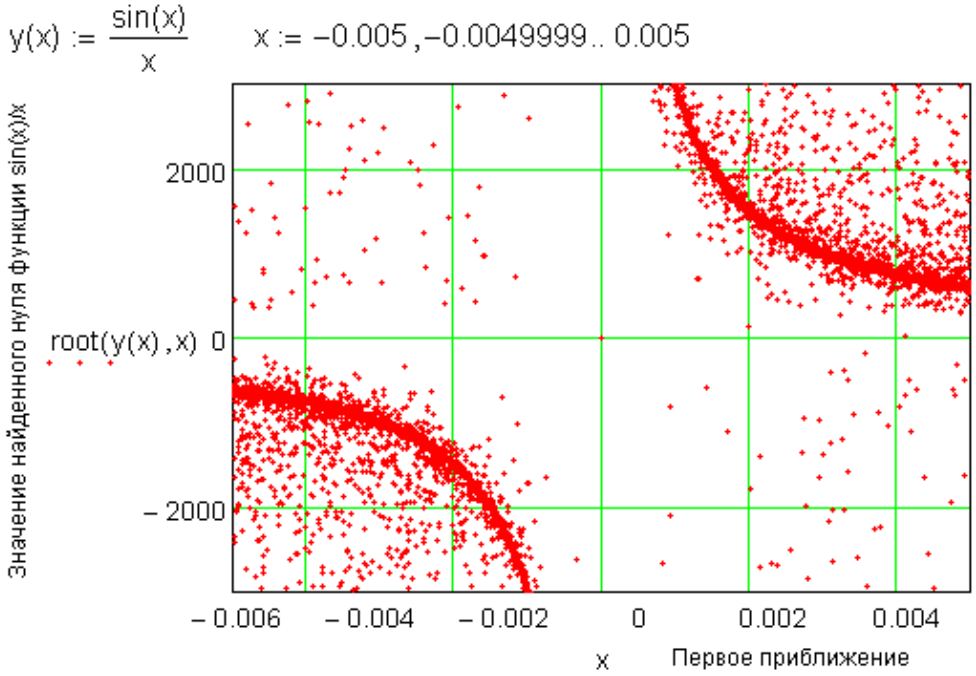


Рис. 3.5. Разброс нулей двухаргументной функции `root`

Поиск нуля функции, показанный на рис. 3.2, методом секущих назвать можно с некоторой натяжкой. Дело в том, что этот метод требует в качестве первого приближения не одно, а два значения аргумента. При одном значении аргумента (одно значение для первого приближения) у кривой, отображающей на графике анализируемую функцию, можно провести не секущую, а *касательную*. Метод касательных (Ньютона) поиска нуля функции показан на рис. 3.6. Его недостаток состоит в том, что для его реализации требуется знание производной анализируемой функции.

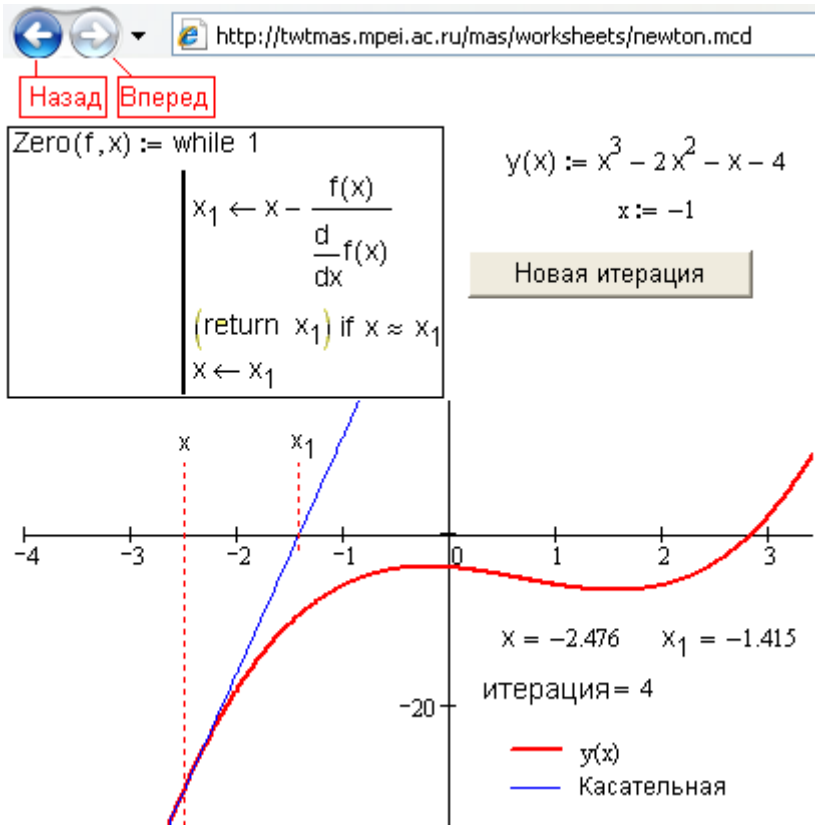


Рис. 3.6. Метод Ньютона

В программе, показанной на рис. 3.6, производная от анализируемой функции определяется численно, что само по себе считается не вполне корректной математической операцией. Во встроенной функции `root` и пользовательской `Zero`, показанной на рис. 3.2, заложен метод *секущих — Ньютона*: при первой итерации у кривой проводится (почти) касательная, затем приближения ведутся через секущие.

"Непредсказуемость" функции `root` в ее двухаргументном варианте (см. рис. 3.4 и 3.5), а также тот факт, что возвращаемый ею нуль зависит от значения переменной, находящейся вне списка аргументов функции `root` (*см. разд. 1.3.1* с комментариями на этот счет), заставила разработчиков Mathcad ввести в седьмую версию этого математического пакета *четырехаргументный* вариант функции `root`. Теперь ее синтаксис выглядит так:

```
root(f(var), var[, a, b])
```

Квадратные скобки в описании функции означают *необязательность* элементов, заключенных в эти скобки. Поменялась не только форма, но и содержание функции `root` — в ее четырехаргументный вариант заложен метод *половинного деления*. Его

суть хорошо иллюстрирует старая притча о том, как можно поймать льва в пустыне: "... для этого нужно оградить ее забором, перегородить этот загон пополам и посмотреть, в какой половине оказался лев. Далее половину со львом снова разделить пополам и так поступать до тех пор, пока лев не окажется в некоем ящике". Программу метода половинного деления при "поимке" нуля функции можно увидеть на рис. 3.7.

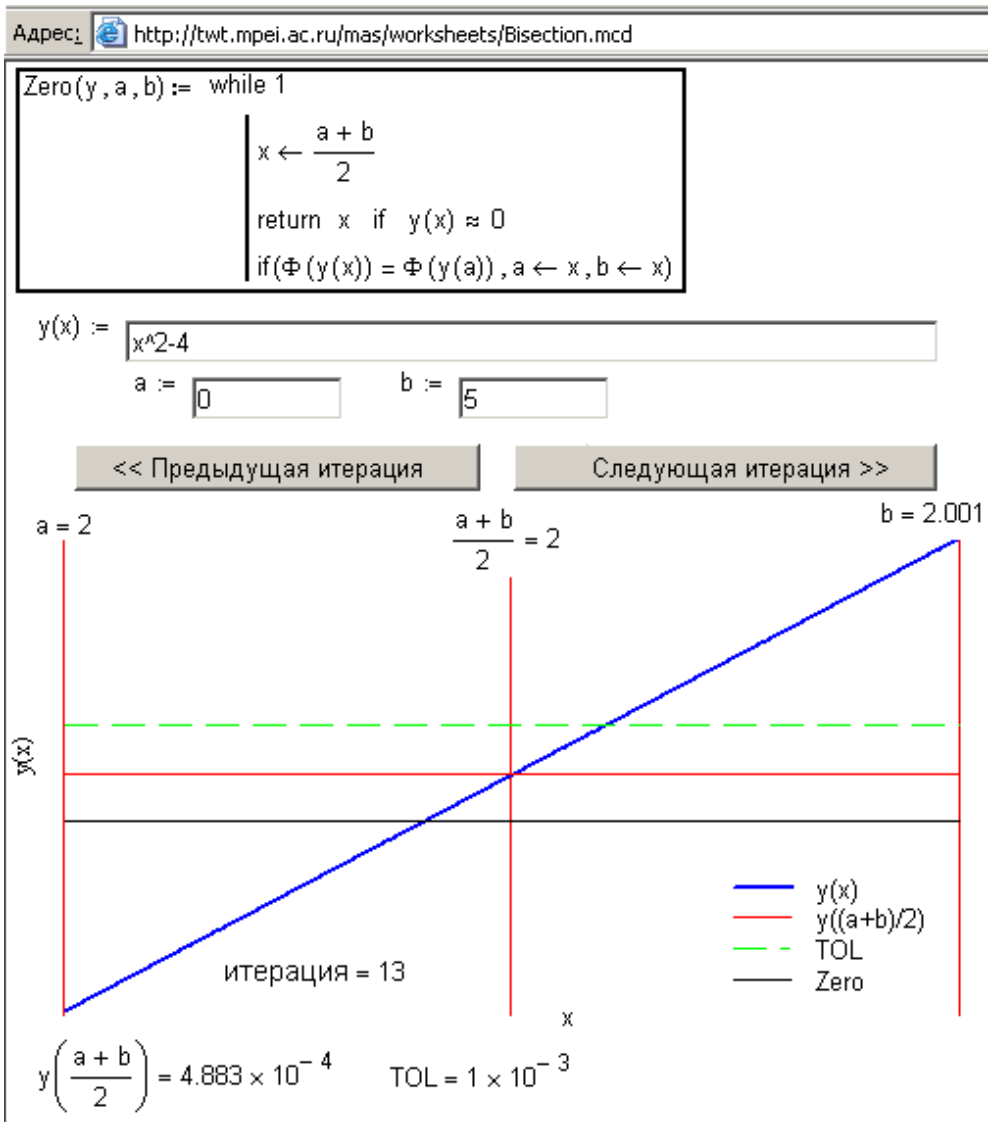


Рис. 3.7. Метод половинного деления

На рис. 3.7 зафиксирован момент 13-й итерации к нулю функции, когда значение `y((a+b)/2)` стало по модулю меньше значения `TOL` (решение задачи). На рис. 3.8 показана трассировка четырехаргументной функции `root`.

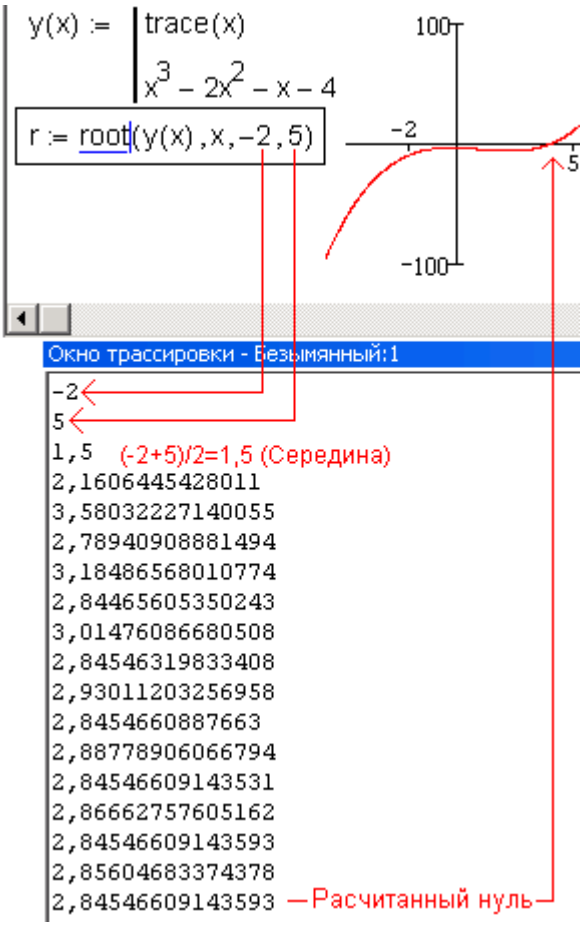


Рис. 3.8. Трассировка четырехаргументной функции `root`

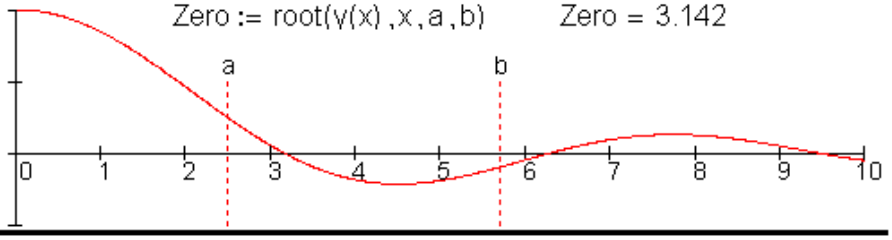
Функция `root`, реализуя метод половинного деления, уже не требует первого приближения, но предполагает задание интервала [`a`, `b`], где необходимо найти нуль функции. На рис. 3.9 показано, как с помощью уже двух слайдеров выставляются значения переменных `a` и `b`, и какой нуль из многих существующих возвращает при этом функция `root` с дополнительными двумя аргументами.

a :=

$$y(x) := \frac{\sin(x)}{x}$$

b :=

Zero := root(y(x) ,x ,a ,b) Zero = 3.142

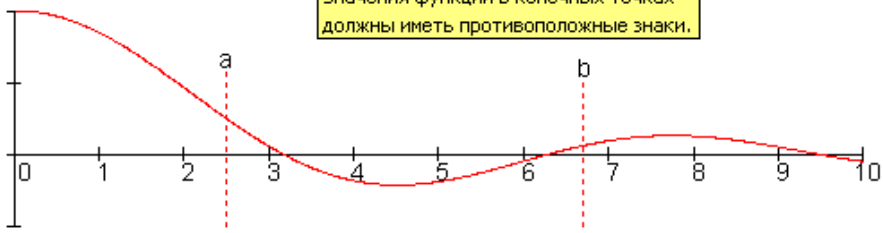


a :=

b :=

Zero := root(y(x) ,x ,a ,b) Zero = ∎

Значения функции в конечных точках должны иметь противоположные знаки.



a :=

b :=

Zero := root(y(x) ,x ,a ,b) Zero = 3.142

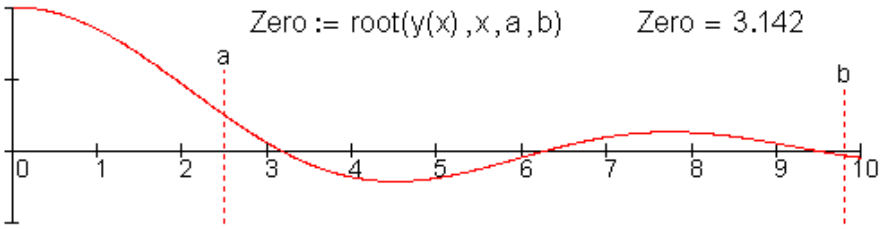


Рис. 3.9. Испытание четырехаргументной функции `root`

Метод половинного деления, заложенный в функцию `root`, требует, чтобы значения анализируемой функции в точках `a` и `b` имели противоположный знак. В этом случае, если анализируемая функция, конечно, непрерывна, на отрезке [`a`, `b`] будет находиться, как минимум, один нуль. В противном случае функция `root` будет возвращать сообщение об ошибке, призывающей пользователя изменить значения `a` и `b` (рис. 3.9, в центре). Если же на отрезке [`a`, `b`] окажется более одного корня (три (рис. 3.9, внизу), пять, семь и т. д. при, повторяю, непрерывности анализируемой функции), то функция `root` вернет один из них. Какой именно — изображено на рис. 3.10, где показан результат сканирования прямоугольной области [`a`, `b`] с помощью четырехаргументной функции `root`.

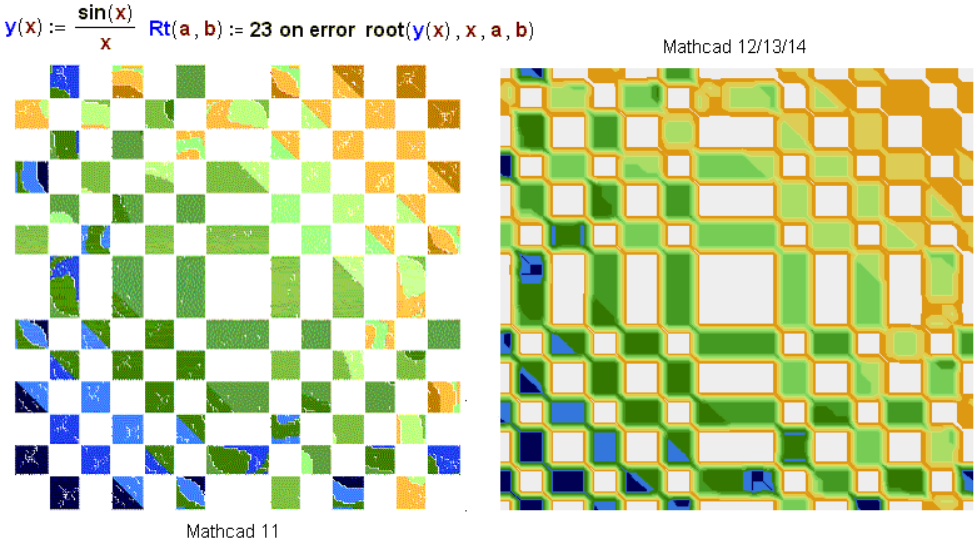


Рис. 3.10. "Отпечаток" нулей четырехаргументной функции `root`

Белые области на рис. 3.10 отмечают заданные значения переменных `a` и `b`, при которых нуль функции `sin(x)/x` не был найден. Цвет в закрашенных областях фиксирует тот или иной найденный нуль: холодные тона — отрицательные нули, а теплые — положительные, если взглянуть на соответствующий цветной рисунок на сайте книги.

Примечание

Если нуль функции не найден, срабатывает оператор Mathcad `on error`, возвращающий значение своего первого операнда (у нас оно равно 23, т. е. число, превышающее значение самого большого нуля в рассматриваемом диапазоне), если во втором операнде происходит сбой, зафиксировавший сообщение об ошибке (см. рис. 3.9, в центре).

Функцию `root` несложно приспособить и для решения *обратных задач*, когда необходимо найти значение аргумента функции по заданному значению самой функции (см. рис. 3.11, где ищется диаметр шара объемом 30 куб. см).

$$V_{шар}(d) := \frac{4}{3} \cdot \pi \cdot \left(\frac{d}{2}\right)^3 \qquad V := 30cm^3$$

$$d := root\left(V_{шар}(d) - V, d, 0mm, 100mm\right)$$

$$d = 38.537\,mm$$

$$V_{шар}(d) = 29.967\,cm^3$$

Рис. 3.11. Решение обратной задачи с использованием функции `root`

Если анализируемая функция представляет собой полином *n*-й степени, то для поиска его нулей можно (нужно) использовать встроенную функцию `polyroots`, пример работы которой показан на рис. 3.12.

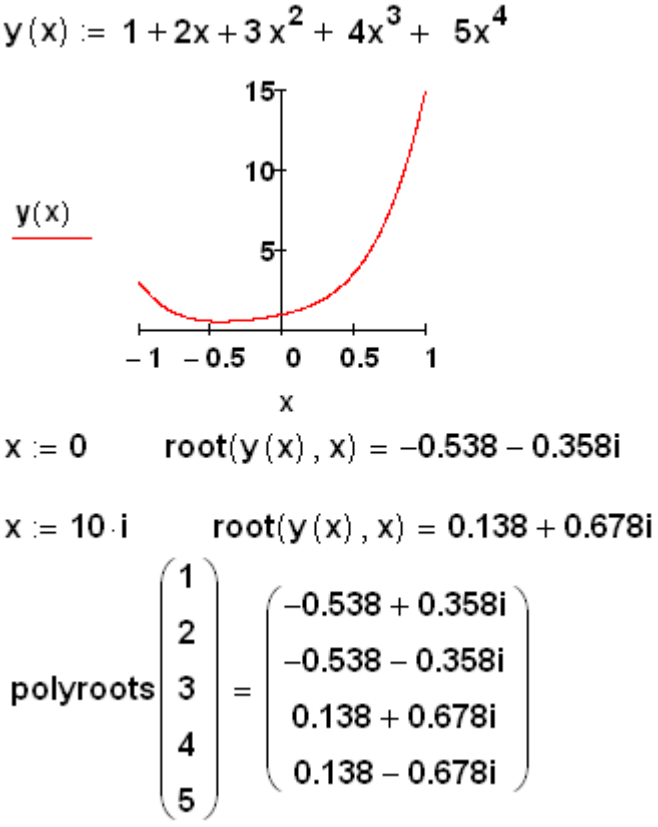


Рис. 3.12. Поиск нулей полинома

Аргументом функции `polyroots` является вектор коэффициентов полинома, а возвращает она вектор всех нулей полинома, как действительных, так и *комплексных*[1]. Кстати, о комплексных нулях. Функция `root`, о которой речь шла выше, в двухаргументном варианте (см. рис. 3.4) выгодно отличается от своего четырехаргументного аналога (рис. 3.9) тем, что она может возвращать и комплексные корни, что зафиксировано на рис. 3.13.

[1] Я специально выбрал для рис. 3.12 полином, у которого все нули комплексные.

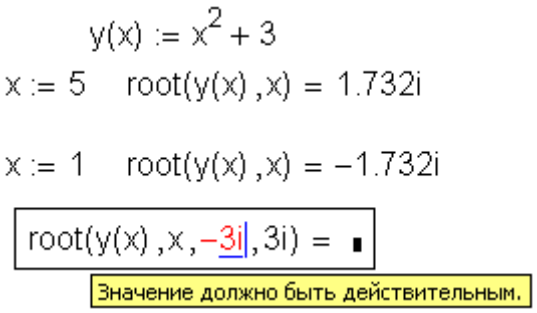


Рис. 3.13. Поиск комплексных нулей

Нередко в составленном выражении пользователь не видит полинома, хотя само выражение таковым является. На рис. 3.14 показано, как оператор символьных преобразований `coeffs` (полиноминальные коэффициенты) из панели инструментов **Символьные** "вытаскивают" коэффициенты полинома, которые затем можно использовать в функции `polyroots` для поиска нулей. Примерно такую же работу делает и оператор и `expand` (развернуть выражение).

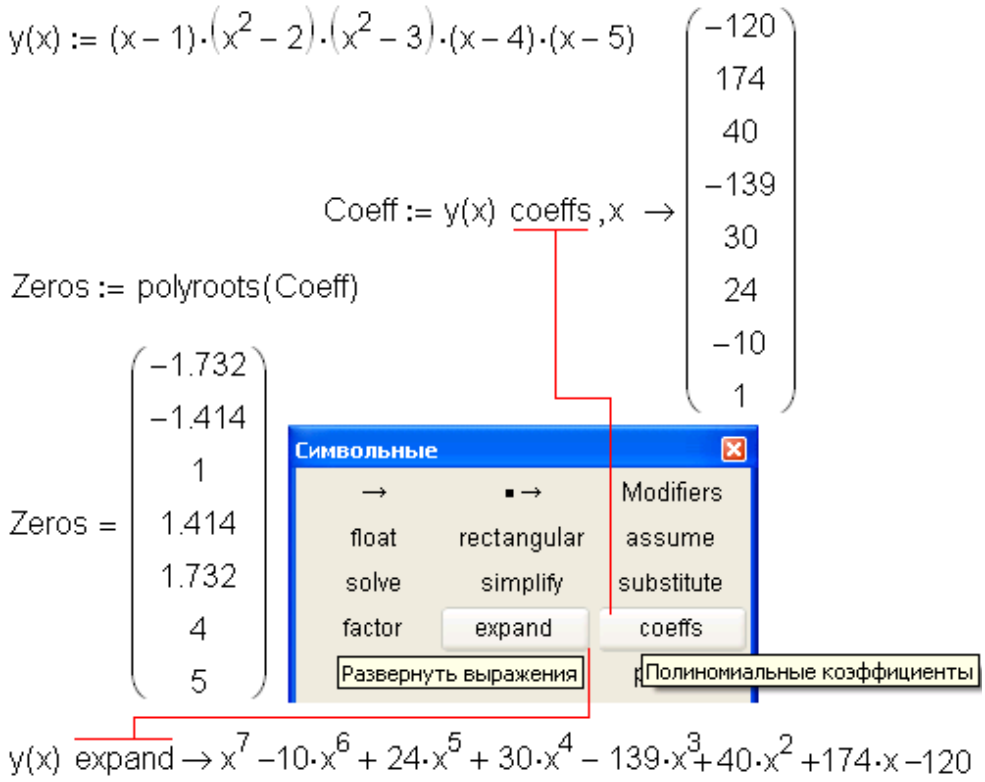


Рис. 3.14. Извлечение коэффициентов полинома

С другой стороны, при решении инженерных задач пользователю не нужны все нули функции, а необходим только определенный — ненулевой и/или некомплексный (действительный), например. В этом случае лучше вызывать не функцию `polyroots`, а функцию `root` в ее двух- или четырехаргументных вариантах, намечая требуемый нуль первым приближением (см. рис. 3.4) или диапазоном (см. рис. 3.9).

Находить нули функции помогает также и *графика* Mathcad, о которой мы рассказали в *разд. 1.5*. На декартовом графике *лупой* или *трассировкой* можно уточнить начальное приближение к нулю и/или сделать проверку правильности решения (рис. 3.15).

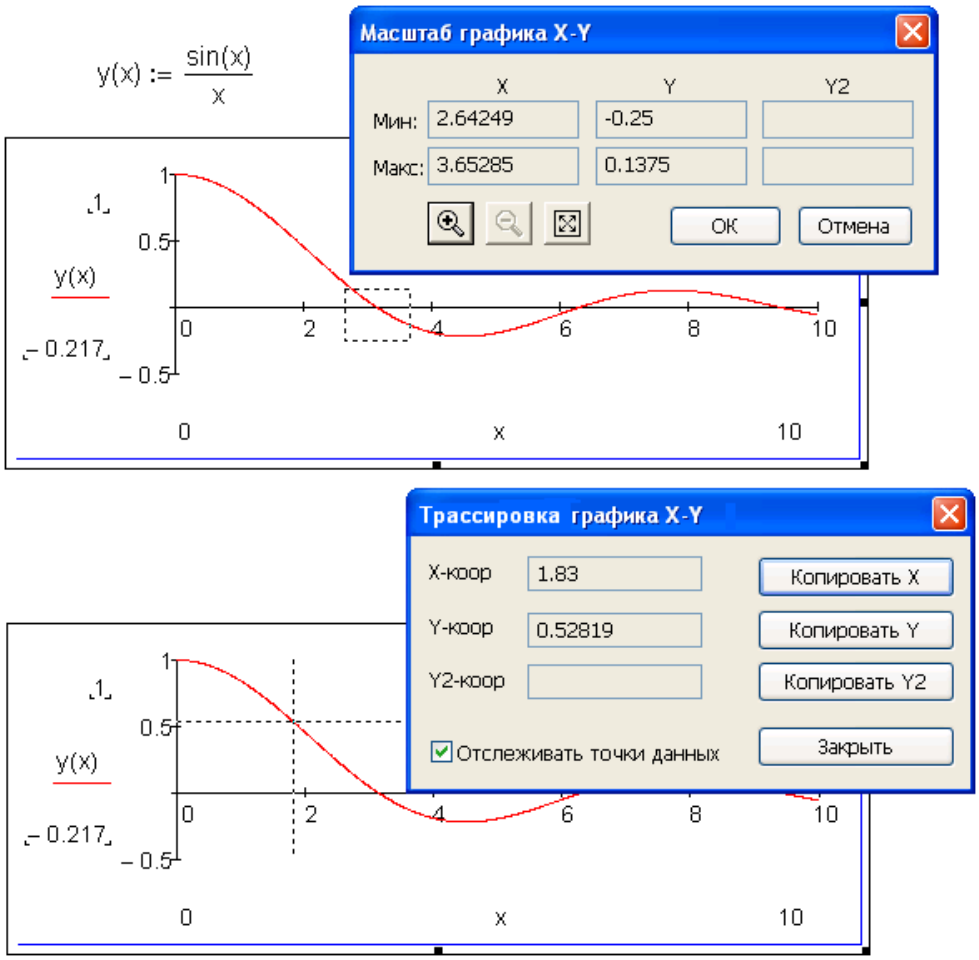


Рис. 3.15. Уточнение нулей с помощью графики

При трассировке графика выбор первого приближение (значение `1.83`, если иметь в виду случай, зафиксированный на рис. 3.15) автоматизирован через кнопку **Копировать X**. При этом значение `1.83` будет сохранено в буфере обмена и его можно будет вставить в сам Mathcad-документ — `x: = 1.83`.

В среде Mathcad 14 новая символьная математика позволяет находить все нули периодических функций, где `z` — это множество целых чисел.

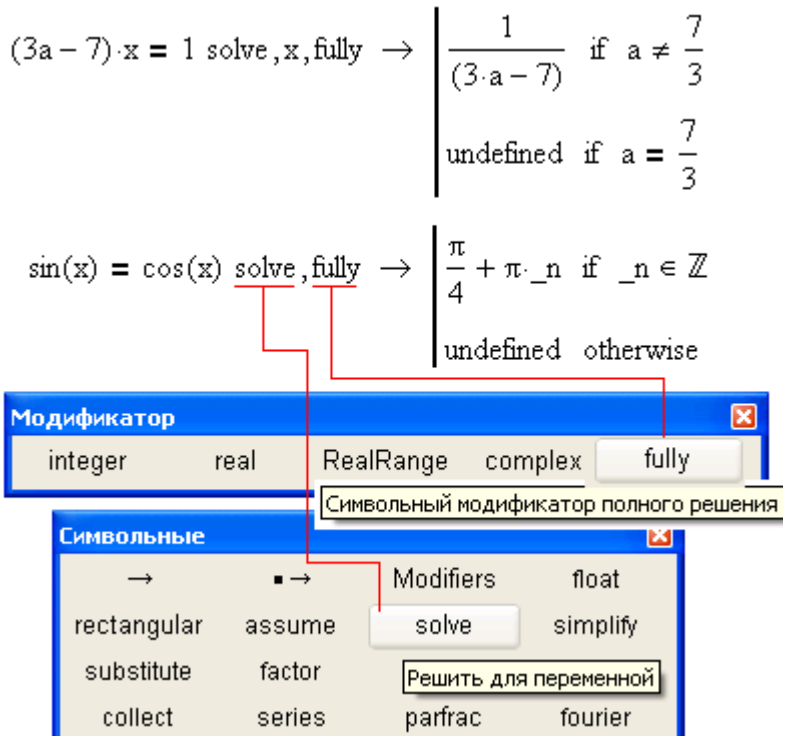


Рис. 3.16. Поиск всех нулей функции средствами символьной математики

3.3. Решение систем алгебраических уравнений

Теперь перейдем к разбору методов решения в среде Mathcad *систем* уравнений, а не одиночных уравнений и функций, отметив при этом, что некоторые системы можно подстановкой свести к одиночным уравнениям и использовать функцию `root`[2].

3.3.1. Решение систем линейных алгебраических уравнений

Сначала поговорим о решении систем *линейных* алгебраических уравнений, которые в матричной форме записываются так: `A·X=B`, где `A` — квадратная матрица коэффициентов при неизвестных (вектор `X`), а `B` — вектор свободных членов. Первоначально для решения такой системы в среде Mathcad был предназначен оператор $A^{-1}\cdot B$ — произведение инвертированной (обратной) матрицы `A` на вектор свободных членов `B`. Затем (в версии 7 и выше) в Mathcad была введена функция `lsolve`.

[2] Знаменитый принцип математики, когда новую задачу можно свести к старой, уже разобранной.

Примечание

В среде Mathcad есть негласное правило отмечать строчными (маленькими) буквами скалярные величины, а прописными (большими) — массивы (векторы и матрицы). Если, конечно, здесь нет других соображений.

Решение системы линейных алгебраических уравнений через произведение инвертированной матрицы коэффициентов при неизвестных на вектор свободных членов ($A^{-1} \cdot B$) и через функцию `lsolve` могут отличаться в том случае, если матрица `A` *вырожденная* (сингулярная), т. е. ее определитель равен нулю. В первом случае ($A^{-1} \cdot B$) решение сразу прерывается соответствующим сообщением об ошибке "singular matrix", а во втором — продолжается, если решение не просто есть, а их... бесчисленное множество. Такая вычислительная ситуация зафиксирована на рис. 3.17, где показано решение системы уравнений с так называемой "телефонной" матрицей `A`, повторяющей кнопки на телефоне.

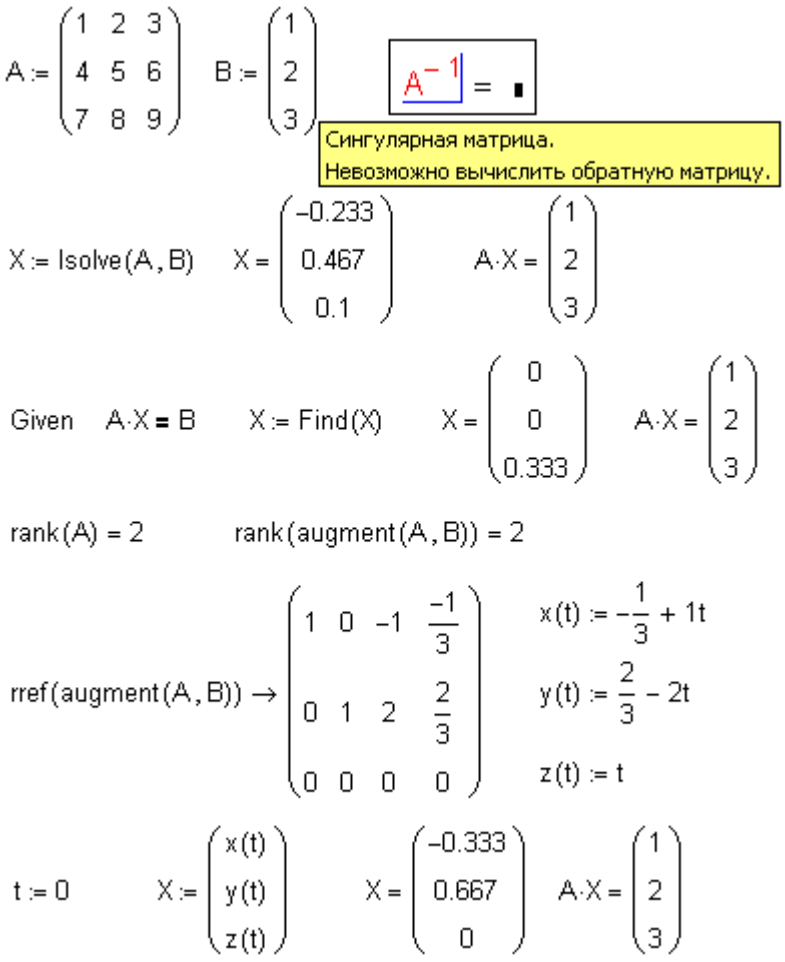


Рис. 3.17. Решение системы линейных алгебраических уравнений с помощью функций `lsolve`, `Find` и `rref`

Система уравнений, коэффициенты при неизвестных и свободные члены которой вводятся в расчет первыми двумя операторами, показанными на рис. 3.17, не может быть решена через произведение A^{-1} на `B`, т. к. матрица `A` сингулярная (вырожденная) — см. третий оператор на рис. 3.17. На рис. 3.17 обратная матрица от матрицы `A` определялась с помощью численной, а не символьной математики Mathcad (оператор =, а не оператор →). Численную же математику по другому называют *математикой приближенных вычислений*, имея в виду, что она, как правило, дает не точные, а приближенные ответы. Приближенный ответ возможен и при инвертировании вырожденной матрицы, что может привести к неверному ответу при решении соответствующей системы линейных алгебраических уравнений (рис. 3.18).

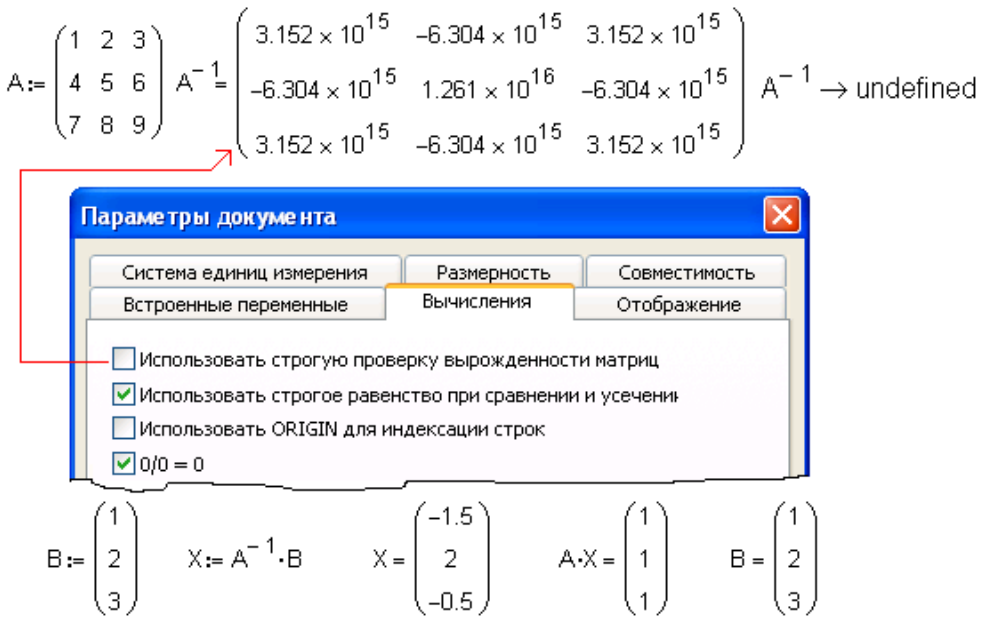


Рис. 3.18. Одна из ошибок при решении системы линейных алгебраических уравнений — неправильное вычисление обратной матрицы

В среде Mathcad только в 2000-й версии стало возможным избежать грубой ошибки, показанной на рис. 3.18. В Mathcad был введен контроль за вырожденностью матрицы при работе и с численной математикой (сброшенный флажок **Использовать строгую проверку вырожденности матрицы**).

Линейная алгебра в этом случае гласит, что у системы с вырожденной матрицей может либо не быть ни одного решения, либо существовать бесконечное множество решений ("впадание в крайности"), два из которых и найдены на рис. 3.17 (вторая и третья строки операторов, заканчивающихся проверкой решения `A·X=`*`исходный вектор свободных членов системы`*. На рис. 3.19 показано, как средствами символьной математики можно найти это множество решений через уравнение линии.

$$z1(x,y) := 1x + 2y + 3z = 1 \text{ solve}, z \rightarrow \frac{-1}{3}\cdot x - \frac{2}{3}\cdot y + \frac{1}{3}$$

$$z2(x,y) := 4x + 5y + 6z = 2 \text{ solve}, z \rightarrow \frac{-2}{3}\cdot x - \frac{5}{6}\cdot y + \frac{1}{3}$$

$$z3(x,y) := 7x + 8y + 9z = 3 \text{ solve}, z \rightarrow \frac{-7}{9}\cdot x - \frac{8}{9}\cdot y + \frac{1}{3}$$

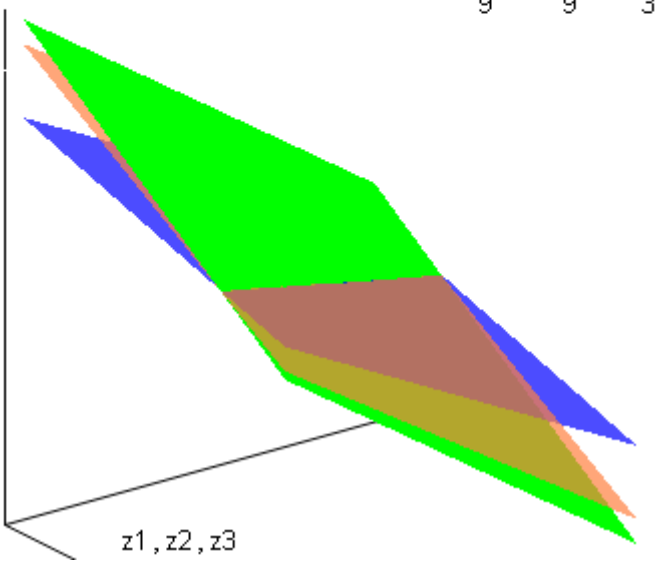


Рис. 3.19. Графическая иллюстрация множества корней системы трех линейных алгебраических уравнений

Эту ситуацию в линейной алгебре описывает теорема Кронекера — Капелли (см., например, ее изложение на сайте **http://www.exponenta.ru/educat/class/courses/la/theme5/theory.asp**), гласящая, что система линейных уравнений с нулевым детерминантом (вырожденность матрицы `A`) может иметь решение, если ранги самой матрицы `A` и расширенной матрицы `augment(A, B)` равны и меньше порядка анализируемой системы (см. четвертую строку операторов на рис. 3.17). Расширенную матрицу с помощью встроенной функции `rret` можно свести к ступенчатой гауссовой, из которой легко "выуживается" решение нашей системы в виде независимой переменной `z` (система, кстати, может иметь и комплексные решения) и двух функций `y(z)` и `x(z)` — см. рис. 3.17, внизу.

Примечание

Встроенная Mathcad-функция `augment` сливает воедино "по горизонтали" две и более матрицы (вектора, скаляра). Аналогичная "вертикальная" функция носит имя `stack`.

Решение нашей системы с вырожденной матрицей коэффициентов при неизвестных (т. е. аналитическое (параметрическое) описание прямой линии, образованной пересечением трех плоскостей) можно получить проще, чем показано на рис. 3.17. Для этого нужно отказаться от инструментов, применяемых к линейным системам (функции `lsolve` и `rref`, см. рис. 3.17), а прибегнуть к универсальному оператору `solve` (см. панель **Символьные** на рис. 3.1), предназначенному для аналитического решения уравнений общего вида (рис. 3.20).

Графическая интерпретация задачи сводится к тому, что три плоскости могут пересекаться либо в одной точке, либо в линии ("книга" с тремя листами: наш случай — см. рис. 3.19), либо вовсе не иметь общих точек пересечения (параллельность двух или трех плоскостей).

$$\begin{pmatrix} 1x+2y+3z=1 \\ 4x+5y+6z=2 \\ 7x+8y+9z=3 \end{pmatrix} \text{solve}, \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} z-\frac{1}{3} & -2\cdot z+\frac{2}{3} & z \end{pmatrix}$$

Одно из уравнений лишнее

$$\begin{pmatrix} 1x+2y+3z=1 \\ 4x+5y+6z=2 \end{pmatrix} \text{solve}, \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} z-\frac{1}{3} & -2\cdot z+\frac{2}{3} & z \end{pmatrix}$$

Рис. 3.20. Решение системы линейных алгебраических уравнений с помощью оператора `solve`

Функции `lsolve` и `Find`[3], как видно из рис. 3.17, дали два разных решения, лежащих в области `z`, `y(z)` и `x(z)`. Второе отличие в работе этих функций заключается в том, что функция `lsolve` не требует первого приближения и всегда выдает один "только ей ведомый" ответ. Функция же `Find` требует первого приближения и дает ответ в зависимости от его значения. В задаче на рис 3.17 первое приближение для функции `Find` — это ответ, данный функцией `lsolve`. Несмотря на это (функцию `Find` "ткнули носом" в ответ, вернее, в один из ответов) функция `Find` дала новый результат. Ответ функции `Find` зависит не только от первого приближения и от прочих установок (от значения системных переменных `TOL` и `CTOL` — об этом будет сказано чуть ниже), но и от способа решения задачи, на которой она настроена. Список эти способов (алгоритмов) появляется на экране дисплея при нажатии правой кнопки мыши на слове `Find`. Что при этом происходит, показано на рис. 3.21.

[3] О ней мы поговорим ниже.

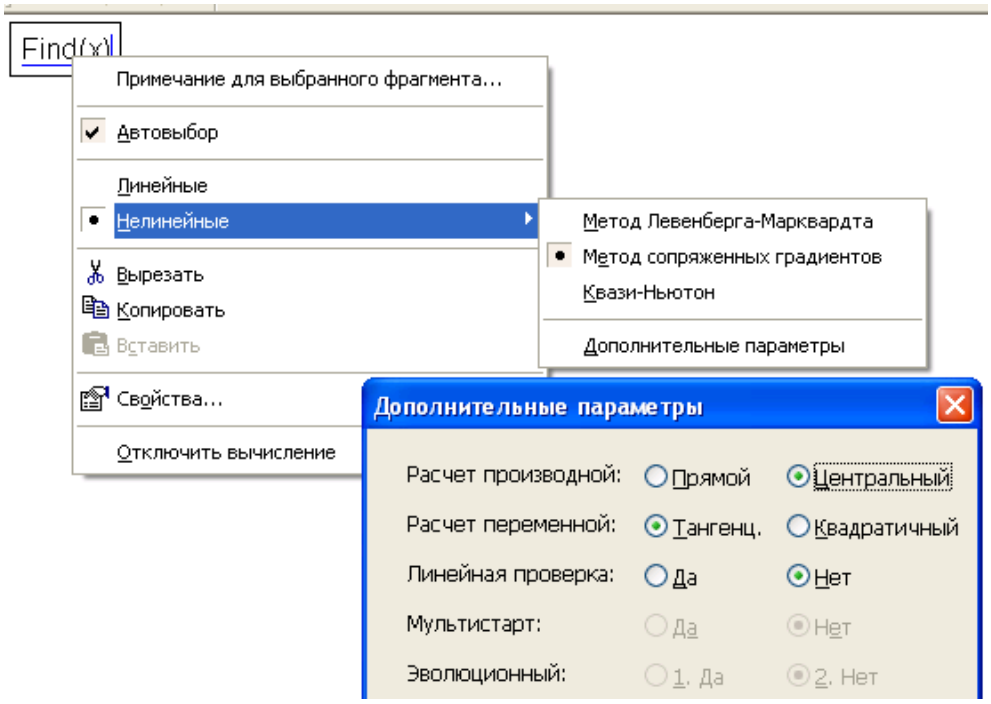


Рис. 3.21. Инструменты настройки Mathcad-функции `Find`

В среде Mathcad 13/14 с помощью функции `lsolve` стало возможным решать так называемые недоопределенные и переопределенные линейные алгебраические уравнения (рис. 3.22).

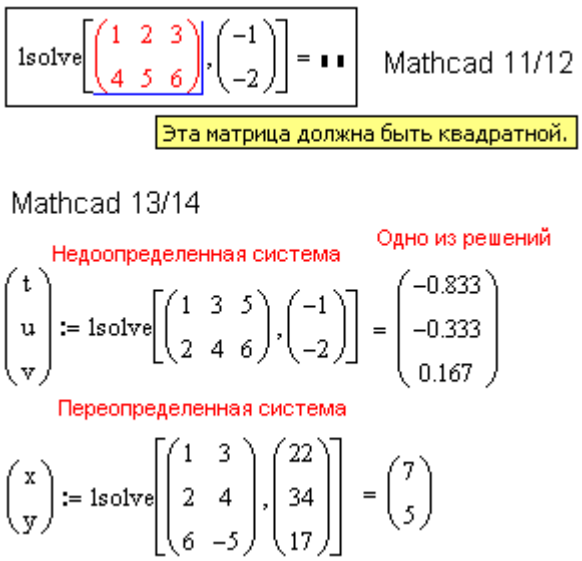


Рис. 3.22. Решение недоопределенных и переопределенных уравнений

3.3.2. Решение систем нелинейных алгебраических уравнений

А сейчас "повозимся" с системой уже *нелинейных* уравнений.

На рис. 3.23 и 3.24 показано решение системы двух нелинейных алгебраических уравнений средствами символьной и численной математики Mathcad соответственно. При решении системы инструментами символьной математики в переменную `All_Sol` (от англ. *all solution* — все решения) заносится матрица, строки которой — это корни системы. Их 24, что подсчитывается функцией `rows`. Из этих корней только четыре действительные. Это можно подсчитать, рассмотрев матрицу `All_Sol` (рис. 3.23) или построив график системы (рис. 3.24). Как был построен этот график, рассказано на сайте книги — главное же, что на нем видны эти четыре точки пересечения двух кривых, отображающих два уравнения нашей системы.

Given

$$x^2 \cdot y^3 - 7 \cdot x^3 \cdot y^3 = 7$$

$$(x^2 + y^2)^2 = 7\,(x^2 - y^2)$$

Поиск корней системы

$$All\_Sol := Find(x, y)^T\ float, 5 \rightarrow$$

1.2014	−.86820
−.82517 − .63490·i	−.84785 − .37507·i
−.82517 + .63490·i	−.84785 + .37507·i
.38908 + .86932·i	−.66745 − .83448·i
.38908 − .86932·i	−.66745 + .83448·i
2.5504	−.39970
$-2.6876 + 6.1184 \cdot 10^{-2} \cdot i$	−.17568 − .32065·i
$-2.6876 - 6.1184 \cdot 10^{-2} \cdot i$	−.17568 + .32065·i
−.28491 + .17760·i	$-5.5662 \cdot 10^{-2} - 2.6751 \cdot i$
−.28491 − .17760·i	$-5.5662 \cdot 10^{-2} + 2.6751 \cdot i$
$5.0922 \cdot 10^{-2} - .38482 \cdot i$	$-2.4418 \cdot 10^{-2} - 2.5597 \cdot i$
$5.0922 \cdot 10^{-2} + .38482 \cdot i$	$-2.4418 \cdot 10^{-2} + 2.5597 \cdot i$
$1.5092 \cdot 10^{-2} - .85613 \cdot i$	$4.3778 \cdot 10^{-2} - 1.1628 \cdot i$
$1.5092 \cdot 10^{-2} + .85613 \cdot i$	$4.3778 \cdot 10^{-2} + 1.1628 \cdot i$
.37739 + .17285·i	$6.9290 \cdot 10^{-2} - 2.7099 \cdot i$
.37739 − .17285·i	$6.9290 \cdot 10^{-2} + 2.7099 \cdot i$
$2.6909 + 6.5104 \cdot 10^{-2} \cdot i$	.18103 − .33220·i
$2.6909 - 6.5104 \cdot 10^{-2} \cdot i$	.18103 + .33220·i
−2.5582	.38389
−.34601 + .85818·i	.61366 − .86123·i
−.34601 − .85818·i	.61366 + .86123·i
−1.1363	.84602
.87730 − .64607·i	.88230 − .35602·i
.87730 + .64607·i	.88230 + .35602·i

Подсчет числа корней

$$rows(All\_Sol) = 24$$

Рис. 3.23. Полное решение системы двух нелинейных уравнений с помощью функций `Find` при ее аналитическом вызове (Mathcad 11/12/13)

Сама система на рис. 3.24 имеет неизвестные не в записи x и y, а в записи x_0 и x_1, т. е. в виде двух компонентов вектора x. К векторной записи неизвестных (x_0, x_1, x_2, ..., а не x, y, z, ...) приходится прибегать по ряду причин. Одна из них в том, что, если верить документации Mathcad, блоком Given/Find можно решить систему с числом неизвестных до 256, а количество аргументов функции Find, как и любой другой встроенной функции с переменным числом аргументов, не может превышать 50. Убрать эту нестыковку и поможет векторная запись неизвестных системы уравнений. Но главное преимущество вектора неизвестных по отношению к неизвестным скалярам состоит в том, что такой подход позволяет делать запись решения более компактной за счет, в том числе, использования операторов суммы и произведения (см., например, рис. 4.9).

Второе отличие решения, показанного на рис. 3.24, от решения, продемонстрированного на рис. 3.23, состоит в том, что функция Find не возвращает найденные значения неизвестных, превращающих уравнения в тождества, а формирует пользовательскую функцию с именем Sol, через вектор-аргумент которой в решение вводятся нужные первые приближения. Функция Sol в Mathcad-документе, показанном на рис. 3.24, вызывается пять раз при разных значениях этого первого приближения и возвращает четыре корня нашей системы и... один сбой, происшедший, когда функция Find не смогла решить задачу и вернула встроенное сообщение об ошибке, призывающее пользователя изменить первое приближение и/или значения встроенных (системных) переменных TOL (от англ. *tolerance* — погрешность) и/или CTOL (от англ. *constrains tolerance* — погрешность ограничений).

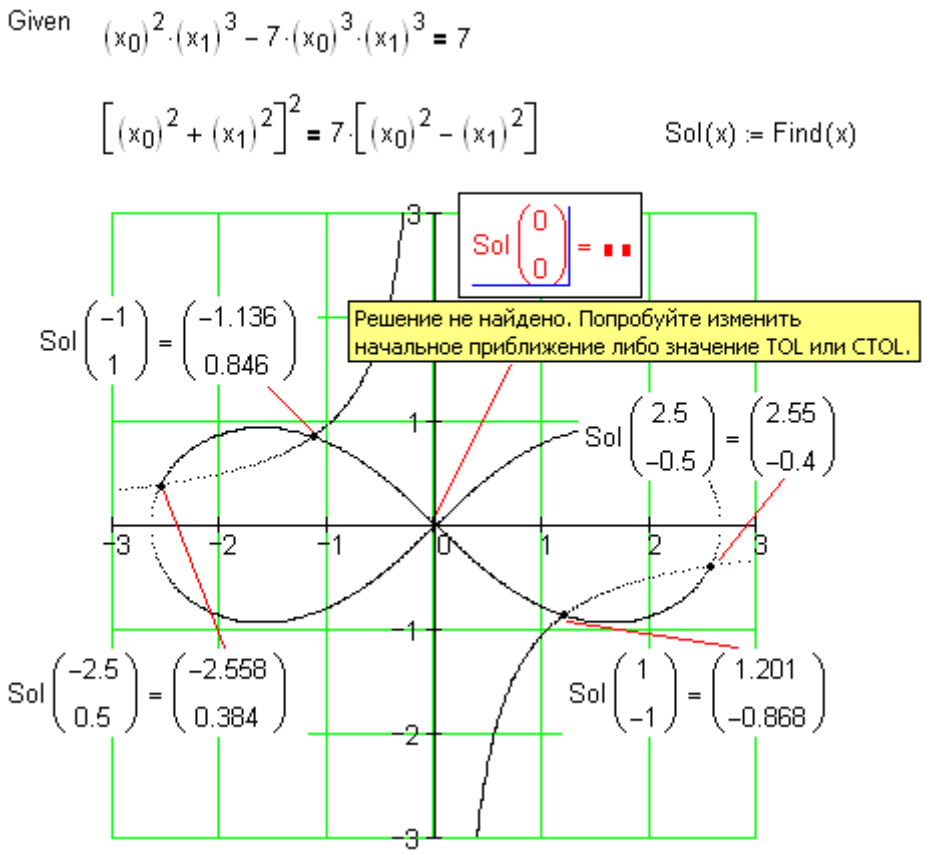


Рис. 3.24. Решение системы двух нелинейных уравнений с помощью функций `Find` при ее численном вызове

В среде Mathcad 14 ее новая символьная математика более строго подходит к полному решению системы нелинейных алгебраических уравнений и дает ответ с альтернативой, малая часть которого показана на рис. 3.25.

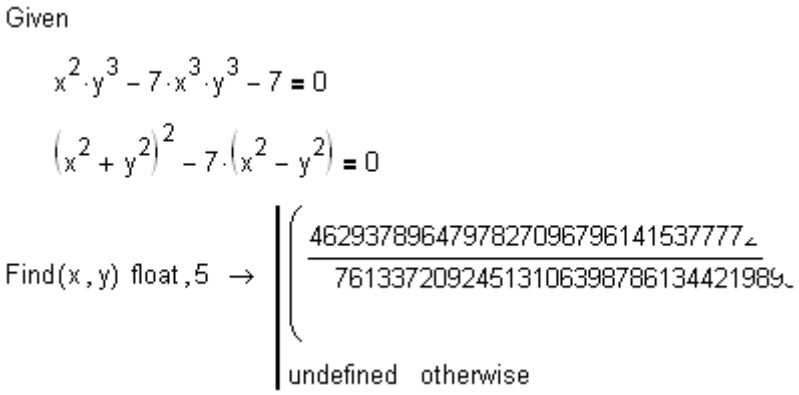


Рис. 3.25. Решение системы двух нелинейных уравнений с помощью функций `Find` при ее аналитическом вызове (Mathcad 14/15)

Можно *просканировать* поле первых приближений и узнать, из какой точки корень из четырех возможных возвращает функция `Find`, а где она дает сбой. Результаты такого сканирования представлены на рис. 3.26 и 3.27.

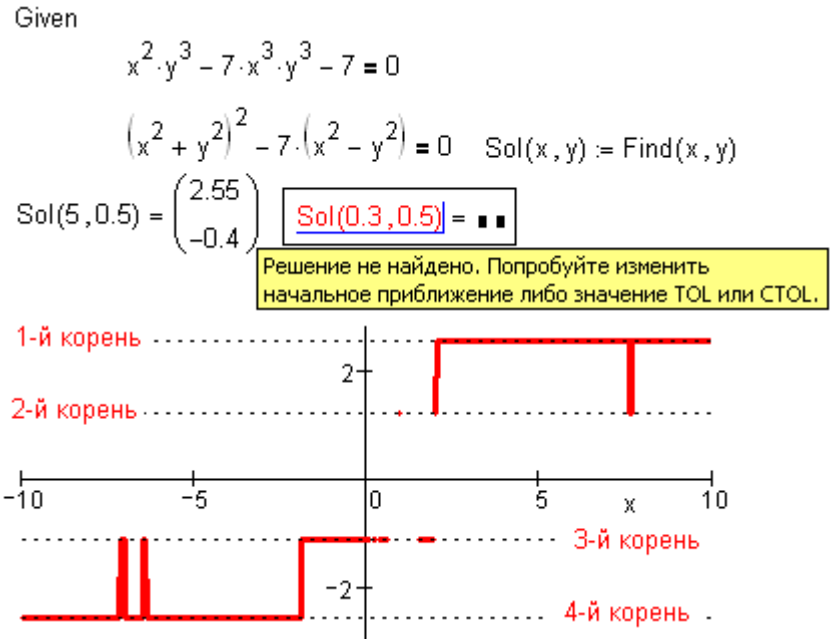


Рис. 3.26. Анализ стабильности решения системы двух нелинейных уравнений при фиксированном значении приближения по одной неизвестной

На рис. 3.26 строится декартов график корней, которые возвращает функция `Find` при изменении значения `x` от –10 до 10 и при фиксированном значении `y=0.5`. Из

графика видно, что в районе `x=0.2÷0.7` функция `Find` дает сбои — возвращает не один корень из четырех возможных, а сообщение об ошибке, которое на графике отображается разрывом. Более интересная и более информативная картинка получается не на линии (рис. 3.27), а на плоскости при сканировании решения уже не по одному, а по двум аргументам (неизвестным).

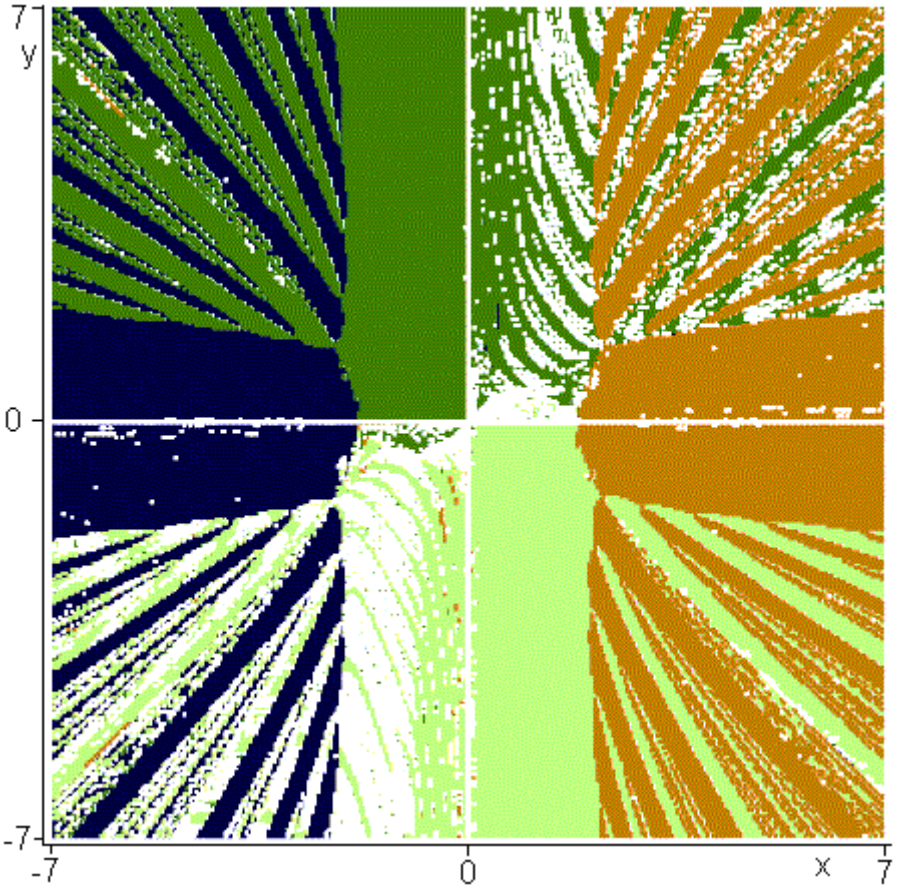


Рис. 3.27. "Картинка" численного поиска корней системы двух нелинейных алгебраических уравнений по алгоритму Левенберга — Маркворта

На рис. 3.27 цветом (оттенками серого, но на сайте книги можно найти и цветную картинку) выделены области на плоскости *xy*, принадлежащие одному из четырех корней нашей системы: точки, выбор которых в качестве первых приближений приводит к возврату функцией `Find` соответствующего, одного из четырех корней. Незакрашенные области ("белые пятна" на карте) — это точки, старт от которых приводит к сбою при поиске корня, т. е. к возврату не пары чисел, а сообщения об ошибке. При построении картинки, показанной на рис. 3.27, также как и в случае, отображен-

ном на рис. 3.10, был использован оператор `on error` (обработчик ошибок), позволивший гладко просканировать область *xy*.

Когда автор показал цветной вариант рис. 3.27 человеку, разбирающемуся в изобразительном искусстве, то он сказал, что это... "Пейзаж в Овере после дождя" Винсента Ван Гога. Правда, этот "искусствовед" не успел надеть свои очки, а рис. 3.27 был показан ему издали. Но при более детальном сравнении была отмечена схожесть цветовой гаммы и "фактуры" (полоски грядок) рисунков. Так ли это — пусть читатель решит сам, сравнив рис. 3.27 и рис. 3.28, а, еще лучше, не рисунки книги, а их цветные аналоги на сайте, открыв их на экране дисплея в двух окнах одновременно[4].

Рис. 3.28. Винсент Ван Гог "Пейзаж в Овере после дождя"

Но оставим в стороне "эстетическую сторону" рис. 3.27 и поговорим о практических делах.

Привязка цвета на графике к найденному корню более четко видна на рис. 3.29, где, во-первых, сужен диапазон сканирования на плоскости *xy* (не 7—7, а 3—3), а во-вторых, на "пейзаж" наложены графики наших анализируемых уравнений. Читатель

[4] Рис. 3.28 хорошо будет "висеть" в комнате любителя Mathcad, а рис. 3.27 хорошо "ляжет" на линолеум или кафель.

может взять другие два уравнения и, используя данную или иную цветовую гамму корней и иные настройки их поиска *(см. далее)*, попытаться создать новые "произведения изобразительного искусства", самостоятельные или перекликающиеся с известными.

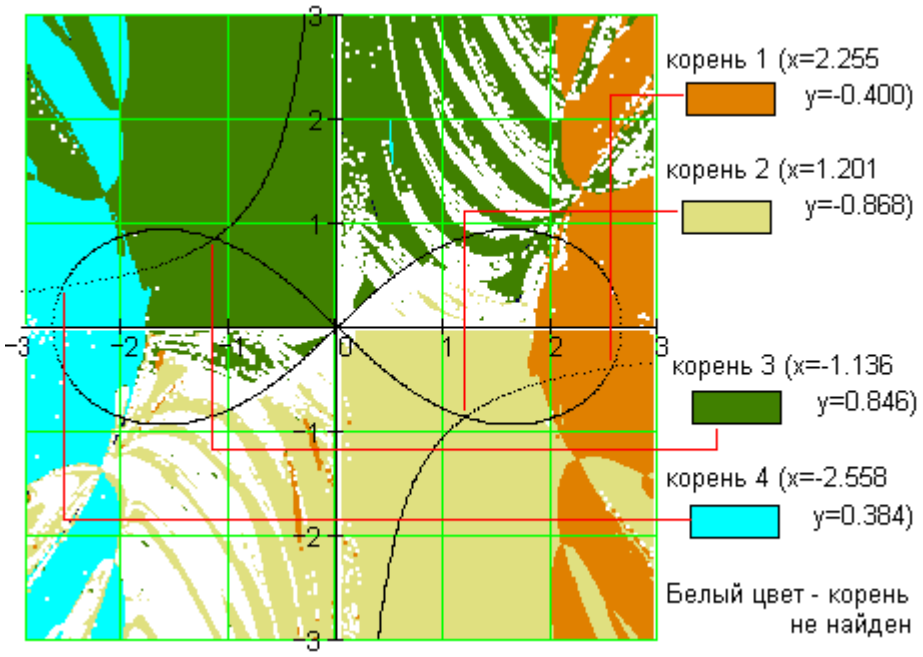


Рис. 3.29. "Пейзаж" численного поиска корней системы двух нелинейных алгебраических уравнений с наложенным графиком уравнений

Напомним, что в среде Mathcad реализованы *три метода* решения систем аналитических уравнений и систем: метод Левенберга — Марквардта, метод сопряженных градиентов и квазиметод Ньютона — см. рис. 3.21, где показано локальное меню, появляющееся после нажатия правой кнопки мыши на слове `Find` и позволяющее выбирать нужный метод решения. Замена метода Левенберга — Марквардта на метод сопряженных градиентов или квазиметод Ньютона резко меняет картину поиска корней нашей системы (рис. 3.30 и 3.31) и может служить неким графическим критерием сходимости того или иного алгоритма решения данной вычислительной задачи. В частности процент "белых пятен", который не сложно подсчитать, может давать уже не качественную, а количественную оценку сходимости конкретного метода решения на конкретной задаче.

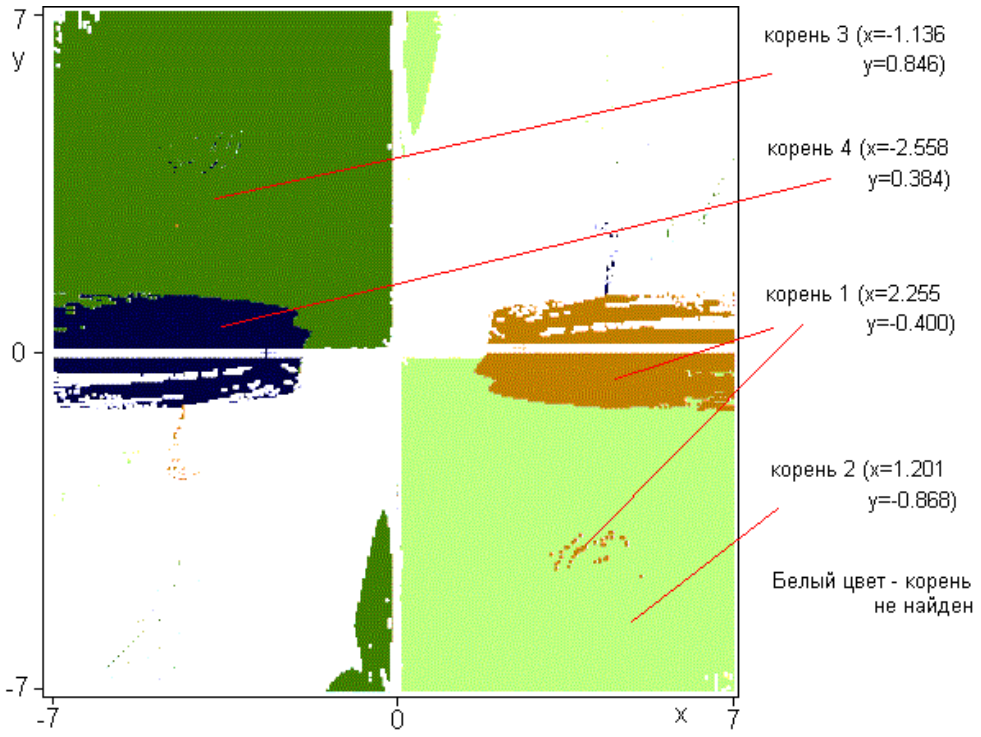


Рис. 3.30. "Пейзаж" численного поиска корней системы двух нелинейных алгебраических уравнений по алгоритму сопряженных градиентов

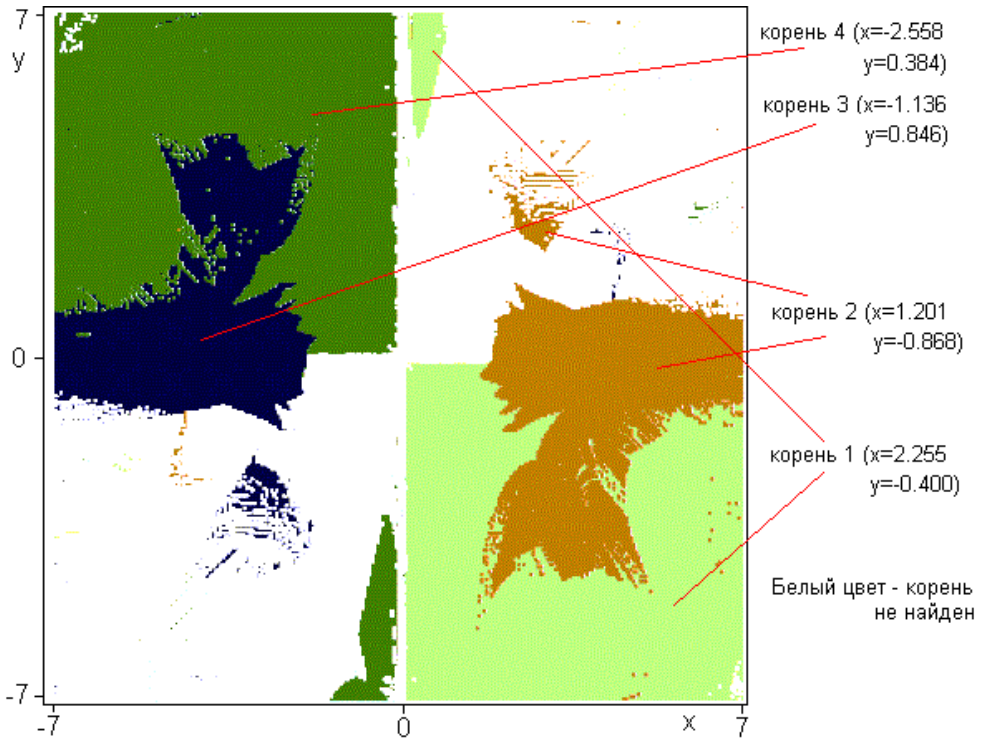


Рис. 3.31. "Пейзаж" численного поиска корней системы двух нелинейных алгебраических уравнений по квазиметоду Ньютона

Резко меняет картину поиска корней системы и изменение значения встроенной в Mathcad переменной `CTOL`, отвечающей за точность расчета. Функция `Find` возвращает значения своих аргументов, при которых невязка системы (разница между значениями левых и правых частей уравнений) не превышает по модулю значения переменной `CTOL`, которая при "создании пейзажей", показанных на рис. 3.27, 3.29—3.31, была равна `0.001` (значение по умолчанию). На рис. 3.32 "пейзаж" нашей системы двух уравнений "прорисован" для $\texttt{CTOL}=10^{-7}$.

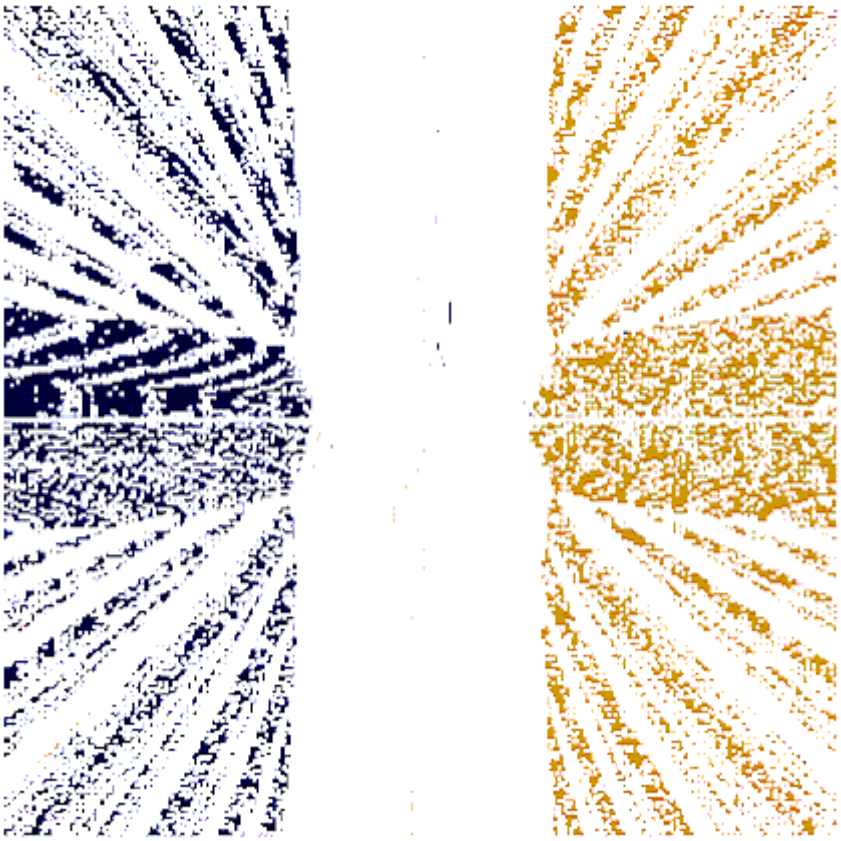


Рис. 3.32. "Пейзаж" численного поиска корней системы двух нелинейных алгебраических уравнений по алгоритму Левенберга — Марквардта с повышенной точностью

Из рис. 3.32 видно, что повышение точности поиска корней системы уравнений приводит к тому, что только два корня из четырех можно найти. Кроме того, сильно расширяется область "белых пятен" — точек, откуда корни вовсе не рассчитываются.

3.4. Помощь пользователя при решении уравнений и систем

Вышеописанные встроенные средства решения аналитических уравнений и их систем и, в частности, блок `Given`/`Find`, иногда дают сбой, и это вызвано, главным образом, их универсальностью. Решение широкого круга задач ведется в их общей постановке без учета специфики некоторых частных случаев. А из них — из частных случаев, как правило, и состоит реальная расчетная практика. Сказанное можно повторить и по-другому, более категорично. Встроенные средства Mathcad да и других

математических программ[5] годятся для решения несложных задач вспомогательного плана, возникающих по ходу решения основной. На решении серьезных задач практического плана эти средства иногда дают сбой или, что намного хуже, неправильный ответ. Сказанное проиллюстрируем реальным примером из химии. На рис. 3.33 показана попытка решения с помощью блока `Given/Find` задачи по расчету значения pH (водородный показатель, минус логарифм от концентрации ионов водорода) в буферном водном растворе[6].

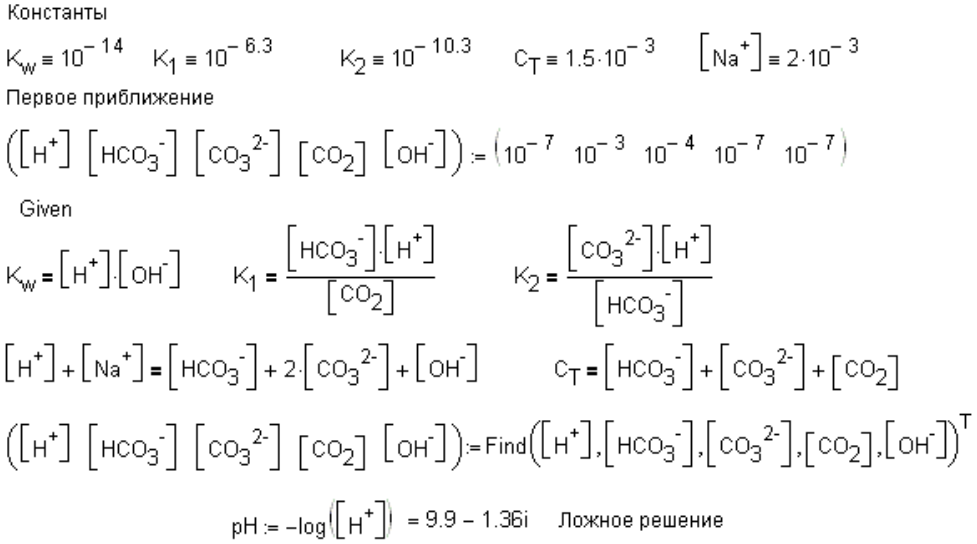


Рис. 3.33. Попытка решения системы уравнений через блок `Given-Find`

Решение задачи сводится к поиску корня системы пяти нелинейных уравнений: сумма концентраций катионов равна сумме концентраций анионов (уравнение электронейтральности раствора) и т. д.[7] Пять уравнений остаются при предельном упрощении задачи. Если же в задачу не вводить упрощения и допущения, то она может вылиться в несколько десятков уравнений и неравенств (ограничений). Так вот блок

---

[5] Этот тезис относится не только к встроенным средствам решения аналитических уравнений и их систем, но и к средствам решения других задач — оптимизация *(см. главу 4)*, решение дифференциальных уравнений *(см. главу 6)* и прочее.

[6] Буферность природных водоемов проявляется в частности в том, что сброс в них кислых стоков не приводит к резкому снижению значения pH. Буферностью, увы, не обладают чистые капли дождя, что приводит к резкому понижению pH в них при сорбции загрязнений атмосферы (проблема кислотных дождей). Буферности почти нет в чистейшей воде Байкала, что определяет ранимость этого уникального озера при сбросе в него стоков. С буферностью сталкиваешься когда, например, набираешь текст или макетируешь книгу на компьютере. В абзац вставляются дополнительные слова, а его длина не меняется. И только когда "буферность" абзаца исчерпана, его конец резко перескакивает на новое место.

[7] Мы не будем вдаваться в "химизм" задачи — главное тут математика — "математизм".

Given/Find не справился с задачей о значении pH буферного раствора даже в ее простейшем варианте: Выдается ответ в виде комплексного числа. Вернее, блок Given/Find с задачей справился, но выдал не тот корень системы, который мы ожидали[8]. Не помогали ни смена значений первых приближений, ни системных переменных TOL и CTOL, ни других установок, показанных на рис. 3.21. Решение этой проблемы (ахиллесова пята всех программ) в отказе от универсального встроенного решателя и переход к созданию индивидуальной расчетной методики, предназначенной для решения конкретной задачи. Так на рис. 3.34 показано, как задача о значении pH буферного раствора успешно решена за счет логарифмирования левых и правых частей уравнений системы, а на рис. 3.35 — за счет привлечения средств символьной математики Mathcad, выдавших все решения, из которых пользователю остается только выбрать нужное.

[8] В принципе, было бы хорошо, если бы в среде Mathcad можно было отключать работу с комплексными числами — отсутствие ответа все-таки лучше чем "глупый" ответ. Да и счет в этом режиме велся бы быстрее.

$$K_w \equiv 10^{-14} \quad K_1 \equiv 10^{-6.3} \quad K_2 \equiv 10^{-10.3} \quad C_T \equiv 1.5 \cdot 10^{-3} \quad Na \equiv 2 \cdot 10^{-3}$$

$$\left(\left[H^+\right] \left[HCO_3^-\right] \left[CO_3^{2-}\right] \left[CO_2\right] \left[OH^-\right]\right) := \left(10^{-7} \quad 10^{-3} \quad 10^{-4} \quad 10^{-7} \quad 10^{-7}\right)$$

Given

$$\log\left(K_w\right) = \log\left(\left[H^+\right]\right) + \log\left(\left[OH^-\right]\right)$$

$$\log\left(K_1\right) = \log\left(\left[H^+\right]\right) + \log\left(\left[HCO_3^-\right]\right) - \log\left(\left[CO_2\right]\right)$$

$$\log\left(K_2\right) = \log\left(\left[CO_3^{2-}\right]\right) + \log\left(\left[H^+\right]\right) - \log\left(\left[HCO_3^-\right]\right)$$

$$\log\left(\left[H^+\right] + Na\right) = \log\left(\left[HCO_3^-\right] + 2 \cdot \left[CO_3^{2-}\right] + \left[OH^-\right]\right)$$

$$\log\left(C_T\right) = \log\left(\left[HCO_3^-\right] + \left[CO_3^{2-}\right] + \left[CO_2\right]\right)$$

$$\begin{pmatrix} \left[H^+\right] \\ \left[HCO_3^-\right] \\ \left[CO_3^{2-}\right] \\ \left[CO_2\right] \\ \left[OH^-\right] \end{pmatrix} := \mathrm{Find}\left(\left[H^+\right], \left[HCO_3^-\right], \left[CO_3^{2-}\right], \left[CO_2\right], \left[OH^-\right]\right)$$

$$\begin{pmatrix} \left[H^+\right] \\ \left[HCO_3^-\right] \\ \left[CO_3^{2-}\right] \\ \left[CO_2\right] \\ \left[OH^-\right] \end{pmatrix} = \begin{pmatrix} 1.279 \times 10^{-10} \\ 1.078 \times 10^{-3} \\ 4.221 \times 10^{-4} \\ 2.751 \times 10^{-7} \\ 7.816 \times 10^{-5} \end{pmatrix} \qquad pH := -\log\left(\left[H^+\right]\right) = 9.89$$

Рис. 3.34. Решения системы уравнений через блок `Given/Find` и логарифмизацию уравнений

$K_w \equiv 10^{-14}$ $K_1 \equiv 10^{-6.3}$ $K_2 \equiv 10^{-10.3}$ $C_T \equiv 1.5 \cdot 10^{-3}$ $[Na^+] \equiv 2 \cdot 10^{-3}$

Given

$$K_w = [H^+] \cdot [OH^-] \qquad K_1 = \frac{[HCO_3^-] \cdot [H^+]}{[CO_2]} \qquad K_2 = \frac{[CO_3^{2-}] \cdot [H^+]}{[HCO_3^-]}$$

$$[H^+] + [Na^+] = [HCO_3^-] + 2 \cdot [CO_3^{2-}] + [OH^-] \qquad C_T = [HCO_3^-] + [CO_3^{2-}] + [CO_2]$$

$$[H^+] := \left(\text{Find}\left([H^+], [HCO_3^-], [CO_3^{2-}], [CO_2], [OH^-]\right)^T \right)^{\langle 0 \rangle} \rightarrow \begin{pmatrix} -2.000375918950490516 8 \cdot 10^{-3} \\ -1.253884021860433543 9 \cdot 10^{-7} \\ -7.827149453153203563 9 \cdot 10^{-12} \\ 1.279461987409796483 5 \cdot 10^{-10} \end{pmatrix}$$

$$pH := -\log(|[H^+]|) = \begin{pmatrix} 2.699 - 1.364i \\ 6.902 - 1.364i \\ 11.106 - 1.364i \\ 9.893 \end{pmatrix}$$ Правильное решение

Рис. 3.35. Решения системы уравнений через аналитический вызов функции `Find`

Но при усложнении задачи — при снятии с нее некоторых ограничений и это не помогает.

Радикальным средством решения подобных задач остается программирование. На рис. 3.36 показано, как с опорой на метод половинного деления, который мы уже рассматривали (см. рис. 3.7 и 3.9), задаются граничные значения pH (`1` и `14`), а далее ведутся приближения к решению до тех пор, пока (`while`) не будет достигнута нужная точность расчета.

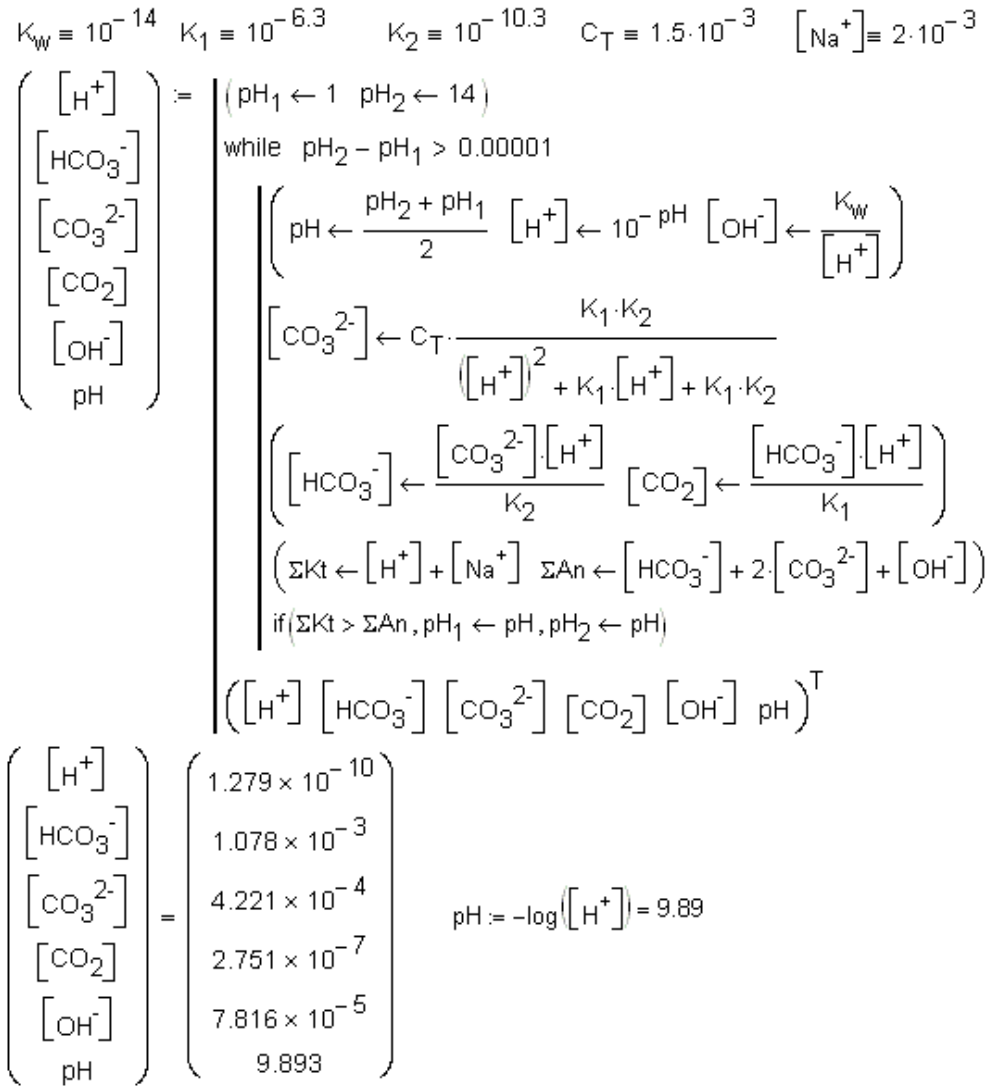


Рис. 3.36. Решения системы уравнений через программирование

Программное решение системы нелинейных алгебраических уравнений имеет то преимущество, что через этот инструмент можно не только правильно рассчитать одну точку (рис. 3.36), но и легко создать функцию пользователя и, если нужно, отобразить ее графически (рис. 3.37).

$$K_w(T) \equiv 10^{\frac{-4780.13}{T\cdot K^{-1}} - 0.019559\cdot\left(T\cdot K^{-1}\right) + 7.8560}$$

$$K_1(T) \equiv 10^{\frac{-2513.98}{T\cdot K^{-1}} - 0.023122\cdot\left(T\cdot K^{-1}\right) + 8.9632}$$

$$K_2(T) \equiv 10^{\frac{-3287.20}{T\cdot K^{-1}} - 0.028377\cdot\left(T\cdot K^{-1}\right) + 9.1324} \qquad C_T \equiv 1.5\cdot 10^{-3} \quad \left[Na^+\right] \equiv 2\cdot 10^{-3}$$

$$pH(T) := -\log\left[\begin{array}{l} \left(pH_1 \leftarrow 1 \quad pH_2 \leftarrow 14\right) \\ \text{while} \quad pH_2 - pH_1 > 0.00001 \\ \quad \left(pH \leftarrow \frac{pH_2 + pH_1}{2} \quad \left[H^+\right] \leftarrow 10^{-pH} \quad \left[OH^-\right] \leftarrow \frac{K_w(T)}{\left[H^+\right]}\right) \\ \quad \left[CO_3^{2-}\right] \leftarrow C_T \cdot \frac{K_1(T)\cdot K_2(T)}{\left(\left[H^+\right]\right)^2 + K_1(T)\cdot\left[H^+\right] + K_1(T)\cdot K_2(T)} \\ \quad \left(\left[HCO_3^-\right] \leftarrow \frac{\left[CO_3^{2-}\right]\cdot\left[H^+\right]}{K_2(T)} \quad \left[CO_2\right] \leftarrow \frac{\left[HCO_3^-\right]\cdot\left[H^+\right]}{K_1(T)}\right) \\ \quad \left(\Sigma Kt \leftarrow \left[H^+\right] + \left[Na^+\right] \quad \Sigma An \leftarrow \left[HCO_3^-\right] + 2\cdot\left[CO_3^{2-}\right] + \left[OH^-\right]\right) \\ \quad \text{if}\left(\Sigma Kt > \Sigma An, pH_1 \leftarrow pH, pH_2 \leftarrow pH\right) \\ \left[H^+\right] \end{array}\right]$$

$pH(293K) = 10.01$ $\quad T := 273K..\,373K$ $\quad T := 373K..\,573K$

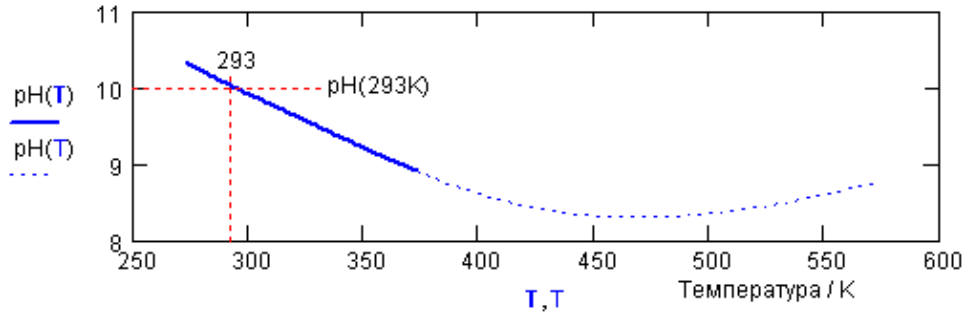


Рис. 3.37. Создание функции пользователя через решение системы уравнений

Кстати о точности. Беда блока `Given/Find` в том, что точность расчета, хранимая в переменной `CTOL`, одинаково прилагается ко всем уравнениям системы, зажатым между ключевым словом `Given` и функцией `Find`. В реальной же практике у разных уравнений должна быть разная степень приближения их к тождествам, в которые в принципе должны превращаться уравнения при постановке в них найденных значений неизвестных. Уравнения можно постараться "уравнять" по отношению к переменной `CTOL`, введя в них *нормирующие множители*. Но и после этого, повторяю, все равно очень часто приходится отказываться от встроенных средств и прибегать к разработке индивидуальных расчетных методик.

Дивертисмент.

Системы уравнений в классической литературе

В классической литературе можно встретить задачи, решение которых требует привлечения уже не калькулятора, выполняющего отдельные арифметические действия (сложение/вычитание, умножение/деление, взятие процента и т.д.), а компьютера, способного решать более сложные задачи за счет программной реализации алгоритмов – последовательного или циклического выполнения сложения, вычитания и других арифметических и прочих действий.

Рассмотрим три задачи из классической отечественной и зарубежной литературы.

Рассказ А.П. Чехова «Репетитор» начинается с такого диалога:

— Теперь по арифметике... Берите доску. Какая следующая задача?

Петя плюет на доску и стирает рукавом. Учитель берет задачник и диктует:

— "Купец купил 138 арш. черного и синего сукна за 540 руб. Спрашивается, сколько аршин купил он того и другого, если синее стоило 5 руб. за аршин, а черное 3 руб.?" Повторите задачу.

Петя повторяет задачу и тотчас же, ни слова не говоря, начинает делить 540 на 138.

— Для чего же вы это делите? Постойте! Впрочем, так... продолжайте. Остаток получается? Здесь не может быть остатка. Дайте-ка я разделю!

Зиберов делит, получает 3 с остатком и быстро стирает.

"Странно... — думает он, ероша волосы и краснея. — Как же она решается? Гм!.. Это задача на неопределенные уравнения, а вовсе не арифметическая..."

Учитель глядит в ответы и видит 75 и 63.

"Гм!.. странно... Сложить 5 и 3, а потом делить 540 на 8? Так, что ли? Нет, не то".

— Решайте же! — говорит он Пете.

— Ну, чего думаешь? Задача-то ведь пустяковая! — говорит Удодов Пете. — Экий ты дурак, братец! Решите уж вы ему, Егор Алексеич.

Егор Алексеич берет в руки грифель и начинает решать. Он заикается, краснеет, бледнеет.

— Это задача, собственно говоря, алгебраическая, — говорит он. — Ее с иксом и игреком решить можно. Впрочем, можно и так решить. Я, вот, разделил... понимаете? Теперь, вот,

надо вычесть... понимаете? Или, вот что... Решите мне эту задачу сами к завтраму... Подумайте...
Петя ехидно улыбается. Удодов тоже улыбается. Оба они понимают замешательство учителя. Ученик VII класса еще пуще конфузится, встает и начинает ходить из угла в угол.
— И без алгебры решить можно, — говорит Удодов, протягивая руку к счетам и вздыхая. — Вот извольте видеть...
Он щелкает на счетах, и у него получается 75 и 63, что и нужно было.
— Вот-с... по-нашему, по-неученому.

Такую систему двух алгебраических линейных уравнений с двумя неизвестными ($x + y = 138$ и $5x + 3y = 540$) можно решить без компьютера и даже без калькулятора, выполнив несложные арифметические действия. На счетах (как Удодов-старший) это сейчас, конечно, мало кто способен сделать. Мы эту задачу решим на компьютере (на современных «счетах»), воспользовавшись самым популярным «суперкалькулятором» – программой Mathcad – см. рис. 1д.

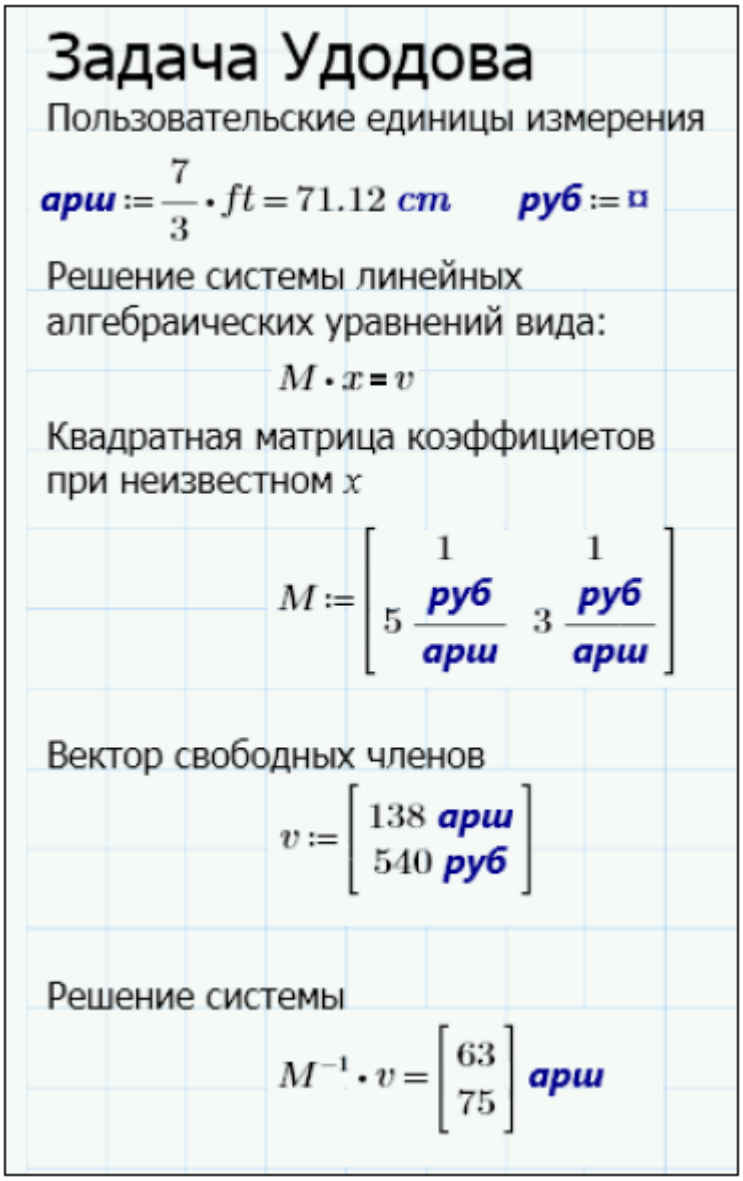


Рис. 1д

Программа Mathcad работает не просто с величинами, а с физическими величинами (длинна, масса, стоимость и т.д.): в программу Mathcad *вши-ты* единицы измерения: (метры, килограммы и т.д.). Но аршина как и многих других устаревших (национальных) единиц измерения в Mathcad, естественно, нет. Тут можно вспомнить (или найти в Интернете), что ар-

шин равнялся трети сажени, которая в свою очередь равна семи футам («семь футов под килем!»). Английский же фут (ft) встроен в Mathcad – см. первый оператор (:=) на рис. 1, с помощью которого в расчет вводится *пользовательская* единица длины – арш (аршин – семь третей фута). Второй оператор присваивания := вводит в расчет пользовательскую единицу стоимости руб, связывая ее с встроенной «нейтральной» (не $, € и т.д.) единицей – ¤[1]. Можно отметить, что задачу Удодова, как и

[1] Пакету Mathcad можно «дать национальность» – сразу настроить его на рубли, например. Тогда второй оператор, показанный на рис. 1д, не понадобиться.

Многие другие подобные задачи, можно решить и без привлечения единиц измерения (так, например, будет решена вторая задача). Но единицы измерения, во-первых, повышают наглядность задачи, уровень ее комментирования и, во-вторых и главных, снижают риск ошибки в решении, когда, например, аршины пытаются сложить с рублями.

На рис. 1д показано, как задача Удодова решается в среде Mathcad с использованием операторов работы с векторами и матрицами. Для этого в расчет вводится и заполняется квадратная матрица M, хранящая коэффициенты при неизвестных системы уравнений, и вектор v – вектор свободных членов. Решение задачи (63 и 75 аршин сукна) находится через перемножение инвертированной (обратной) матрицы M-1 на вектор v.

В задаче Удодова число уравнений *равно* числу неизвестных. Но в жизни и, соответственно, в литературе так бывает далеко не всегда. Два последующих примера тому иллюстрация.

Как понимает читатель, решение, показанное на рис. 1д, годится для системы n линейных уравнений, где n может быть больше двух – равняется семи, например, что будет показано в следующей задаче.

Из повести Ф.М. Достоевского «Игрок» можно «выудить» семь цитат, в которых зашифрованы курсы европейских валют времена написания повести:

Цитата 1

— Вы их немедленно получите, — ответил генерал, покраснев немного, порылся у себя в бюро, справился в книжке, и оказалось, что за ним моих денег около ***ста двадцати рублей****.*

— Как же мы сосчитаемся, — заговорил он, — надо переводить на талеры. Да вот возьмите ***сто талеров****, круглым счетом, — остальное, конечно, не пропадет.*

и далее

Вам следует дополучить с меня эти четыре ***фридрихсдора*** *и* ***три флорина*** *на здешний расчет.*

Цитата 2

Полина просто рассердилась, когда я передал ей всего только ***семьсот гульденов****.*

и далее

Слушайте и запомните: возьмите эти ***семьсот флоринов*** *и ступайте играть, выиграйте мне на рулетке сколько можете больше; мне деньги во что бы ни стало теперь нужны.*

Цитата 3
*Я начал с того, что вынул **пять фридрихсдоров**, то есть **пятьдесят гульденов**, и поставил их на четку.*
Цитата 4
*— Да-с, вот взяла да и выиграла двенадцать тысяч флоринов! Какое двенадцать, а золото-то? С золотом почти что **тринадцать** выйдет. Это сколько по-нашему? Тысяч шесть, что ли, будет?*
*Я доложил, что и за **семь** перевалило, а по теперешнему курсу, пожалуй, и до **восьми** дойдет.*
Цитата 5 *— Oui, madame, — вежливо подтвердил крупер, — равно как всякая единичная ставка не должна превышать разом **четырех тысяч флоринов**, по уставу, — прибавил он в пояснение.*
*Я поставил самую большую позволенную ставку, в **четыре тысячи гульденов**, и проиграл.*
Цитата 6
*Ей проходилось получить ровно **четыреста двадцать фридрихсдоров**, то есть **четыре тысячи флоринов** и **двадцать фридрихсдоров**.*
Цитата 7
*— Полина, вот **двадцать пять тысяч флоринов** — это **пятьдесят тысяч франков**, даже больше.*

В «Игроке» Ф.М. Достоевского «зашифрована» система семи уравнений с пятью неизвестными, решив которую, можно узнать курс валют (талер, фридрихсдор, флорин, гульден и франк) по отношению к рублю:

120 рублей = 100 талеров + 4 фридрихсдора + 3 флорина
700 гульденов = 700 флоринов
5 фридрихсдоров = 50 гульденов
13 000 флоринов = 8 000 рублей
4 000 флоринов = 4 000 гульденов
420 фридрихсдора = 4 000 флоринов + 20 фридрихсдоров
25 000 флоринов = 50 000 франков

Если число уравнений превышает число неизвестных, то такая система уравнений называется *переопределенной*.

Задачу «Игрока» тоже можно решить без компьютера, последовательно высчитывая курсы отдельных валют (гульден равен флорину, фридрихсдор равен десяти гульденам и т.д.), а в конце решения определить их стоимость по отношению к рублю. Но мы, как и в случае с первой задачей, обратимся к помощи компьютера. Пусть он сам упрощает, что считает нужным.

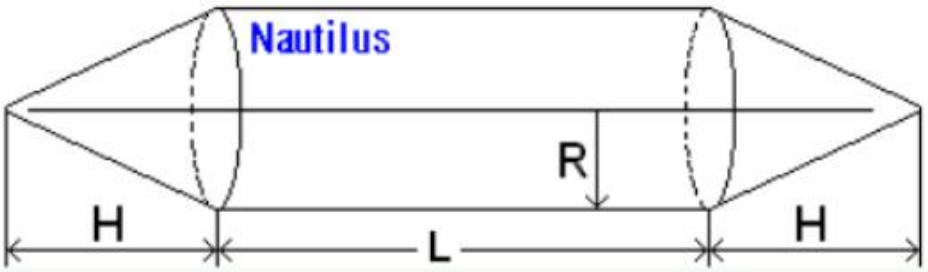
Рис. 2д

На рис. 2д показано решение в среде Mathcad задачи о курсах валют с использованием тех же матричных операторов. Но «решающий» оператор `M-1 · v` (перемножение инвертированной матрицы коэффициента при неизвестных системы на вектор свободных членов) здесь работать не будет, так как матрица `M` в данном случае не квадратная и, следовательно, ее нельзя инвертировать. Тут на помощь приходит встроенная в Mathcad функция `lsolve`, которая также предназначена для поиска решения системы линейных алгебраических уравнений, но может работать и с неквадратными матрицами коэффициентов при неизвестных. Из решения, показанного на рис. 2д видно, что во второй половине XIX века гульден стоил 61 копейку, талер — полтора гульдена (флорина), а фридрихсдор был в 10 раз дороже гульдена. При таком раскладе генерал заплатил учителю (Алексею Ивановичу) 117 руб. 73 коп («около 120 руб.» — см. цитату 1), а «бабуленька» выиграла на рулетке 7930 руб. («до восьми тысяч дойдет» — см. цитату 4).

Если же число уравнений алгебраической системы меньше числа неизвестных, то такая система называется *недоопределенной*. Ее можно «увидеть» в описании подводной лодки «Наутилус» – «героине» романов Жюль Верна «Двадцать тысяч лье под водой» и «Таинственный остров». Вот что можно узнать из разговора капитана Немо с профессором Аронаксом о размерах подводной лодки: "Вот, господин Аронакс, чертежи судна, на котором вы находитесь. Судно представляет собой сильно удли-

ненный цилиндр с коническими концами. <...> Площадь его равняется одной тысяче одиннадцати и сорока пяти сотым квадратных метров, объем равен одной тысяче пятистам и двум десятым кубических метров; короче говоря, корабль, полностью погруженный в воду, вытесняет тысячу пятьсот и две десятых кубических метров, или тонн, воды.»

Если принять допущение, что центральная часть подводной лодки – это круглый прямой цилиндр, а ее нос и корма – это круглые прямые конусы одинаковой высоты (вернее, длины – см. чертеж на рис. 3д), то задача сводится к решению двух нелинейныхз алгебраических уравнений с тремя неизвестными (недоопределенная задача!).



Solve

$R := 4\ m \qquad L := 70\ m$

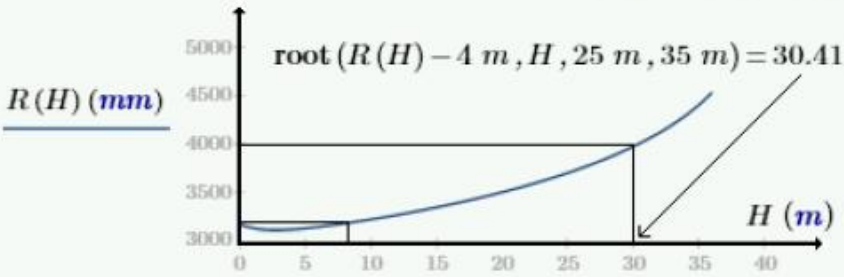
$\pi \cdot R \cdot \sqrt{R^2 + H^2} + 2 \cdot \pi \cdot R \cdot L + \pi \cdot R \cdot \sqrt{R^2 + H^2} = 1011.45\ m^2$

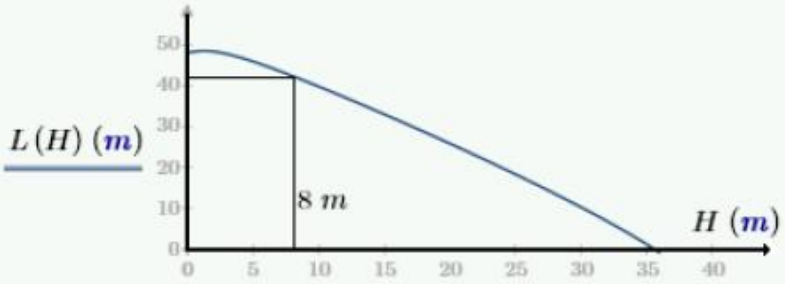
$\frac{1}{3} \cdot \pi \cdot R^2 \cdot H + \pi \cdot R^2 \cdot L + \frac{1}{3} \cdot \pi \cdot R^2 \cdot H = 1500.2\ m^3$

$Size(H) := \text{Find}(R, L)$

$R(H) := Size(H)_0 \qquad L(H) := Size(H)_1 \qquad H := 0\ m, 10\ cm .. 36\ m$

$\text{root}(R(H) - 4\ m, H, 25\ m, 35\ m) = 30.41\ m$





Одно из решений

$H := 8\ m \qquad R(H) = 3171\ mm \qquad L(H) = 42.16\ m$

Рис. 3д

Для решения систем нелинейных уравнений в среде Mathcad есть специальный блок `Solve` (см. рис. 3д), в котором нужно записывать, во-первых, первое приближение к решению (мы тут можем предположить, что радиус цилиндрической части лодки (`R`) равен 4 м, а длина этого цилиндра (`H`) – 70 м), во-вторых, саму систему уравнений, определяемой геометрией лодки, и, в-третьих, встроенную в Mathcad функцию `Find` (Найти), которая возвращает значения своих аргументов (у нас в задаче это переменные `R` и `H`), превращающие систему уравнений в систему тождеств. Так как наша система уравнений недоопределена, то функция `Find` будет возвращать не конкретные значения переменных `R` и `L`, а две функции `R(H)` и `L(H)`, связывающие радиус и высоту цилиндра лодки с высотой конусов – с длиной носа (кормы) лодки. Эти две зависимости на рис. 3д отображены графически.

Из первого графика видно, что радиус цилиндрической части лодки может меняться в пределах от 3 до 4,5 метров. Короче говоря, задача Жюль Верна о размерах «Наутилуса» может иметь множество решений. Одно из них (H = 8 м, R = 3,18 м и L = 42,16 м) показано в конце решения на рис. 3 и отмечено линиями на графиках. Дополнительно с помощью встроенной в Mathcad функции `root` (корень) найдено и второе решение: если б радиус цилиндра лодки равнялся 4 м, то длины концов должны быть равны 30,41 м.