Глава 2. Вычислительный ренессанс: Метод Ньютона

Я <…> решал на черной доске какое-то длинное алгебраическое уравнение. В одной руке я держал изорванную мягкую «Алгебру» Франкера, в другой — маленький кусок мела, которым испачкал уже обе руки, лицо и локти...

Лев Толстой «Юность»

Основной упрек, бросаемый обществом в адрес школы, – это её оторванность от жизни. Мол, в школе учат одному, а в жизни приходится сталкиваться совсем с другим.

Если говорить о школьной математике, то этот упрек конкретизируется так. В школе учат *аналитическим* (*точным*) методам решениям надуманных уравнений и их систем, а в жизни к реальным, а не недуманным уравнениям, отображающим реальные физические законы и процессы (см., например, рис. 1.2 и 1.5 в главе 1 с подобными уравнениями), приходится применять *численные* (*приближенными*) методы решений, о которых в школе даже не упоминают.

В культовом итальянском фильме "Амаркорд" (1973 год – рис. 2.1) учительница объясняет ученику, как нужно решать алгебраическое уравнение с неизвестной величиной x, преобразовав его к виду $a\ x^2 + b\ x + c = 0$. Школьнику следует найти численные значения коэффициентов a, b и c, подставить их в известную формулу корней квадратного уравнения с дискриминантом под квадратным корнем и получить ответ. И никак иначе! Вывод формулы решения квадратного уравнения – это отдельная тема на уроках школьной математики. Где-то пытаются вывести подобные формулы для кубического уравнения (формула Кардано) и даже для уравнения четвертой степени, декларирую при этом что пятая степень – это предел возможностей, некий тупик!

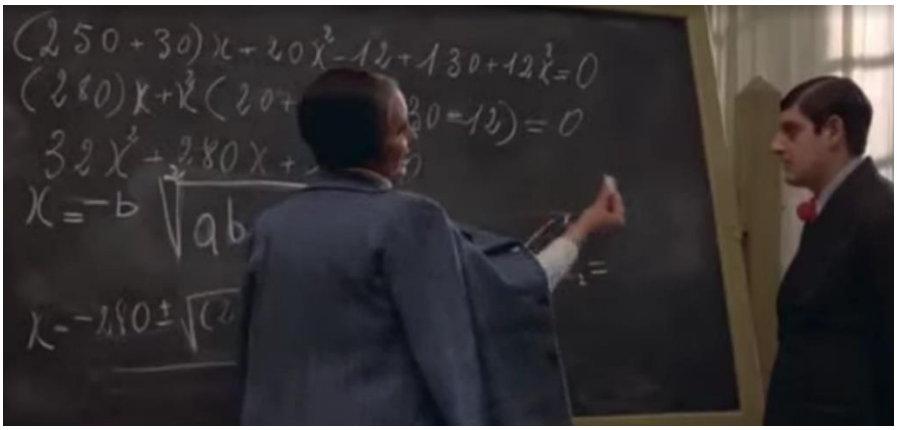


Рис. 2.1. Кадр из фильма "Амаркорд": решаем квадратное уравнение

Раньше численные методы решения алгебраических уравнений в школе не рассматривались потому, что они были очень трудоемкими даже при работе с электронным калькулятором. Но в настоящее время в школах стала доступна вычислительная техника с современными математическими программами – с SMath, например, которые можно использовать не только на уроках информатики, но и на уроках математики для численного решения уравнений и их систем. Более того, учитель в классе может писать на электронной доске не мелом, а специальным пером, вводя в расчет живые формулы – формулы, по которым можно считать, строить графики, делать анимацию…

Мы пока затрагиваем только алгебраические уравнения. Но то, о чем будет сказано ниже, касается и дифференциальных уравнений см. главу 10 и [1].

При этом наблюдается некий ***вычислительный ренессанс***. Пользователи современных быстродействующих компьютеров возвращаются к простым и понятным («старым добрым») алгоритмам, которые раньше применялись редко из-за медленного счета ЭВМ первых поколений.

Давайте за пару минут установим на компьютер отечественную свободно распространяемую математическую программу SMath (www.smath.com) и с её помощью покажем, как можно численно найти нуль функции или решить систему уравнений (найти её корни) простейшим и наглядным *методом Ньютона*. Напомним, что нуль функции – это такое значение аргумента, при котором функция равна нулю, а её график пересекает ось абсцисс (см. рис. 2.3, 2.6 и 2.8). Корень же системы уравнений – это значение неизвестных, превращающих уравнения в тождества. Если же оперировать двумя уравнениями, преобразовав их в функции переносом правых частей в левые, то корень такой системы – это координаты точки пересечение графиков функций (см. рис. 2.12, 2.13 и 2.17).

В расчетах мы будем использовать булевый оператор «примерно равно», который есть только в среде SMath (рис. 2.2). Он работает в паре с оператором «не примерно равно» (отрицание + примерно равно) и настраивается через диалоговое окно Сервис / Опции / Вычисления. Число три, стоящее в окне ввода на рис. 2.2 означает, что два вещественных числа, стоящих в операторе «примерно равно», должны отличаться по модулю друг от друга не более, чем на 10^{-3}. Внизу рисунка 2.2 показано, что число π равно примерно 3.142 (ответ 1 – булевое «да»), а не 3.14 (0 – булевое «нет»), если оперировать точностью, равной одной тысячной.

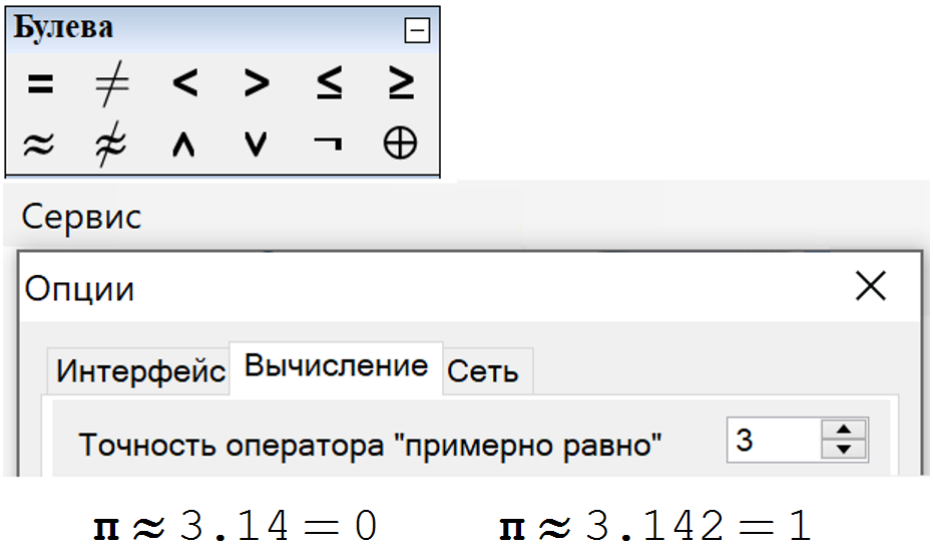


Рис. 2.2. Настройка оператора «примерно равно» и два примера его вызова

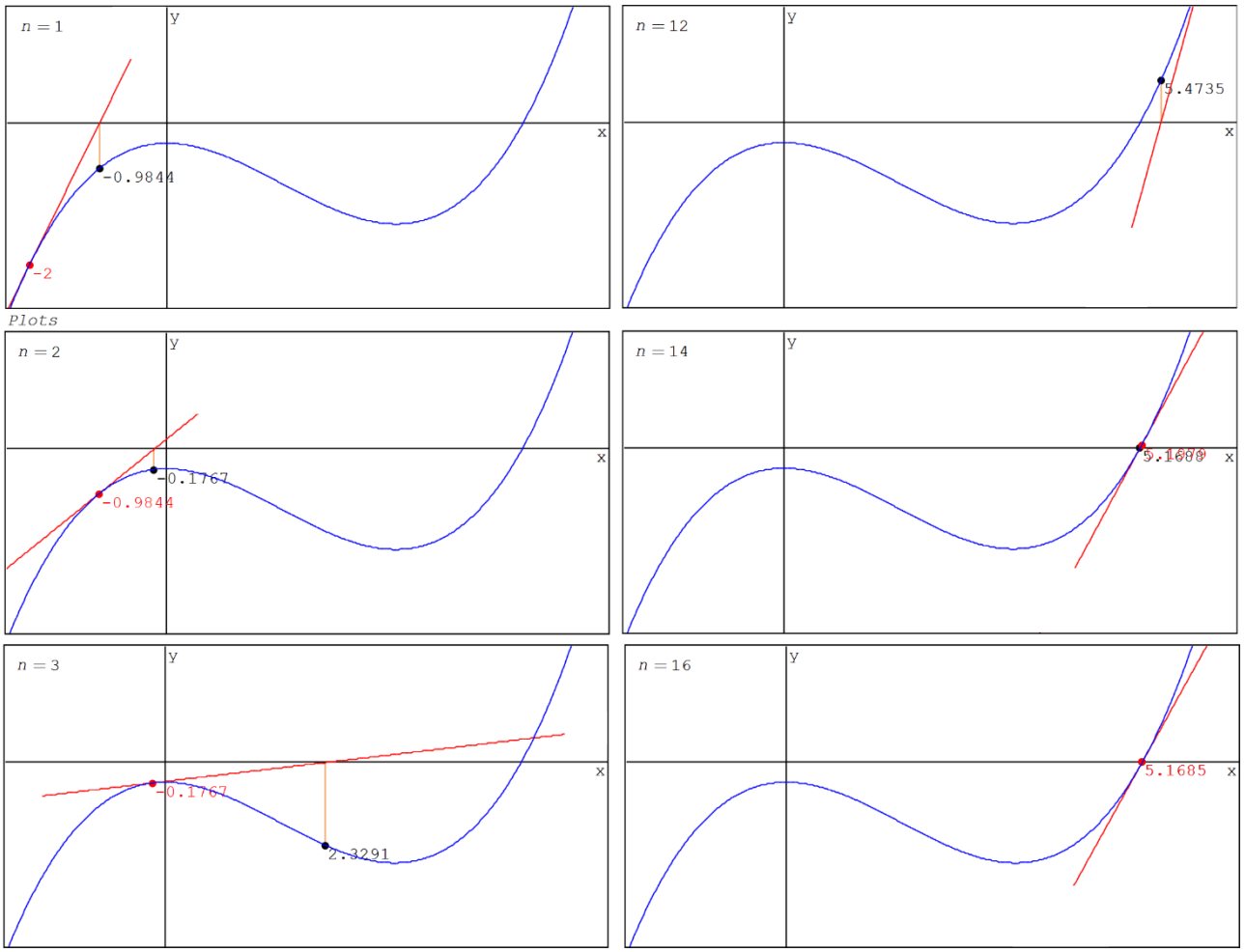
На рисунке 2.3 средствами графики отображена суть метода Ньютона по принципу «Лучше один раз увидеть, чем сто раз услышать». Задается анализируемая функция $y(x)$ – кубический полином[1] и первое предположение ($x_1 = -2$) – начальная точка на графике анализируемой функции, через которую проводится касательная. Альтернативное название метода Ньютона – это *метод касательных*. Исаак Ньютон (1642-1727) вместе с Готфридом Лейбницем (1646-1716) стоял у истоков

[1] Здесь специально взят кубический, а не квадратный полином (как на рис. 1) с неким горбом, через который будет несколько раз "перекатываться" наша касательная, пока она не переберётся к решению.

дифференциального исчисления[2] с его производной, касательной, интегралом (см. рис. 1.10 в главе 1) и прочим, упомянутом в учебнике эпиграфа. Наша касательная пересекает ось абсцисс в точке $x_2 = -0.9844$, которая будет считаться вторым приближением. Через эту точку проводится новая касательная, которая пересекает ось абсцисс в точке -0.1767. Эти действия повторяются до тех пор, пока значение функции в очередной точке пересечения касательной с осью абсцисс станет равным нулю (примерно нулю!), и когда наши две точки сольются в одну и обе окажутся на оси абсцисс. В нашем расчете для этого понадобилось 16 приближений (итераций), шесть из которых (n=1, 2, 3, 12, 14 и 16) отображены на рис. 2.3. На 15-й итерации еще видны две раздельные точки и два раздельных значения их абсцисс – красное и черное[3]. Но на 16-й, последней итерации совпали визуально и сами точки, и эти два значения их абсцисс 5.1685 – красное написано поверх черного. Это и есть искомый нуль функции, найденный с некоторым приближением.

$y(x) := x^3 - 5 \cdot x^2 - 4.5$ $x_1 := -2$ 1 n Нажимаем на галочки и меняем кадры анимации

Расчет



[2] https://ru.wikipedia.org/wiki/Спор_Ньютона_и_Лейбница_о_приоритете

[3] Есть такой роман Стендаля "Le Rouge et le Noir". Там тоже можно узреть некие итерации главного героя между двумя карьерными линиями – военной (красное) и церковной (черное).

Рис. 2.3. Графическое отображение приближения к нулю функции

Функция $y(x)$ и ее первое приближение к нулю x_1 задаются оператором := (присвоить). А номер итерации n задается по-другому – через элемент управление Controls/Numeric up-down, что более удобно в нашей задаче. Если щелкать мышкой по галочкам этого элемента интерфейса (Controls) – обычной и перевернутой, то значение переменной n будет мгновенно меняться вниз или вверх с единичным шагом[4]. Это позволит видеть наши касательные, иллюстрирующие метод Ньютона, почти как в анимации. Но можно создать и настоящую анимацию, сохранить ее в виде gif-файла, например, и показывать отдельно без связки с пакетом SMath, что описано в главе 1 учебного пособия [2].

На рисунке 2.4 показаны операторы, скрытые на рис. 2.3 в свернутой области с именем `Расчет`.

Там, во-первых, аналитически берется производная анализируемой функции, далее вводится уравнение касательной в точке x_1, а затем создается функция с именем $Zero$ (Нуль) с бесконечным циклом (while 1 – «пока рак на горе не свистнет!»), выход из которого (**`break`**) определяется пользователем через задание значения переменной n – число итераций (см. рис. 2.3). Текущий номер итерации хранится в переменной j. В среде американской версии SMath – в Mathcad здесь можно применить оператор Return, которого в SMath пока нет.

Переменная `Plots` хранит данные для построения семи графиков:

- графика самой функции;
- графика ее касательной – прямой линии;
- красной точки размером в 15 единиц предыдущего приближения;
- черной точки размером в 15 единиц текущего приближения;
- вертикальной линии, соединяющей ось ординат с точкой текущего приближения (данные для графика в виде квадратной матрицы);
- красной надписи (с размером шрифта 10) значения аргумента в точке предыдущего приближения;
- черной надписи (с размером шрифта 10) значения аргумента в точке текущего приближения.

[4] Такая настройка возможна после нажатия правой кнопки мыши. Элемент управления Numeric up-down становится доступным после подгрузки соответствующего плагина.

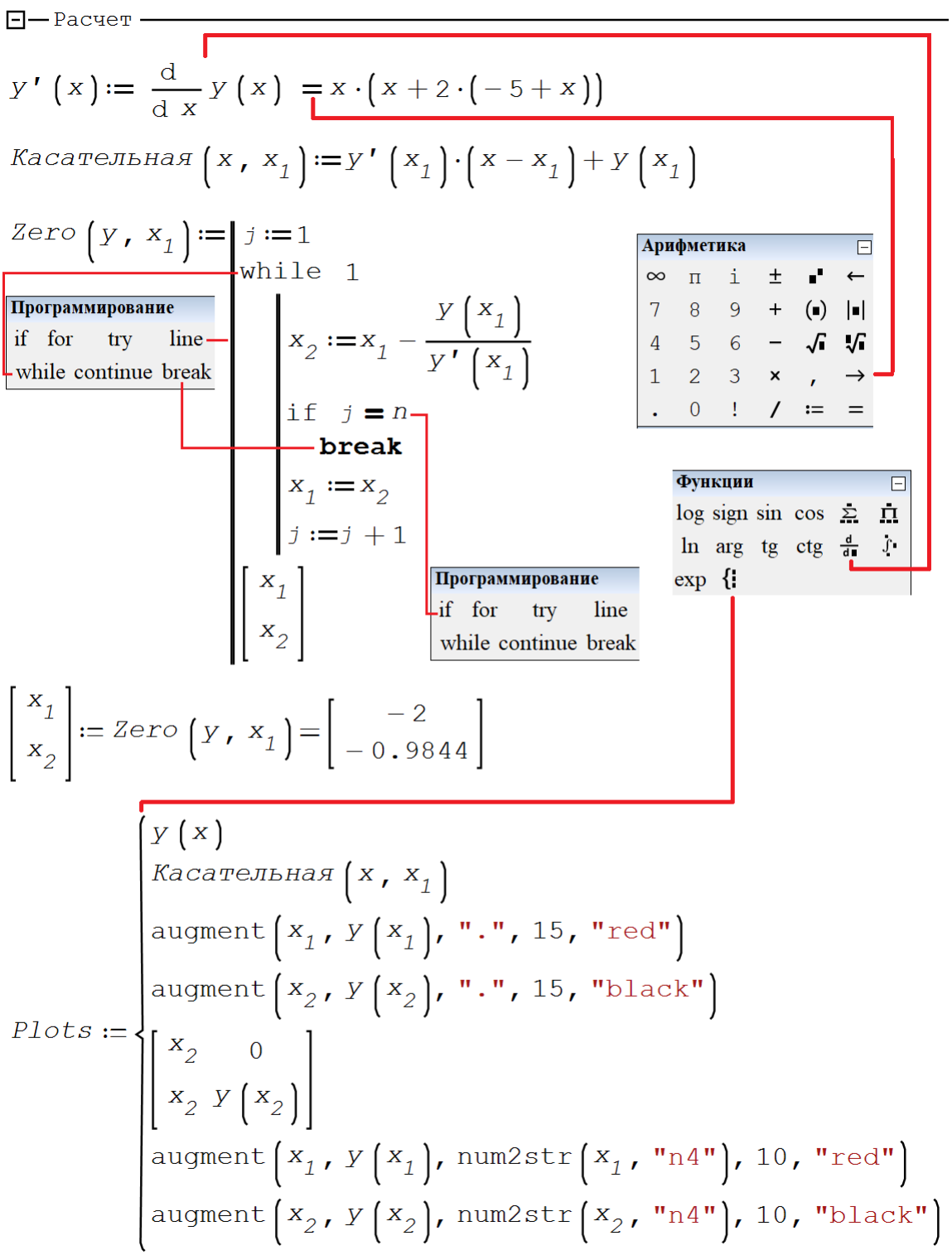


Рис. 2.4. Программа поиска нуля функции методом Ньютона и операторы построения графиков

На рисунке 2.5 можно увидеть график изменения числа итераций в зависимости от значения начального приближения. Мы ограничились диапазоном от минус 20 до плюс 10, но его можно расширить. Или сузить!

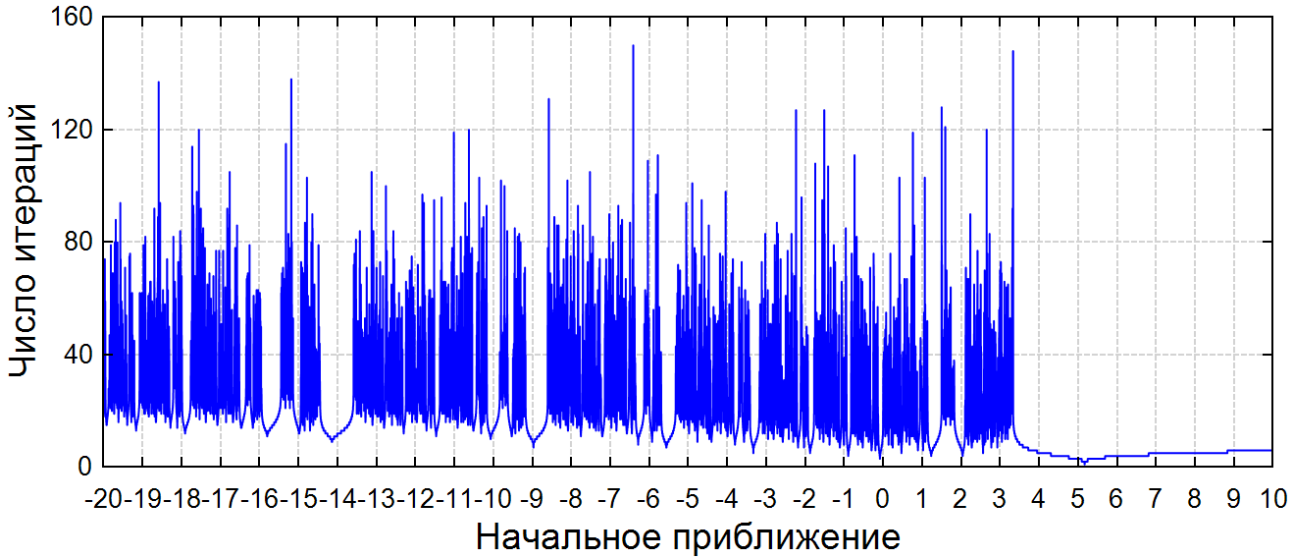


Рис. 2.5. График изменения числа итераций (метод Ньютона) в зависимости от значения начального приближения

Из графика (частокола) на рис. 2.5 видно, что «горб» на кубической параболе резко увеличивает число итераций. Но около искомого корня число итераций резко уменьшается. Правда, если двигаться дальше направо за десятку, то снова появится «частокол».

Функция, у которой ищется нуль, может быть такая сложная, что взятие ее производной выльется в отдельную проблему, занимающую значительные ресурсы компьютера. Одно из решений – это замена касательной на *секущую*, когда задается не одна, а две точки начального предположения. Если искомый нуль находится между двумя этими точками, то поиск нуля функции будет в чем-то походить на еще один широко используемый метод численного поиска нуля функции одного аргумента – на метод *половинного деления* (рис. 2.6).

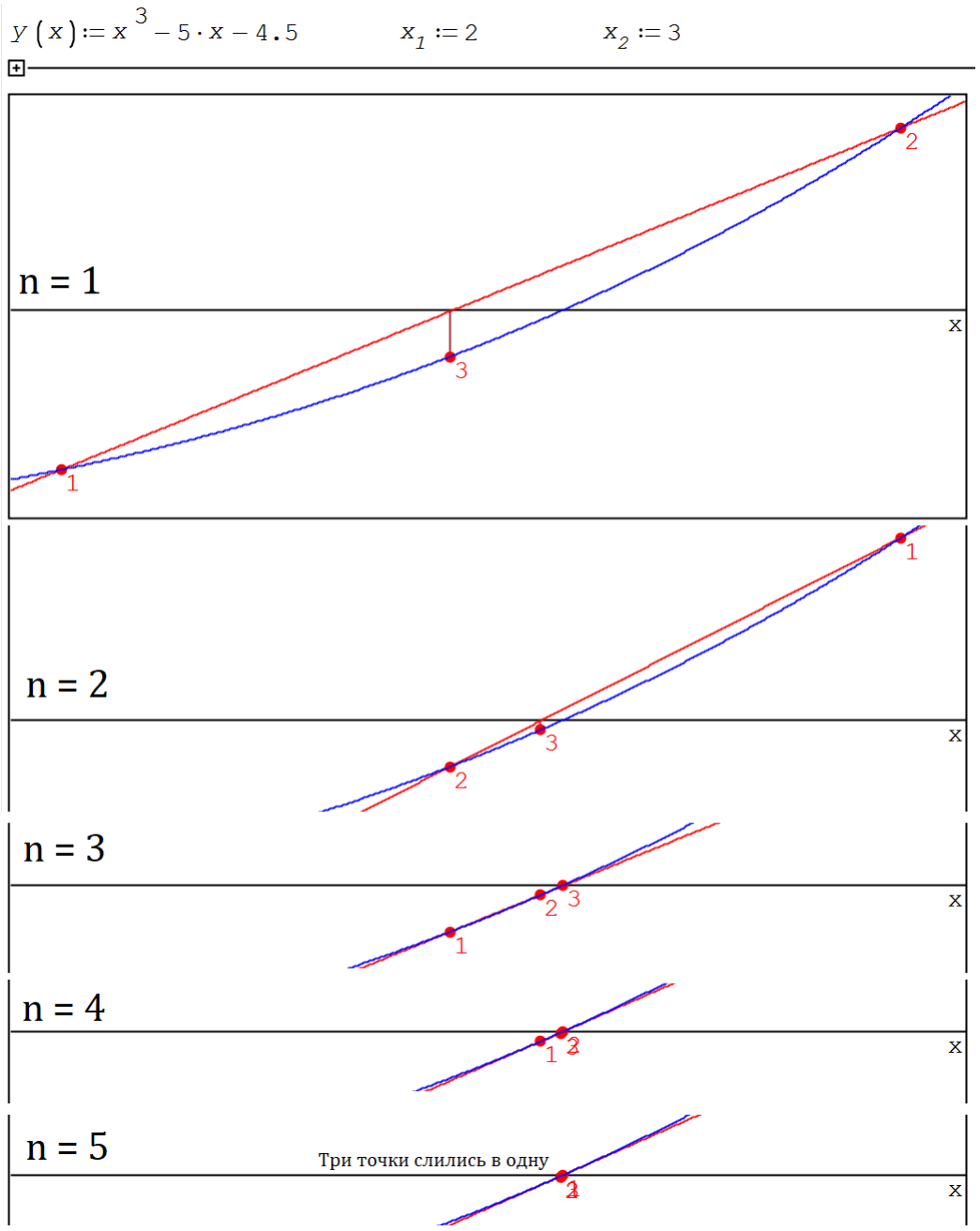


Рис. 2.6. Графическая иллюстрация метода секущих

На рисунке 2.7 показана реализация метода секущих в среде SMath с дополнительными операторами для его графического отображения на рис. 2.6 и 2.8.

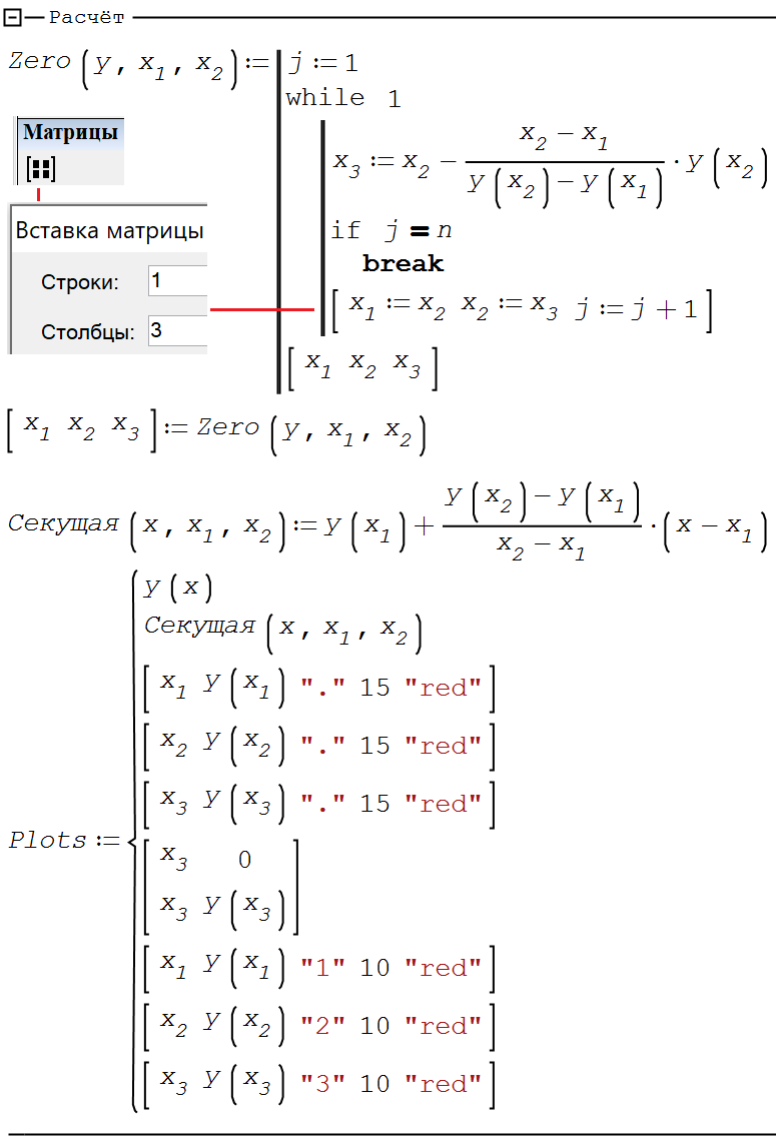


Рис. 2.7. Программа поиска нуля функции методом секущих и операторы построения графиков

Переменная `Plots` хранит данные для построения не семи (рис. 2.3 и 2.6) а уже девяти графиков:

- графика самой функции;
- графика ее секущей – прямой линии;
- трех красных точек размером в 15 единиц текущего x_3, предыдущего x_2 и предыдущего предыдущему x_1 приближениям;
- вертикальной линии, соединяющей ось ординат с точкой текущего приближения x_3;
- трех красных надписей (с размером шрифта 10) значения аргумента в трех точках приближения.

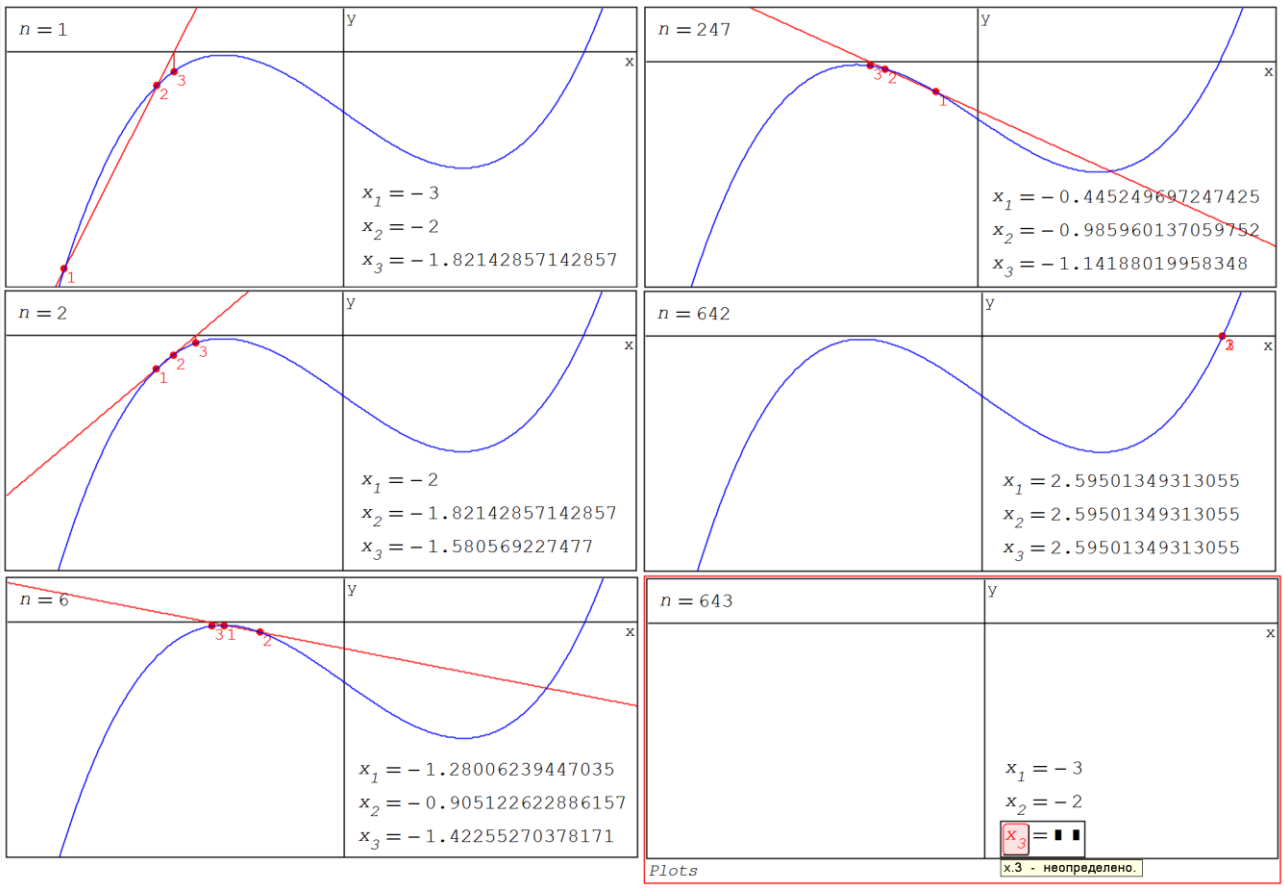


Рис. 2.8. Графическое отображение приближения к нулю функции (метод секущих)

На рисунке 2.9 показан график, аналогичный графику на рис. 2.4, но не для метода Ньютона, а для метода секущих. У этих двух графиков качественно отличаются правые части.

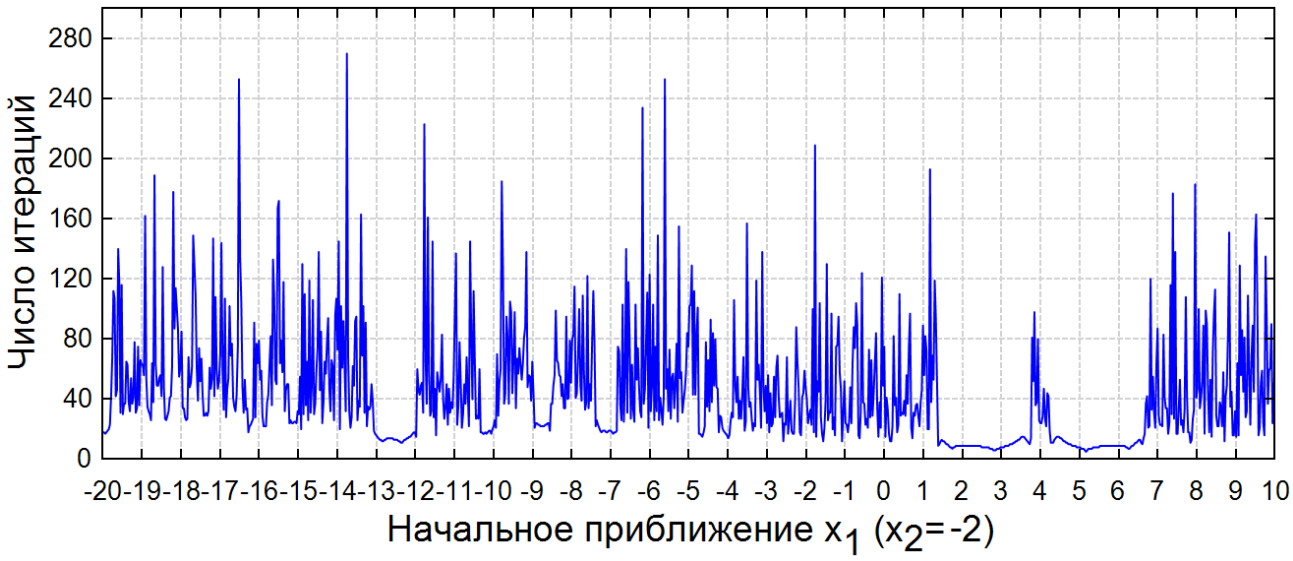


Рис. 2.9. График изменения числа итераций (метод секущих) в зависимости от значения первого начального приближения и фиксированном значении второго приближения

У кубического полинома, который мы использовали выше, нули можно найти аналитически – см. рис. 2.10. Аналитический метод принципиально отличается от численного не только тем, что выдается абсолютно точный ответ в радикалах, но и тем, что выдаются все три корня, включая и комплексные.

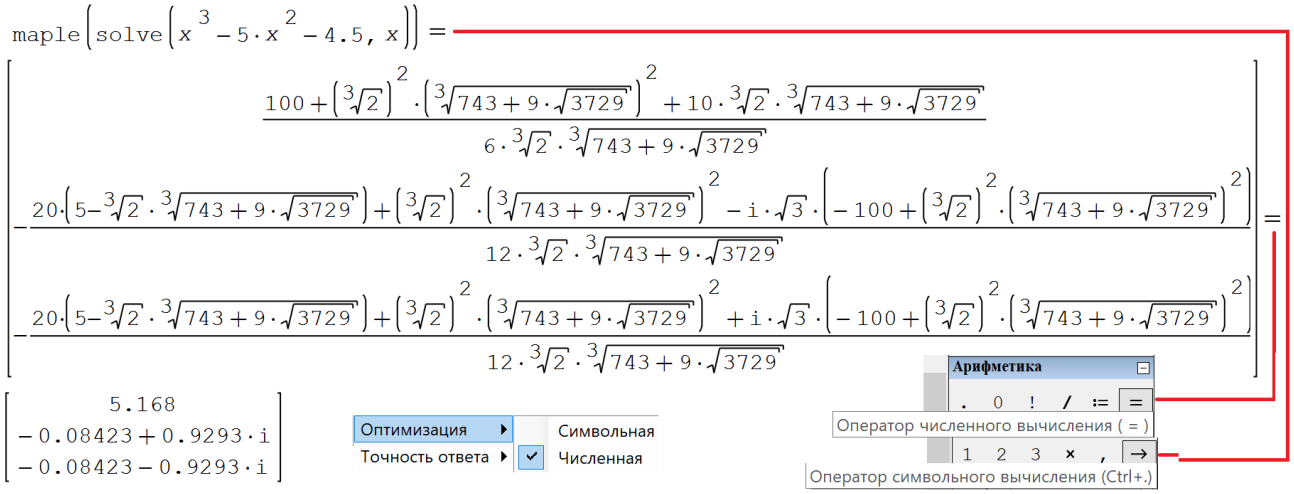


Рис. 2.10. Аналитический поиск нулей кубической параболы

В расчете на рисунке 2.10 задействовано дополнение Maple, подключение которого к среде SMath позволяет вызывать встроенные конструкции математической программы Maple. В частности, вызвана функция `solve` для поиска корней выражения – кубической параболы. Одноименная функция есть и в среде SMath, но она возвращает численный, а не аналитический ответ.

Интересный вопрос: что сделать быстрее и проще – найти нуль функции методом Ньютона или вычислить (опять же приближенно!) сложное выражение, показанное на рис. 2.10, «прихватив» при этом два ненужных комплексных результата. Глядя на корни и дроби рисунка 2.10, можно сказать, что

не только сами математики, но и сама символьная математика[5] часто выдают абсолютно точный и абсолютно бесполезный ответ. Такой ответ математика нужно как-то адаптировать под математический уровень спрашивающего. Ответ символьной математики на рис. 2.10 нужно преобразовывать в десятичное число. А многие задачи по поиску нулей функций вообще не имеют аналитического решения или оно на порядок сложнее того, что показано на рис. 2.10. Так что «думайте сами – решайте сами», что использовать в решении задач – численную или символьную математику.

Метод Ньютона для функции одного аргумента хорошо описан и в бумажной литературе, в интернете. Ничего нового мы здесь не придумали. Мы лишь впервые реализовали его в среде SMath и графически отобразили некую его сходимость – см. рис. 2.5. Более подробно о сходимости метода Ньютона написано здесь https://ru.wikipedia.org/wiki/Метод_Ньютона. А вот для функций двух и более аргументов (для систем уравнений) сведений в интернете почти нет. Восполним этот пробел.

На рисунке 2.11 показано, как в расчет вводится выражение овала Кассини (функция с именем f), а затем на его основе формируются выражения для овала Толстого[6] (f_1 – см. рис. 1.12 в главе 1) и для лемнискаты Бернулли (f_2 – см. рис. 1.14 в главе 1). Эти две замкнутые кривые четвертого порядка показаны ниже на рис. 2.12, где лемниската Бернулли сдвинута слегка вправо и вниз от начала координат на величину 0.5. Далее задаются функция-вектор с именем F и квадратная матрица ее частных производных по двум аргументам. Эта матрица называется матрицей Якоби (J) и высчитывается через встроенную в SMath функцию `Jacobian`. Ниже на рисунке 2.15 мы раскроем содержание этой матрицы. Затем задается первое предположение – значения первых элементов векторов X и Y. Остальные элементы этих векторов задаются в цикле `for` последовательными итерациями. Все это готовится для графического отображения и заносится в переменную $Plots$.

[5] Более правильное название символьной математики – это компьютерная математика аналитических преобразований.

[6] Очков В.Ф., Очкова Н.А. Лев Толстой и математика. – Москва: МПГУ, 2023. – 208 с. (http://www.twt.mpei.ac.ru/ochkov/Tolstoy-Math-3.pdf)

$$f(x, y, a, c) := \left(x^2 + y^2\right)^2 - 2 \cdot c^2 \cdot \left(x^2 - y^2\right) - \left(a^4 - c^4\right)$$ Овал Кассини

$$f_1(x, y) := f\left(x, y, \sqrt{2}, 1\right)$$ Овал Толстого

$$f_2(x, y) := f(x - 0.5, y - 0.5, 2, 2)$$ Лемниската Бернулли

$$F(x, y) := \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} \qquad J(x, y) := \text{Jacobian}\left(F(x, y), \begin{bmatrix} x \\ y \end{bmatrix}\right)$$

$$\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} := \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

$$n := 30 \qquad \text{for } j \in [1..n]$$

$$\begin{bmatrix} X_{j+1} \\ Y_{j+1} \end{bmatrix} := \begin{bmatrix} X_j \\ Y_j \end{bmatrix} - J\left(X_j, Y_j\right)^{-1} \cdot F\left(X_j, Y_j\right)$$

$$Plots := \begin{cases} f_1(x, y) \\ f_2(x, y) \\ \text{augment}(X, Y) \\ \text{augment}(X, Y, ".", 3, "green") \end{cases}$$

Рис. 2.11. Запись шагов поиска корня (решений) системы уравнений

При реализации метода Ньютона для системы двух уравнений нужно будет работать, как уже отмечено, не с одиночной первой производной, а с набором первых производных, который и называется матрицей Якоби. Эта квадратная матрица 2 на 2, повторяем, хранит частные производные наших двух анализируемых функций по двум аргументам (определитель матрицы Якоби называется якобианом). Все остальное дело техники – вычислительной техники – см. на рис. 2.12 зеленые трассировки численного поиска решений из разных начальных точек.

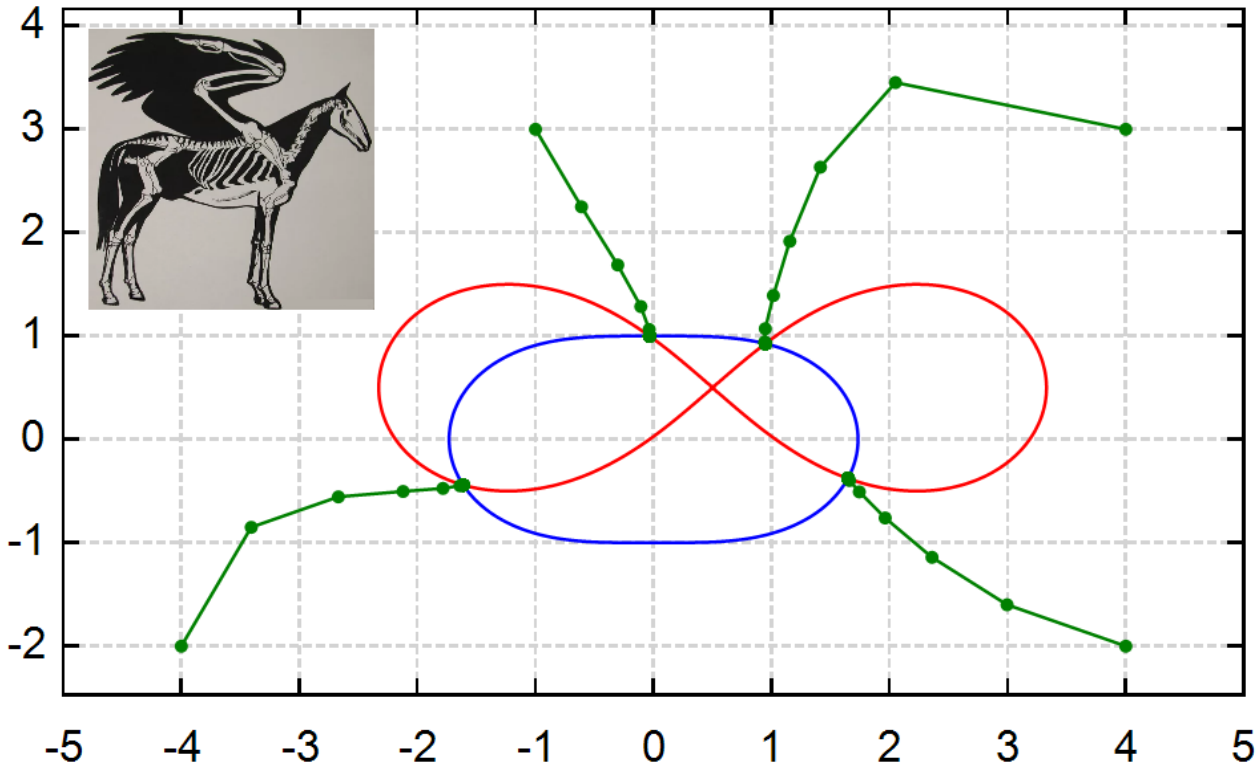


Рис. 2.12. Графическое отображение решения системы двух уравнений методом Ньютона

На рисунке 2.12 показаны случаи, когда точки начального предположения находятся и вне синего овала Толстого, и вне красной лемнискаты Бернулли. Если же эти точки переместить внутрь этих замкнутых кривых, то, казалось бы, для поиска решения потребуется меньшее число итераций – не пять-семь, а две-три. Но не тут-то было – см. рис. 2.13. Собака автора (ее фото на рисунке) ведет себя примерно так, когда ее приглашают к миске с кормом. Она не бежит к угощению по прямой, а медленно делает довольно большой крюк, показывая тем самым, что она вовсе не голодна. Гости нередко поступают таким же образом, когда их приглашают к праздничному столу.

На рисунке 2.13 под первым (верхнем) графиком, показывающим бег авторской собачки около миски с кормом, показан второй (нижний) – увеличение траектории бега вблизи нулевой точки.

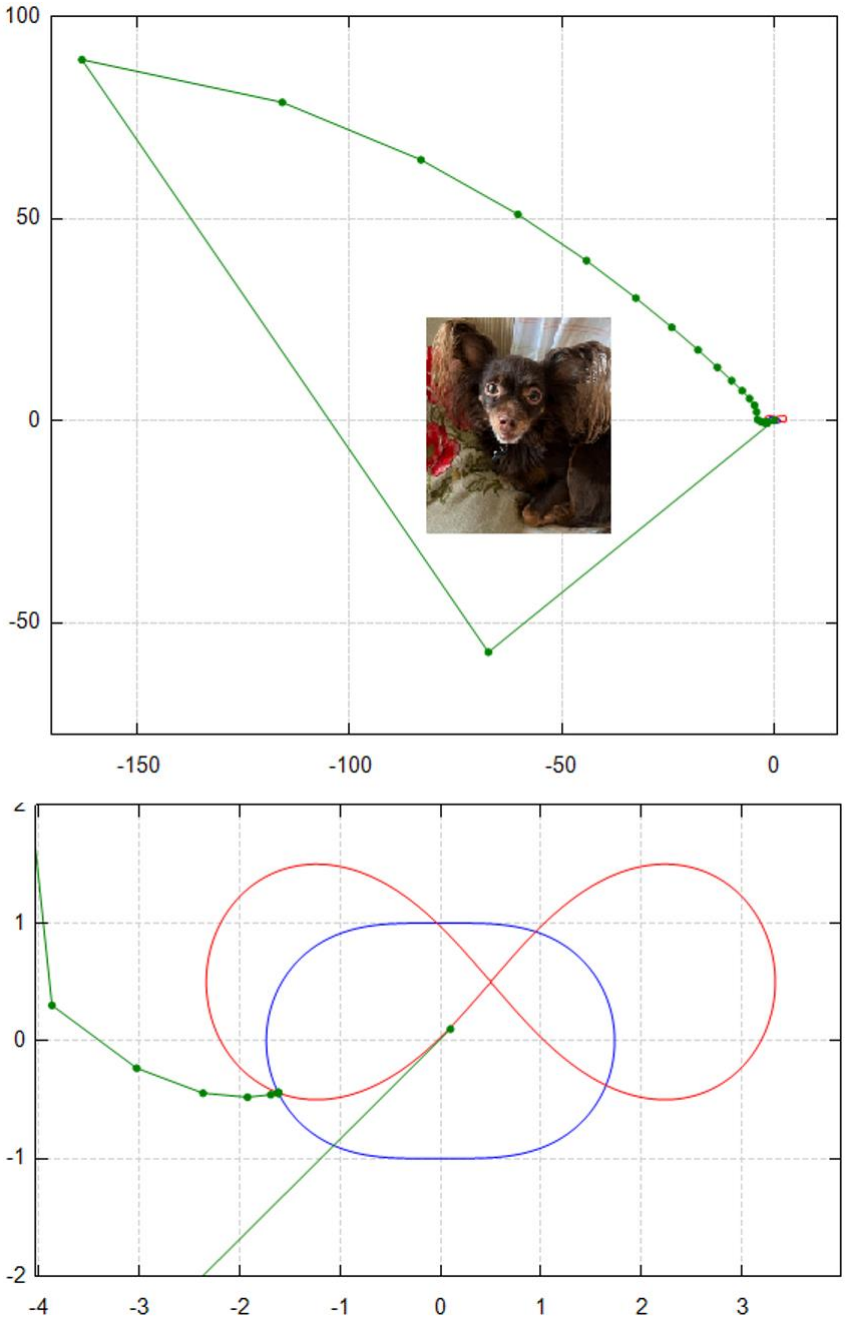


Рис. 2.13. Трассировка особого случая решения системы двух уравнений методом Ньютона: вся траектория расчета и увеличение у начальной и конечной точки

В расчете, показанном на рисунке 2.11, задавалось значение целочисленной переменной *n* – число шагов поиска корня системы. В расчете, показанном на рисунке 2.14, значение переменной *n* не задается, так как число шагов поиска корня системы разное из разных точек первого предположения. Вместо этого используется цикл `for`, у которого не три, а четыре операнда (местодержателя – черных квадратиков). Это гибрид цикла `for` с тремя операндами (см. рис. 2.12) и цикла `while`. В первый операнд помещается параметр `i` цикла и его первое значение (единица), Второй операнд хранит

булево выражение, прерывающее цикл, если выдается ответ 0 (ложь). В третьем операнде задается шаг изменения операнда, а четвертый операнд – это тело цикла.

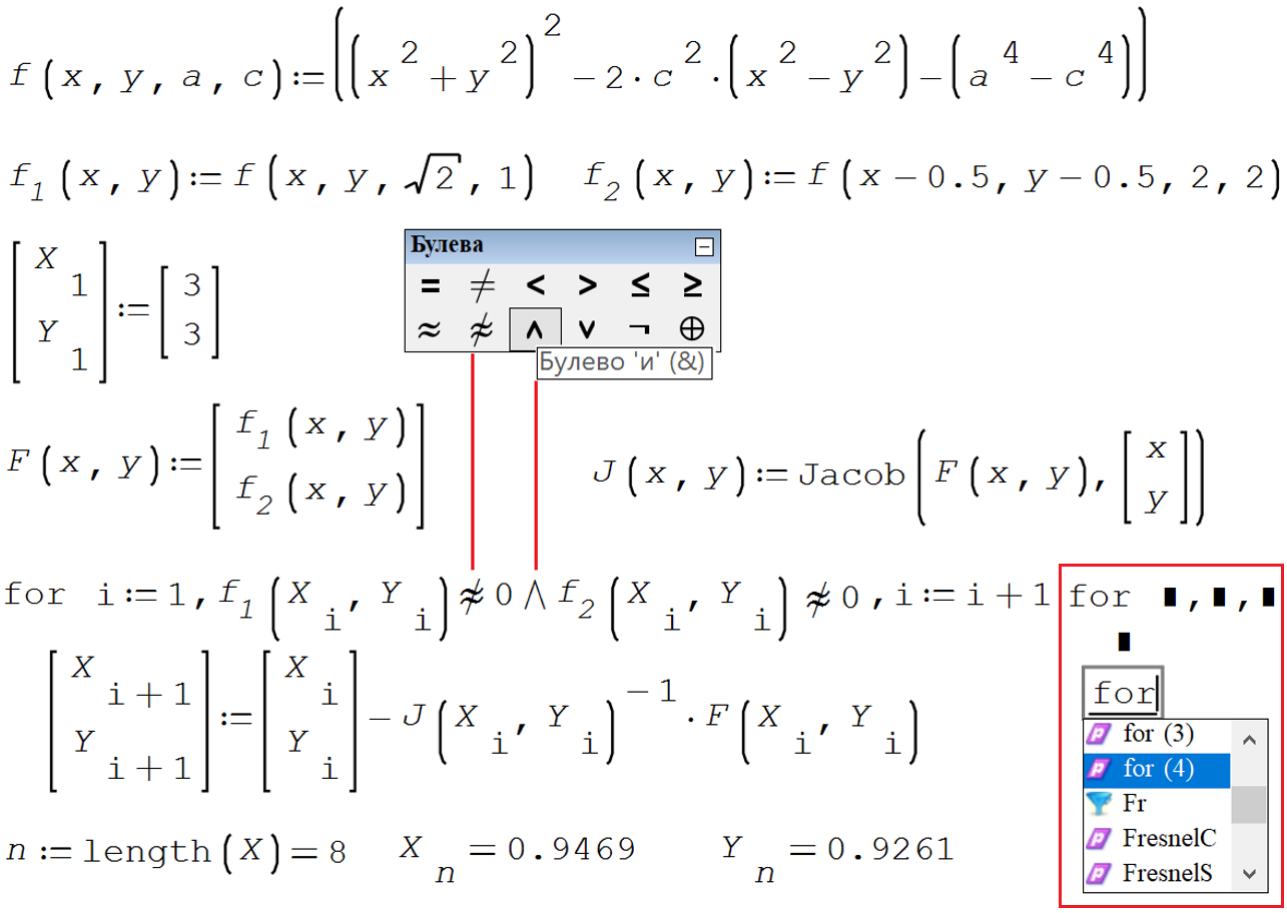


Рис. 2.14. Решение системы двух уравнений методом Ньютона: использование гибрида цикла for и while

Кстати, встроенная в SMath функция `roots` с тремя аргументами, предназначенная для численного решения систем нелинейных уравнений с опорой на первое предположение, была практически бессильна перед овалом Толстого и лемнискатой Бернулли. Она выдавала ошибку во всех пяти случаях, отображенных на рис. 2.12 и 2.13. Если даже взять более простую систему уравнений с двумя, а не с четырьмя действительными корнями (см. рис. 2.15), то в точке x=1 и y=1 встроенная функция `roots` «спотыкается», а пользовательская функция *Newton* выдает одно из двух решений – см. рис. 2.15, ее нижнюю часть.

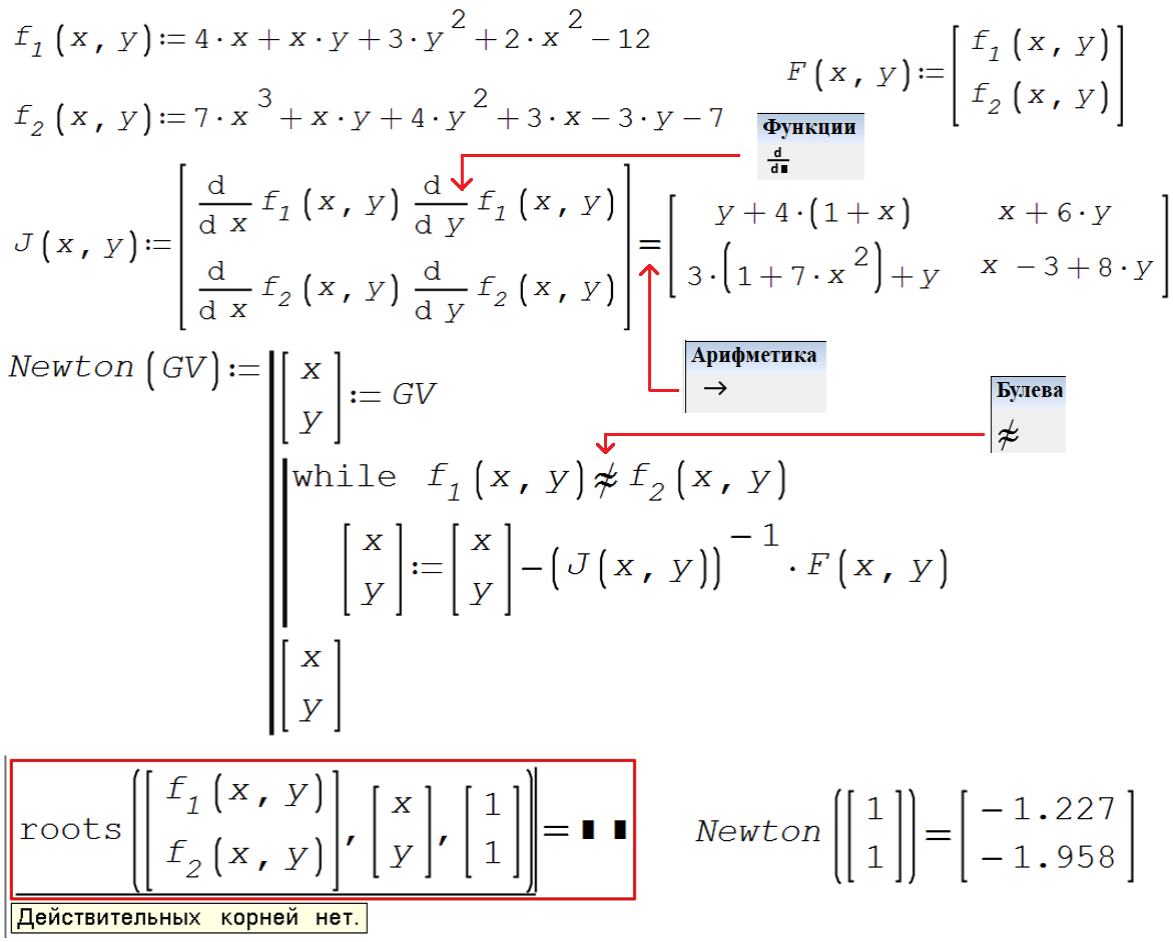


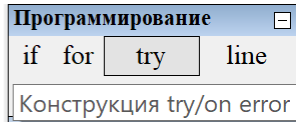
Рис. 2.15. Сравнение встроенной и пользовательской функций поиска корня системы уравнений

В решении на рис. 2.15 мы не использовали встроенную функцию `Jacobian`, а просто ввели матрицу с частными производными, которые вычислены аналитически и показаны. Наша новая система уравнений включает в себя эллипс (уравнение второго порядка) и уравнение третьего порядка – кубику.

Можно просканировать прямоугольную область первых предположений и раскрасить ее в три цвета, отмечающие зоны притяжений двух корней и зону, откуда корни не находятся. Такая авторская технология оценки качества инструментов поиска корней подробно описана в [3]. На рисунке 2.16 показана программа такого сканирования, а на рисунке 2.17 – результаты ее работы, из которой даже «невооруженным взглядом» видно, что качество пользовательской функции *Newton* намного выше качества встроенной функции `Roots`. У функции `Roots` одно преимущества – она работает быстрее: встроенная функция сканировала нашу область примерно час и двадцать минут, а пользовательская два с половиной часа. Здесь использована встроенная в SMath функция `time`, возвращающее машинное время в секундах. Вызывая эту функцию два раза – до и после выполнения программы, можно засечь время этого выполнения.

$x_b := -4 \quad x_e := 2 \quad y_b := -3 \quad y_e := 3 \quad n := 300$

$$M := \left| \begin{array}{l} [\, i1 := 1 \;\; i2 := 1 \,] \\ \text{for } x \in \left[x_b, x_b + \frac{x_e - x_b}{n} .. x_e \right] \\ \quad \text{for } y \in \left[y_b, y_b + \frac{y_e - y_b}{n} .. y_e \right] \\ \quad\quad \text{try} \\ \quad\quad\quad \left[\begin{matrix} x_1 \\ y_1 \end{matrix} \right] := Newton\left(\left[\begin{matrix} x \\ y \end{matrix} \right] \right) \\ \quad\quad\quad \text{if } \left(x_1 \approx -1.2267 \right) \wedge \left(y_1 \approx -1.9575 \right) \\ \quad\quad\quad\quad [\, X1_{i1} := x \quad Y1_{i1} := y \quad i1 := i1 + 1 \,] \\ \quad\quad\quad \text{if } \left(x_1 \approx -0.7522 \right) \wedge \left(y_1 \approx 2.2798 \right) \\ \quad\quad\quad\quad [\, X2_{i2} := x \quad Y2_{i2} := y \quad i2 := i2 + 1 \,] \\ \quad\quad \text{on error} \\ \quad\quad\quad No \\ \left[\begin{matrix} X1 & Y1 \\ X2 & Y2 \end{matrix} \right] \end{array} \right.$$



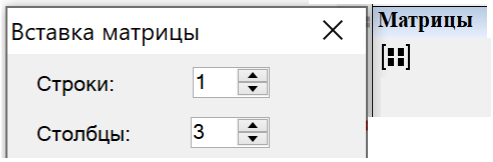


Рис. 2.16. Программа сканирования области первых предположений

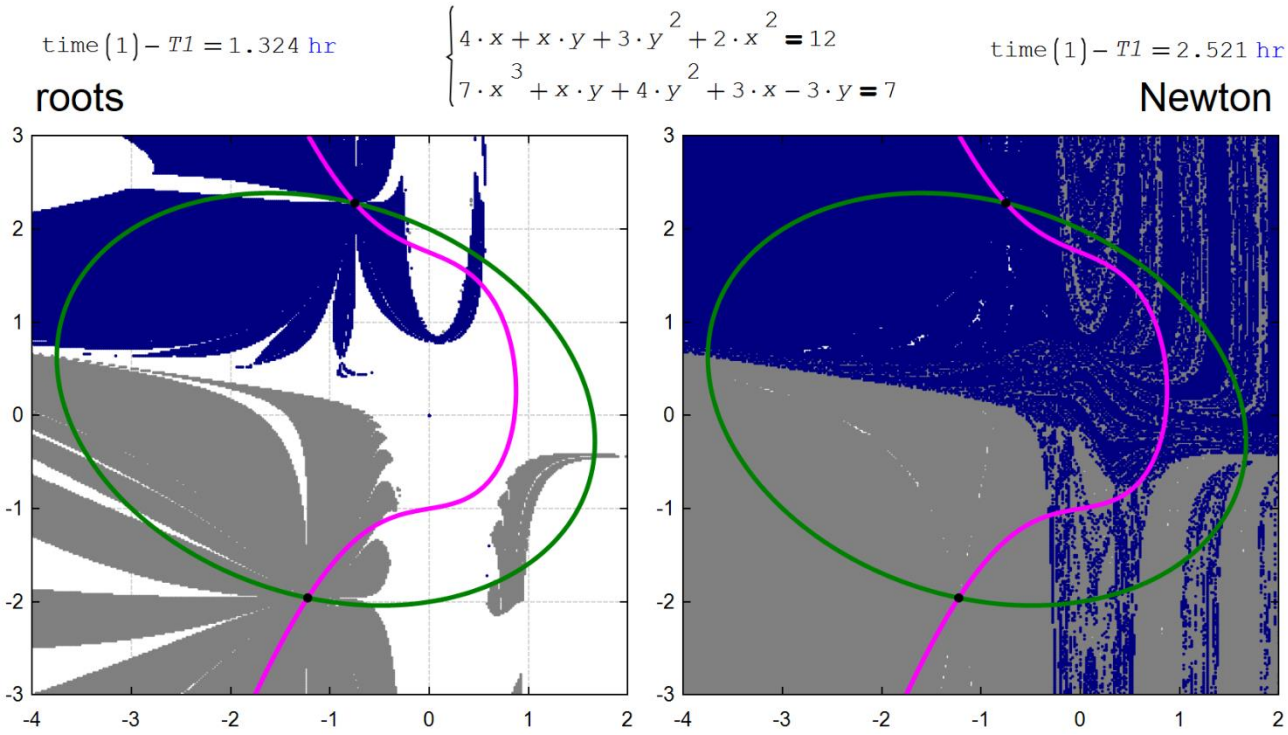


Рис. 2.17. Области притяжения двух корней системы уравнений

В программе на рис. 2.16 есть счетчики числа точек, из которых найден первый или второй корень. Это целочисленные переменные *i1* и *i2*. Они могут выполнять роль количественной оценки качества метода поиска корней системы уравнений. Через них можно подсчитать площади поверхностей с разным цветом на рис. 2.17. Чем меньше белого цвета, тем лучше сходимость метода!

В верхних углах рисунка 2.17 можно увидеть вызов функции `time` с формальным аргументом. Она возвращает машинное время. Если эту функцию вызвать в начале и конце программного блока, то можно зафиксировать время его выполнения. Пользовательская функция Newton работает в два раза медленнее, но качество ее работы выше – почти нет белых пятен.

Если в системе будет не два, а три уравнения, то квадратная матрица Якоби будет состоять из трех строк и трех столбцов. При этом графически отображать решение нужно будет не двумя кривыми, а тремя поверхностями на 3D-графике, взаимное пересечение которых и будет отмечать решение (задание читателю).

Последовательная итерационная конструкция на рис. 2.11 повторяет запись на рис. 2.4 – см. оператор `if`, где от предыдущего приближения вычитается дробь (дробь Ньютона!), числитель которой – это анализируемая функция, а знаменатель – ее производная. При работе с двумя и более уравнениями деление заменяется на матричное умножение обратной матрицы Якоби (матрицы, возведенной в минус первую степень) на функцию *F* – на вектор, хранящий анализируемые функции. Если число умножить на его обратное значений, то получится единица. Если матрицу умножить на ее обратную матрицу, то получиться единичная матрица, у которой главная диагональ хранит единицы, а остальное нули.

На рисунке 2.18 показаны зоны притяжений (зоны влияния) решений системы двух уравнений, первое из которых – это "уравнение сердца" (кривая шестого порядка), а второе – уравнение стрелы,

пронизывающей сердце (кривая нулевого порядка). Получился довольно интересный портрет в трех красках: светло-фиолетовый – это зона притяжения левого корня, темно-фиолетовый – правого корня, а черный – это зона отсутствия решения (не белый цвет, как на рис. 2.17, а черный). Желтые кружечки у висков портрета – это два решения [6].

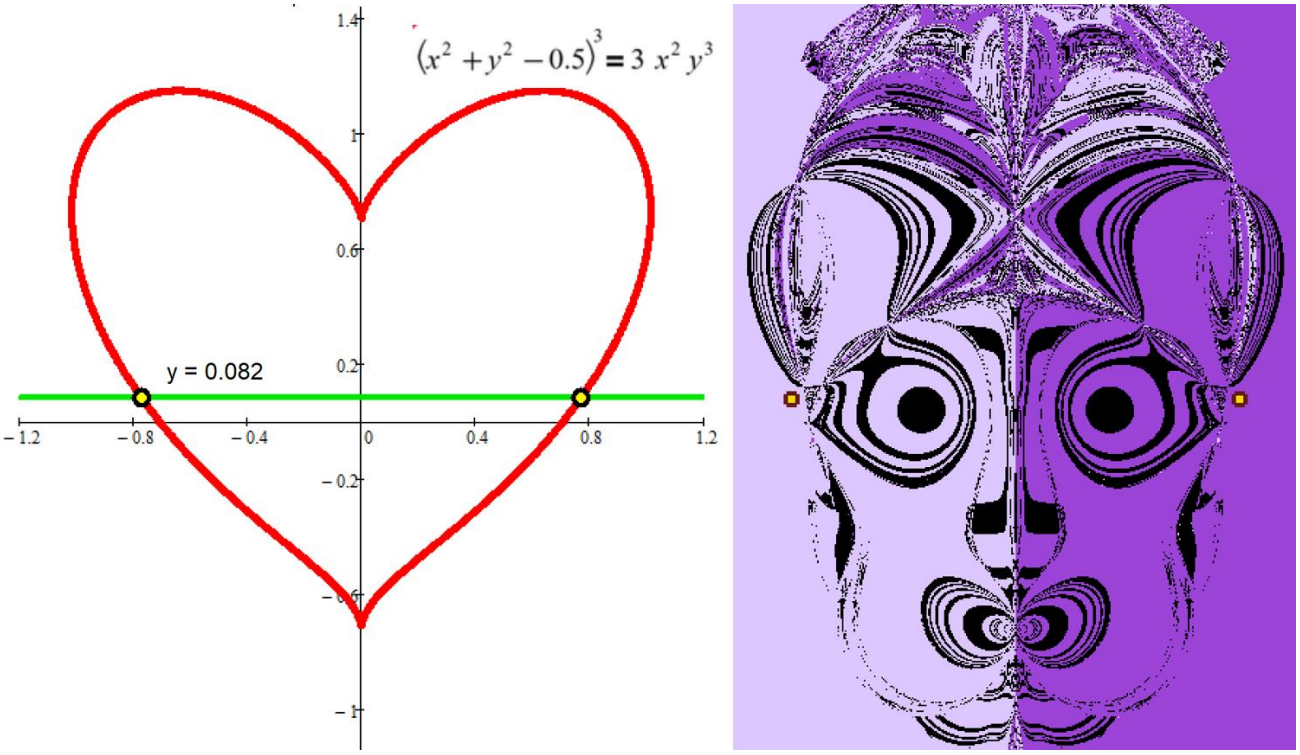


Рис. 2.18. Мистический портрет решения системы двух уравнений

Но и без раскраски на рисунке 2.12 тоже можно увидеть некое изображение – портрет крылатого коня Пегаса (см. левый верхний угол на рис. 2.12). Или Лошарика из знаменитого мультфильма, если размер зеленых точек увеличить.

Выводы

Оптимальный способ решения задачи – это сочетание символа, числа и графики. Что мы, кстати, и делали. Так на рисунках 2.4 и 2.15 показано, что производные вычислялись аналитически (символьно), а остальное делалось численно. Графики также помогли нам решить задачи – локализировать нули и корни. Такое *гибридное* решение задачи – это путь к успеху!

В настоящее время разработано множество сложнейших алгоритмов решения уравнений и их систем. Сейчас уже не понять, для чего это делалось – для повышения точности расчета и/или для уменьшения площади белых или черных зон, показанных на рис. 2.17 и 2.18, или просто – для увеличения скорости работы на старых тихоходных ЭВМ. Современные быстродействующие компьютеры вызвали возрождение (ренессанс) старых добрых простых методов решения задач. Метода Ньютона, например, героя нашего рассказа. В сложных алгоритмах непосвященному разобраться практически невозможно, а простые алгоритмы видны как на ладони. Что очень важно для образовательных целей.

Сложные современные методы можно уподобить современным реактивным лайнерам. Но так хочется смастерить самому простую конструкцию типа дельтаплана и полетать на ней… Что мы и попытались сделать в этой главе учебника.

Задание читателю:

1. Напишите программу поиска нуля функции одного аргумента методом половинного деления
2. Напишите программу поиска корней системы уравнений методом секущих
3. Решите систему трех уравнений с тремя неизвестными методом Ньютона и отобразите на трехмерном графике траекторию поиска корней.
4. Подберите такие системы двух алгебраических уравнений, чтобы портрет их корней был занимателен. Задействуете анимацию. Примеры здесь [5].

Литература:

1. Очков В.Ф., Богомолова Е.П. Это страшное слово дифуры... // Информатика в школе. №1. 2015. С. 55-58 (http://www.twt.mpei.ac.ru/ochkov/ODE.pdf)
2. Очков В.Ф. и др. Информационные технологии инженерных расчетах: SMath & Python. Издательство Лань. 2023 (http://twt.mpei.ac.ru/ochkov/EC-SMath.pdf)
3. Очков В.Ф., Чудова Ю.В., Умирова Н.Р. Портрет корней системы уравнений // Математическое образование № 3 (103), 2022. С. 33-46 (http://www.twt.mpei.ac.ru/ochkov/Portrait-Roots.pdf)
4. Очков В.Ф., Бобряков А.В., Хорьков С.Н. Гибридное решение задач на компьютере // Cloud of Science. Том 4 № 2. 2017. С. 5-26 (http://twt.mpei.ac.ru/ochkov/Hybrid.pdf)
5. https://community.ptc.com/t5/Mathcad/Portrait-of-roots-of-two-equations/m-p/776602
6. Очков В.Ф., Чудова Ю.В., Умирова Н.Р. Портрет корней системы уравнений // Математическое образование № 3 (103), 2022. С. 33-46 (http://twt.mpei.ac.ru/ochkov/Portrait-Roots.pdf)