

## Занятие 2. Вычислительный ренессанс: Метод Ньютона

*Я <...> решал на черной доске какое-то длинное алгебраическое уравнение. В одной руке я держал изорванную мяжку «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 – средней и высшей, – это её оторванность от жизни. Мол, в школе и вузе учат одному, а в жизни приходится сталкиваться совсем с другим.

Если говорить о математике, то этот упрек конкретизируется так. В вузе учат *аналитическим* (*точным*) методам решения надуманных уравнений и их систем, а в жизни к реальным, а не надуманным уравнениям, отображающим реальные физические законы и процессы (см., например, рис. 1.2 и 1.5 в главе 1 или рис. 3.8 в главе 13 с подобными уравнениями), приходится применять *численные* (*приближенными*) методы решений, о которых в аудиториях, как правило, 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 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!

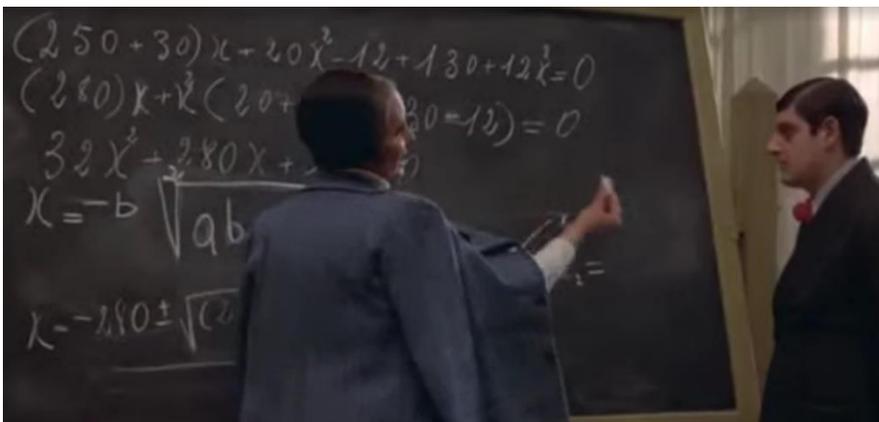


Рис. 2.1. Кадр из фильма "Амаркорд": решаем квадратное уравнение

Раньше численные методы решения алгебраических уравнений не рассматривались потому, что они были очень трудоемкими даже при работе с электронным калькулятором. Но в настоящее время в учебных заведениях 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, преподаватель в аудитории может писать на электронной доске не мелом, а специальным пером, вводя в расчёт живые формулы – формулы, по которым можно считать, строить графики, делать анимацию...

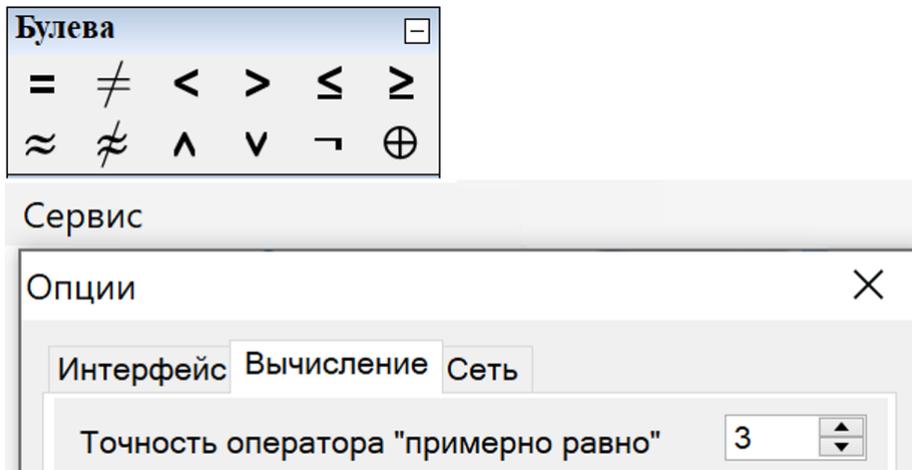
Мы пока затрагиваем только алгебраические уравнения. Но то, о чем будет сказано ниже, касается и дифференциальных уравнений – см. главу 10 и [1].

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.

### 2.1. Простой Ньютон

Давайте за пару минут установим на компьютер отечественную свободно распространяемую математическую программу SMath ([www.smash.com](http://www.smash.com)) и с её помощью покажем, как можно численно найти нули 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 пересечения графиков функций (см. рис. 2.12, 2.13 и 2.17).

В расчётах 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  $\pi$  равно примерно 3.142 (ответ 1 – булевое «да»), а не 3.14 (0 – булевое «нет»), если оперировать точностью, равной одной тысячной.



$$\pi \approx 3.14 = 0 \quad \pi \approx 3.142 = 1$$

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<sup>1</sup> 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 дифференциального исчисления<sup>2</sup> с его производной, касательной, интегралом (см. рис. 1.10 в главе 1) и прочим, упомянутом в эпиграфе этой главы учебника. 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 расчёте 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 ещё видны две отдельные точки и два отдельных значения их абсцисс – "красное" и "черное"<sup>3</sup>. 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.

<sup>1</sup> Здесь специально взят кубический, а не квадратный полином с неким горбом, через который будет несколько раз "перекатываться" наша касательная, пока она не переберётся к решению.

<sup>2</sup> [https://ru.wikipedia.org/wiki/Спор\\_Ньютона\\_и\\_Лейбница\\_о\\_приоритете](https://ru.wikipedia.org/wiki/Спор_Ньютона_и_Лейбница_о_приоритете)

<sup>3</sup> 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).

## Занятие 2

$$y(x) := x^3 - 5 \cdot x^2 - 4.5$$

$$x_1 := -2$$




Нажимаем на галочки и меняем кадры анимации

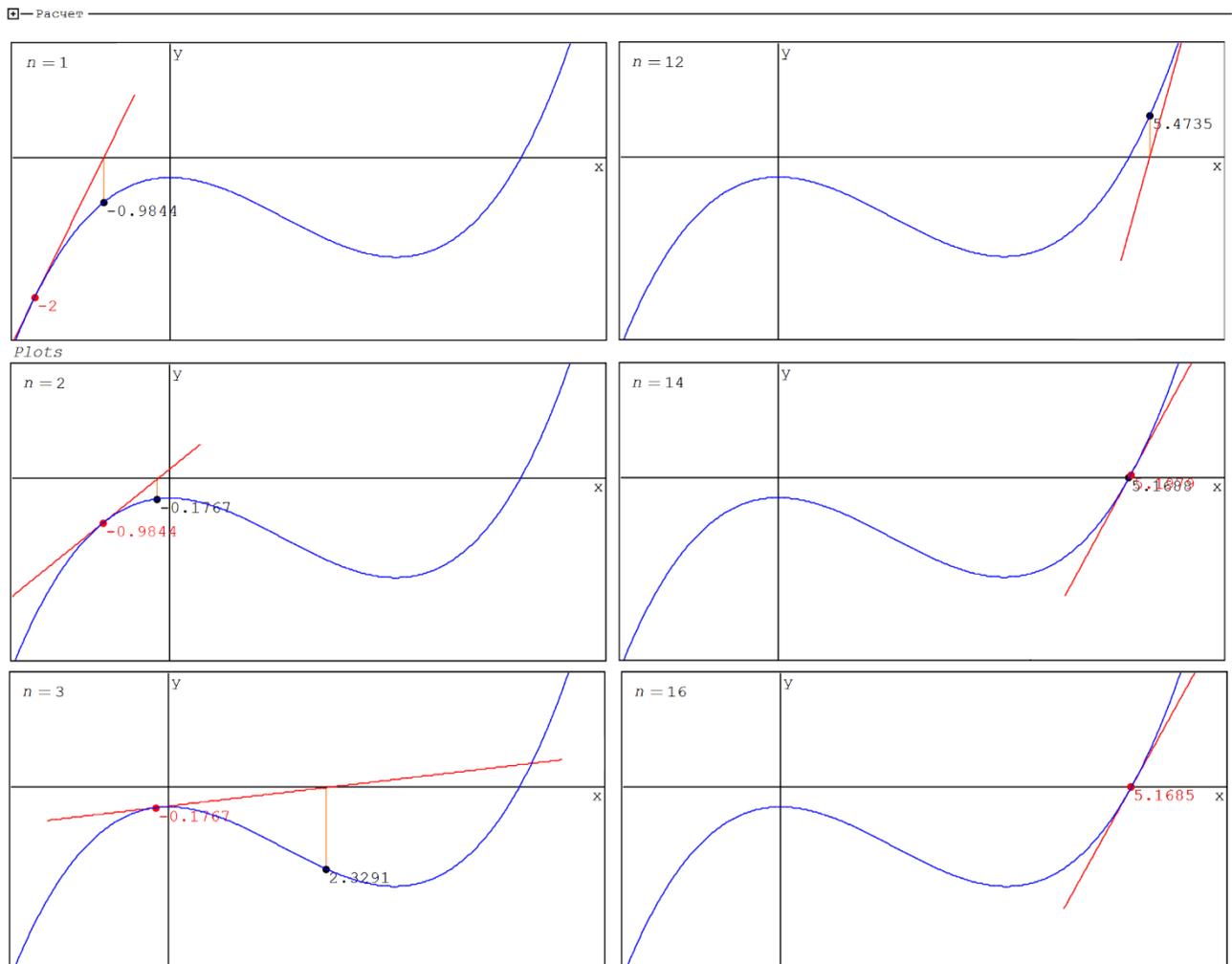


Рис. 2.3. Графическое отображение приближения к нулю функции

Функция  $y(x)$  и её первое приближение к нулю  $x_1$  задаются оператором  $:=$  (присвоить). А номер итерации  $n$  задается по-другому – через элемент управления 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<sup>4</sup>. Это позволит видеть наши касательные, иллюстрирующие метод Ньютона, почти

<sup>4</sup> Такая настройка возможна после нажатия правой кнопки мыши. Элемент управления Numeric up-down становится доступным после подгрузки соответствующего плагина.

## Занятие 2

как в анимации. Но можно создать и настоящую анимацию, сохранить её 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* (Ноль) 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 Mathcad (американской версии SMath) здесь можно применить оператор `Return`, которого в SMath пока нет.

Переменная `Plots` хранит данные для построения семи графиков:

- графика самой функции;
- графика её 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.

Занятие 2

☐ — Расчет

$$y'(x) := \frac{d}{dx} y(x) = x \cdot (x + 2 \cdot (-5 + x))$$

$$\text{Касательная}(x, x_1) := y'(x_1) \cdot (x - x_1) + y(x_1)$$

```
Zero(y, x1) :=
  j := 1
  while 1
    x2 := x1 - (y(x1) / y'(x1))
    if j = n
      break
    x1 := x2
    j := j + 1
  [ x1
    x2 ]
```

Программирование  
if for try line  
while continue break

Арифметика					
∞	π	i	±	■	←
7	8	9	+	( )	
4	5	6	-	√	∛
1	2	3	x	,	→
.	0	!	/	:=	=

Функции					
log	sign	sin	cos	∑	∏
ln	arg	tg	ctg	∫	∫
exp	{:				

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \text{Zero}(y, x_1) = \begin{bmatrix} -2 \\ -0.9844 \end{bmatrix}$$

```
Plots :=
  [ y(x)
    Касательная(x, x1)
    augment(x1, y(x1), ".", 15, "red")
    augment(x2, y(x2), ".", 15, "black")
    [ x2 0
      x2 y(x2) ]
    augment(x1, y(x1), num2str(x1, "n4"), 10, "red")
    augment(x2, y(x2), num2str(x2, "n4"), 10, "black") ]
```

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

## Занятие 2

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

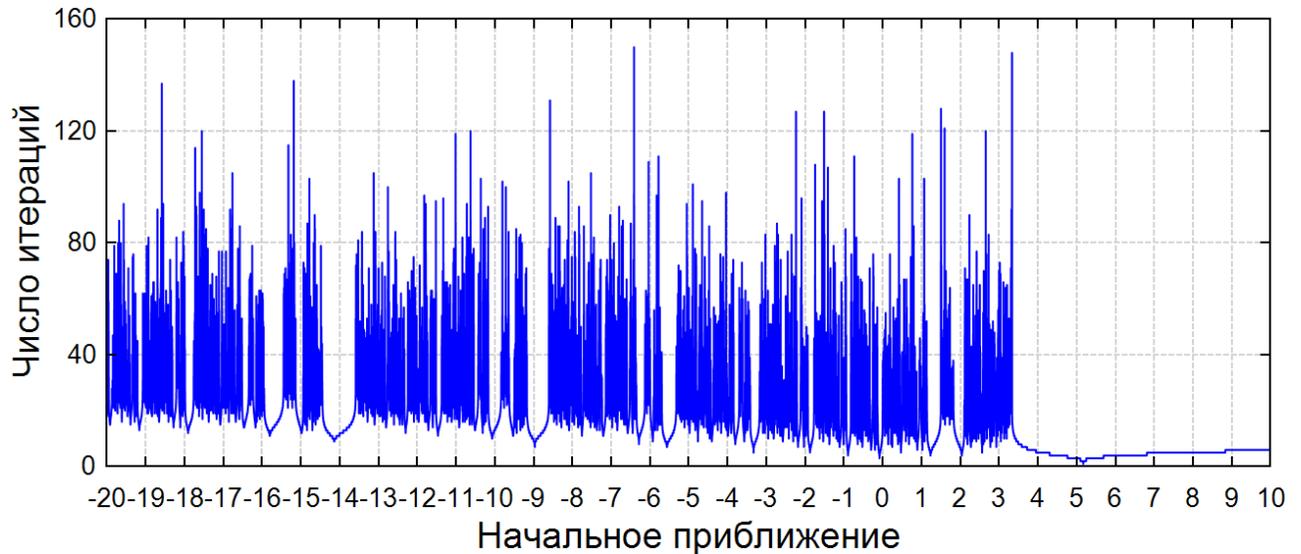


Рис. 2.5. График изменения числа итераций (метод Ньютона) в зависимости от значения начального приближения

Из графика (частотокола) 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 "частотокл".

Функция, у которой ищется нуль, может быть такая сложная, что взятие её 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 искомым нуль находится между двумя этими точками, то поиск нуля функции будет в чем-то походить на ещё один широко используемый метод численного поиска нуля функции одного аргумента – на метод *половинного деления* (см. рис. 2.6, рассматривается другая функция, чем для метода касательной).

Занятие 2

$$y(x) := x^3 - 5 \cdot x - 4.5 \quad x_1 := 2 \quad x_2 := 3$$

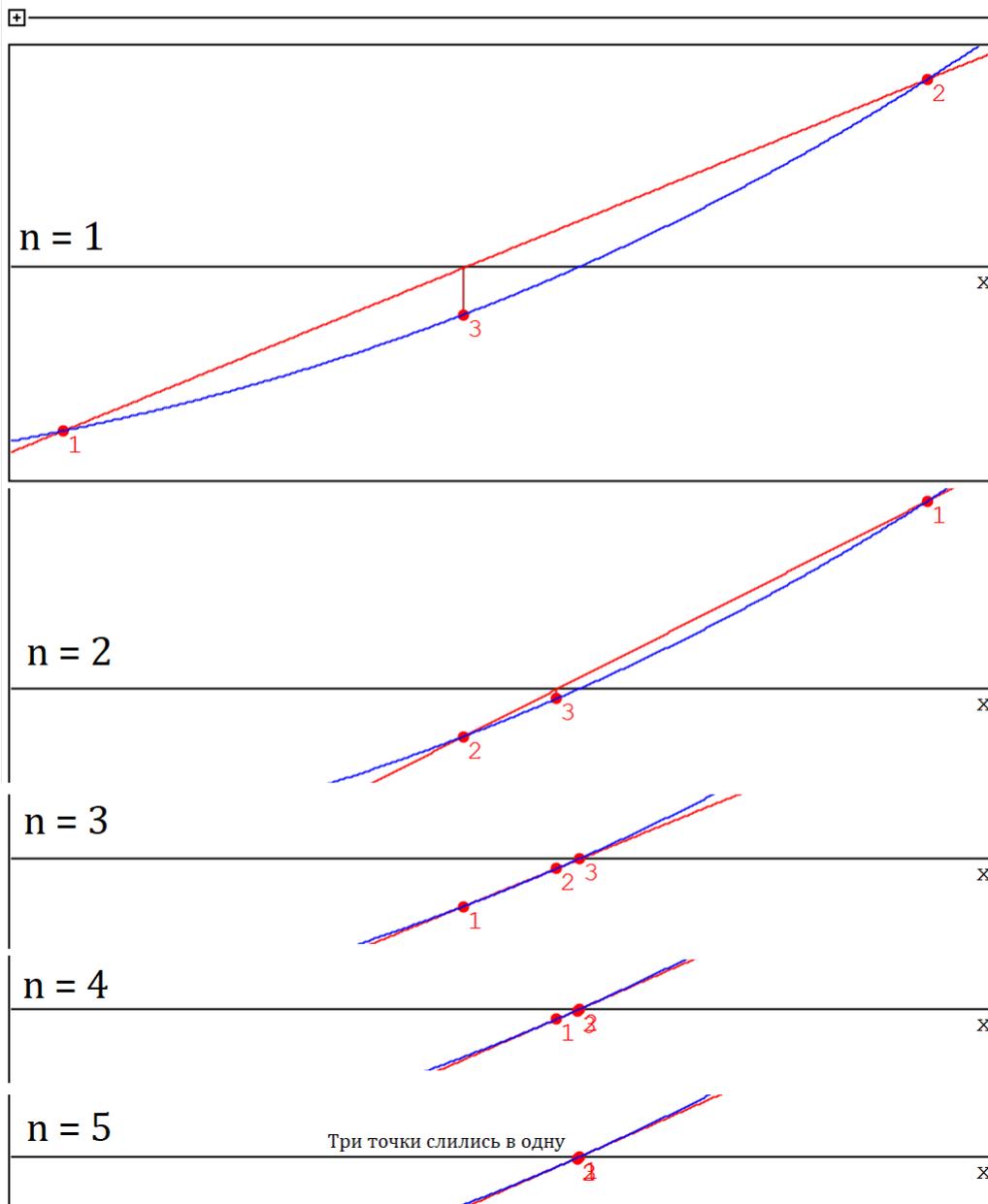


Рис. 2.6. Графическая иллюстрация метода секущих

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.

## Занятие 2

☐ — Расчёт

```

Zero (y, x1, x2) := j := 1
                  while 1
                    x3 := x2 - (x2 - x1) / (y(x2) - y(x1)) * y(x2)
                    if j = n
                      break
                    [ x1 := x2 x2 := x3 j := j + 1 ]
                  [ x1 x2 x3 ]

[ x1 x2 x3 ] := Zero (y, x1, x2)

Секущая (x, x1, x2) := y(x1) + (y(x2) - y(x1)) / (x2 - x1) * (x - x1)

Plots := {
  y(x)
  Секущая (x, x1, x2)
  [ x1 y(x1) "." 15 "red" ]
  [ x2 y(x2) "." 15 "red" ]
  [ x3 y(x3) "." 15 "red" ]
  [ x3 0 ]
  [ x3 y(x3) ]
  [ x1 y(x1) "1" 10 "red" ]
  [ x2 y(x2) "2" 10 "red" ]
  [ x3 y(x3) "3" 10 "red" ]
}

```

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 её 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  $x_3$ ;

## Занятие 2

- трех красных надписей (с размером шрифта 10) значения аргумента в трех точках приближения.

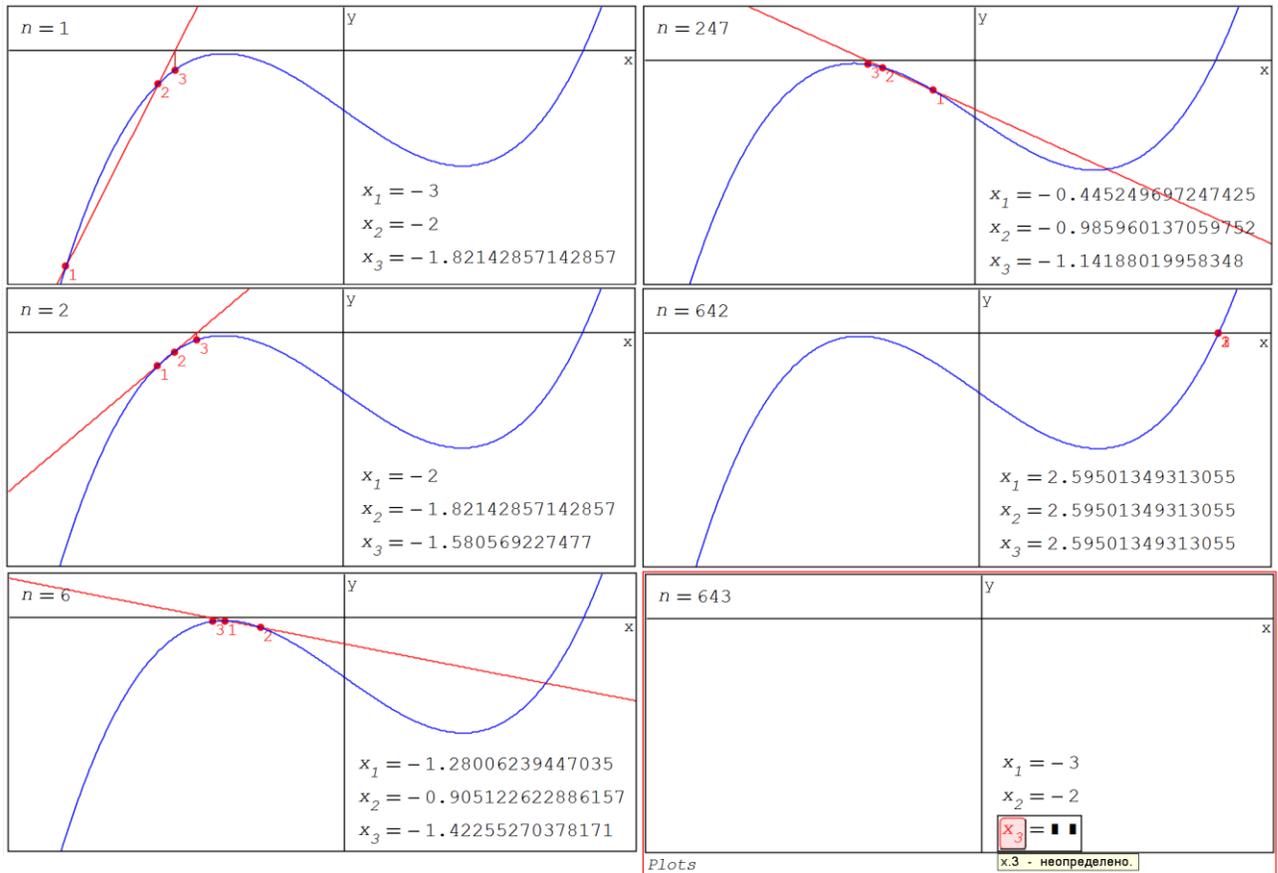


Рис. 2.8. Графическое отображение приближения к нулю функции (метод секущих)

Покраснение переменной  $x_3$  при  $n=643$  означает, что достигнут предел сходимости поиска нуля функции.

На рисунке 2.9 показан график, аналогичный графику на рис. 2.4, но не для метода Ньютона, а для метода секущих. Графики качественно отличаются.

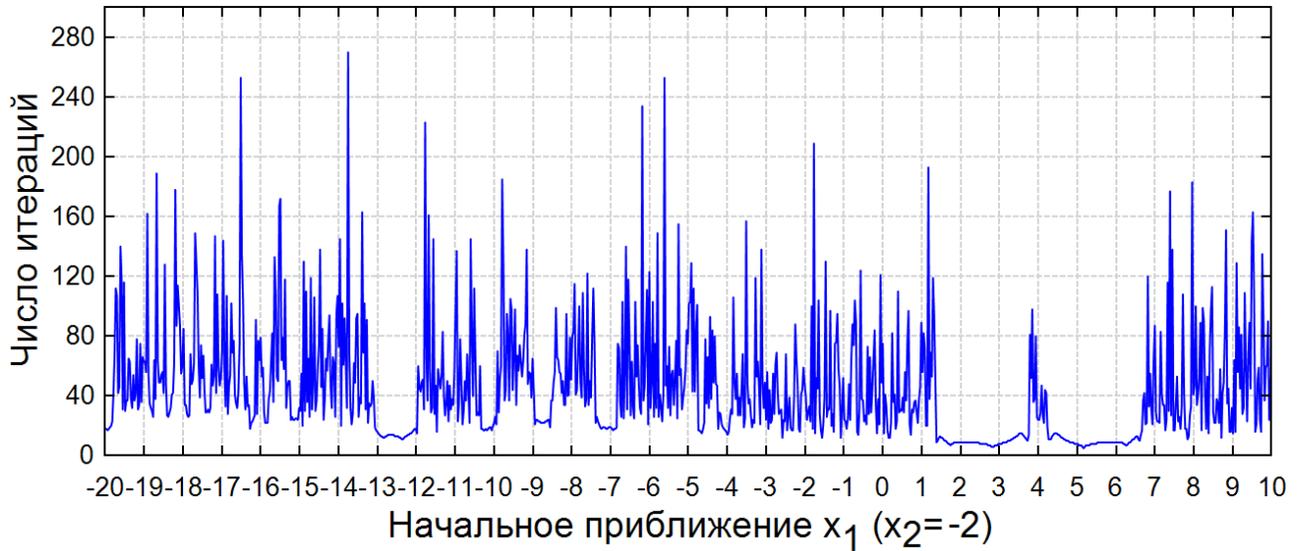


Рис. 2.9. График изменения числа итераций (метод секущих) в зависимости от значения первого начального приближения и фиксированном значении второго приближения

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.

$$\text{maple}\left(\text{solve}\left(x^3 - 5 \cdot x^2 - 4.5, x\right)\right) =$$

$$\left[ \frac{100 + \left(\sqrt[3]{2}\right)^2 \cdot \left(\sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right)^2 + 10 \cdot \sqrt[3]{2} \cdot \sqrt[3]{743 + 9 \cdot \sqrt{3729}}}{6 \cdot \sqrt[3]{2} \cdot \sqrt[3]{743 + 9 \cdot \sqrt{3729}}}, \frac{20 \left(5 - \sqrt[3]{2} \cdot \sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right) + \left(\sqrt[3]{2}\right)^2 \cdot \left(\sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right)^2 - i \cdot \sqrt{3} \cdot \left(-100 + \left(\sqrt[3]{2}\right)^2 \cdot \left(\sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right)^2\right)}{12 \cdot \sqrt[3]{2} \cdot \sqrt[3]{743 + 9 \cdot \sqrt{3729}}}, \frac{20 \left(5 - \sqrt[3]{2} \cdot \sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right) + \left(\sqrt[3]{2}\right)^2 \cdot \left(\sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right)^2 + i \cdot \sqrt{3} \cdot \left(-100 + \left(\sqrt[3]{2}\right)^2 \cdot \left(\sqrt[3]{743 + 9 \cdot \sqrt{3729}}\right)^2\right)}{12 \cdot \sqrt[3]{2} \cdot \sqrt[3]{743 + 9 \cdot \sqrt{3729}}} \right]$$

$$\begin{bmatrix} 5.168 \\ -0.08423 + 0.9293 \cdot i \\ -0.08423 - 0.9293 \cdot i \end{bmatrix}$$

Оптимизация  Символьная

Точность ответа  Численная

Арифметика

. 0 ! / := =

Оператор численного вычисления (=)

1 2 3 x , →

Оператор символического вычисления (Ctrl+.)

Рис. 2.10. Аналитический поиск нулей кубической параболы

В расчёте на рисунке 2.10 задействовано дополнение Maple, подключение которого к среде SMath позволяет вызывать встроенные конструкции математической программы Maple. В частности, вызвана

## Занятие 2

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<sup>5</sup> 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 по форме 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.

### 2.2. Двойной Ньютон

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 сходимостъ – см. рис. 2.5. Более подробно о сходимости метода Ньютона написано здесь [https://ru.wikipedia.org/wiki/Метод\\_Ньютона](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 расчёт вводится выражение овала Кассини (функция с именем  $f$ ), а затем на его основе формируются выражения для овала Толстого<sup>6</sup> ( $f_1$  – 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 её 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`.

---

<sup>5</sup> Более правильное название символьной математики – это компьютерная математика аналитических преобразований.

<sup>6</sup> 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>)

## Занятие 2

$$f(x, y, a, c) := (x^2 + y^2)^2 - 2 \cdot c^2 \cdot (x^2 - y^2) - (a^4 - c^4) \quad \text{Овал Кассини}$$

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

$$f_2(x, y) := f(x - 0.5, y - 0.5, 2, 2) \quad \text{Лемниската Бернулли}$$

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

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

$$n := 30 \quad \text{for } j \in [1..n] \\ \begin{bmatrix} X_{j+1} \\ Y_{j+1} \end{bmatrix} := \begin{bmatrix} X_j \\ Y_j \end{bmatrix} - J(X_j, Y_j)^{-1} \cdot F(X_j, Y_j)$$

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

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.

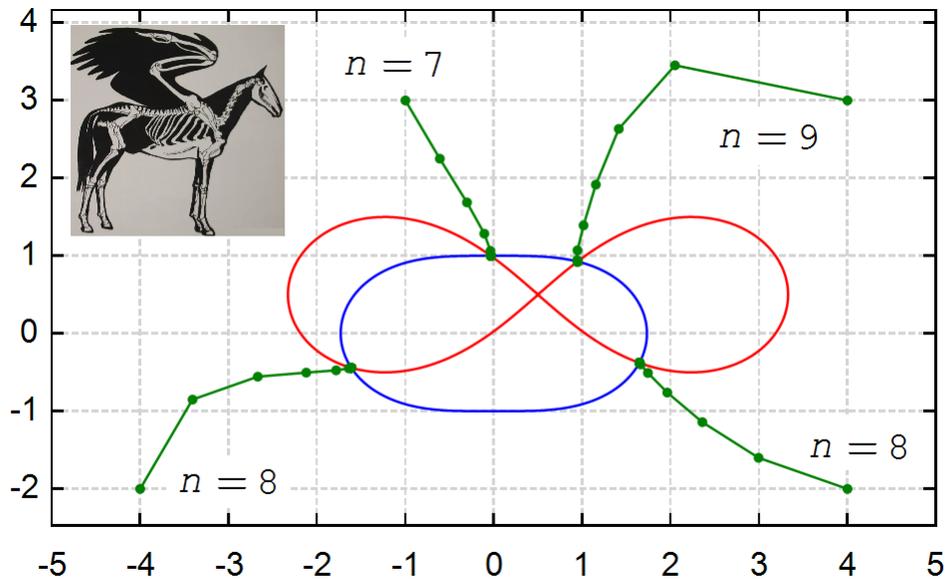


Рис. 2.12. Графическое отображение решения системы двух уравнений методом Ньютона

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 (её фото на рисунке) ведет себя примерно так, когда её приглашают к миске с кормом. Она не бежит к еде 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.

Занятие 2

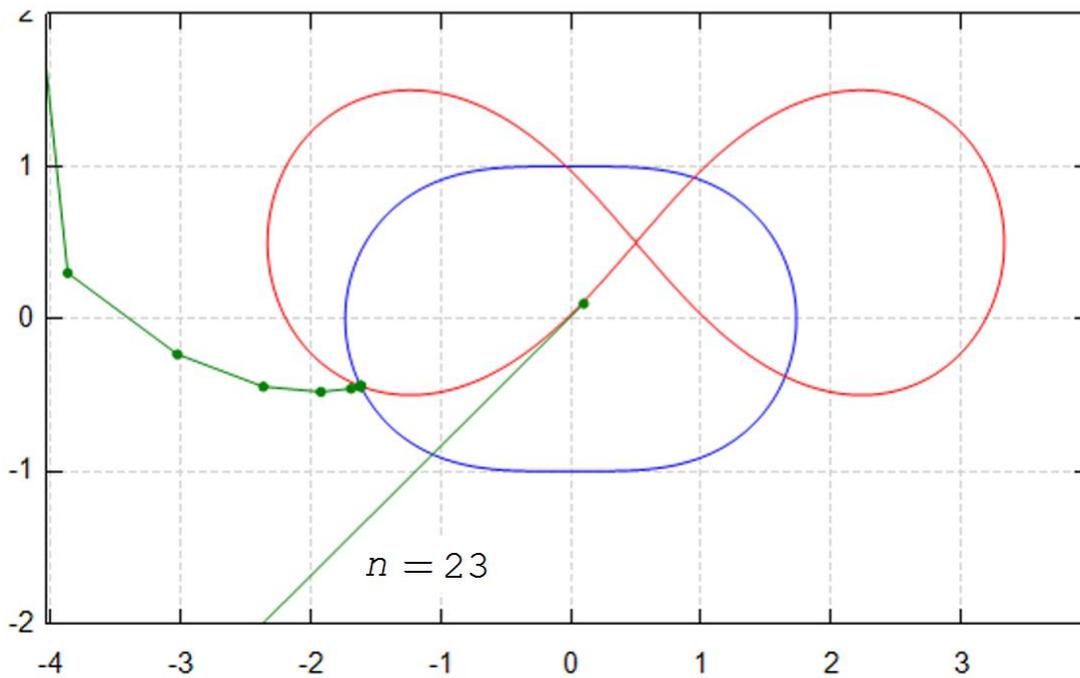
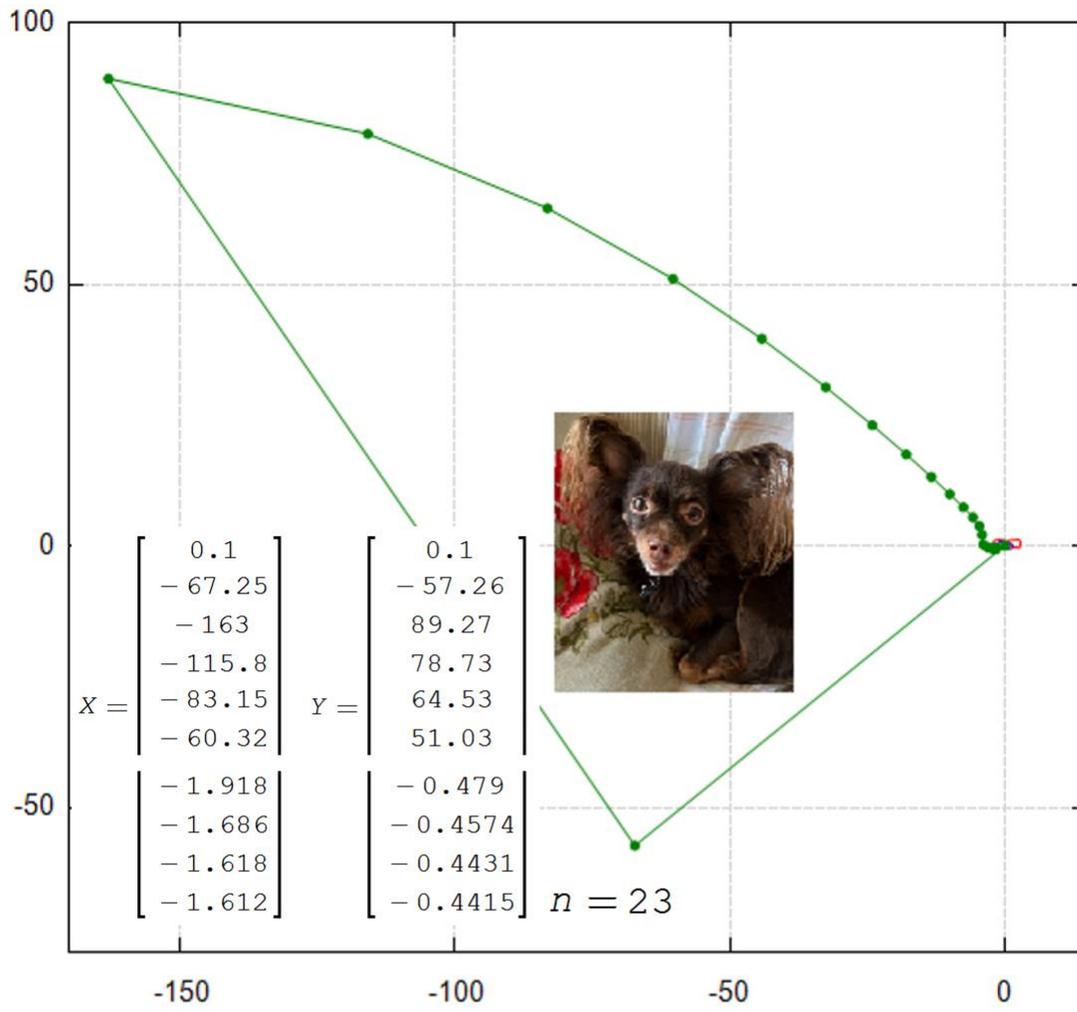


Рис. 2.13. Трассировка особого случая решения системы двух уравнений методом Ньютона: вся траектория расчёта и увеличение  $y$  начальной и конечной точки

В расчёте, показанном на рисунке 2.11, задавалось значение целочисленной переменной  $n$  – число шагов поиска корня системы. Конкретное значение этой переменной нельзя заранее предугадать так, чтобы добиться нужной точности расчёта. На рисунках 2.12 и 2.13 показаны разные значения  $n$  – от семи до 23, полученные при разных начальных предположениях – при разных значениях заданных первых элементов векторов  $X$  и  $Y$ . В расчёте, показанном на рисунке 2.14, 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 цикла  $i$  и его первое значение (единица), Второй операнд хранит булево выражение, прерывающее цикл, если выдается ответ 0 (ложь) – значения двух функций должны быть примерно равны в конце расчёта. В третьем операнде задается шаг изменения операнда, а четвертый операнд – это тело цикла. В данном примере мы могли бы спокойно обойтись с оператором `for` с тремя, а не с четырьмя операндами. Но есть задачи, где с четырьмя операндами работать проще.

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

$$f_1(x, y) := f(x, y, \sqrt{2}, 1) \quad f_2(x, y) := f(x - 0.5, y - 0.5, 2, 2)$$

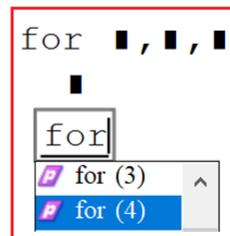
$$\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} := \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

Булева	
=	≠
≈	≉

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

`for i := 1, f1(Xi, Yi) ≈ f2(Xi, Yi), i := i + 1`

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} := \begin{bmatrix} X_i \\ Y_i \end{bmatrix} - J(X_i, Y_i)^{-1} \cdot F(X_i, Y_i)$$



$$n := \text{length}(X) = 8 \quad X_n = 0.9469 \quad Y_n = 0.9261$$

Занятие 2

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

Встроенная в SMath функция `roots` с тремя аргументами, предназначенная для численного решения систем нелинейных уравнений с опорой на первое предположение (см. последний оператор на рис. 1.3 в главе 1), была практически бессильна перед овалом Толстого и лемниской 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, её нижнюю часть. Короче! Мы берем произвольную систему нелинейных уравнений и пытаемся её решить. Встроенная функция `roots` не справляется с задачей – придумывается пользовательская функция `Newton` на основе метода Ньютона.

$$\begin{aligned}
 f_1(x, y) &:= 4 \cdot x + x \cdot y + 3 \cdot y^2 + 2 \cdot x^2 - 12 \\
 f_2(x, y) &:= 7 \cdot x^3 + x \cdot y + 4 \cdot y^2 + 3 \cdot x - 3 \cdot y - 7
 \end{aligned}$$

$$F(x, y) := \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix}$$

$$J(x, y) := \begin{bmatrix} \frac{d}{dx} f_1(x, y) & \frac{d}{dy} f_1(x, y) \\ \frac{d}{dx} f_2(x, y) & \frac{d}{dy} f_2(x, y) \end{bmatrix} = \begin{bmatrix} y + 4 \cdot (1 + x) & x + 6 \cdot y \\ 3 \cdot (1 + 7 \cdot x^2) + y & x - 3 + 8 \cdot y \end{bmatrix}$$

$$\text{Newton}(GV) := \begin{cases} \begin{bmatrix} x \\ y \end{bmatrix} := GV \\ \text{while } f_1(x, y) \neq f_2(x, y) \\ \quad \begin{bmatrix} x \\ y \end{bmatrix} := \begin{bmatrix} x \\ y \end{bmatrix} - (J(x, y))^{-1} \cdot F(x, y) \\ \begin{bmatrix} x \\ y \end{bmatrix} \end{cases}$$

$\text{roots} \left( \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix}, \begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \blacksquare \blacksquare$

$\text{Newton} \left( \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} -1.227 \\ -1.958 \end{bmatrix}$

Действительных корней нет.

### 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 – кубу.

### 2.3. Богиня численной математики

Можно просканировать прямоугольную область первых предположений и раскрасить её 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 её работы, из которой даже «невооруженным взглядом» видно, что качество пользовательской функции `Newton` намного выше качества встроенной функции `roots`. У встроенной функции `roots` одно преимущество – 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. Оператор `try` (пытаться) ведет себя так. Если произойдет ошибка счёта, то работа не прерывается сообщением об ошибке, а выполняется другой оператор или группа операторов.

Занятие 2

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

```

M := [ i1 := 1 i2 := 1 ]
for x ∈ [ x_b, x_b + (x_e - x_b) / n .. x_e ]
  for y ∈ [ y_b, y_b + (y_e - y_b) / n .. y_e ]
    try
      [ x1 ] := Newton ( [ x ] )
      [ y1 ]
      if ( x1 ≈ -1.2267 ) ∧ ( y1 ≈ -1.9575 )
        [ X1 i1 := x   Y1 i1 := y   i1 := i1 + 1 ]
      if ( x1 ≈ -0.7522 ) ∧ ( y1 ≈ 2.2798 )
        [ X2 i2 := x   Y2 i2 := y   i2 := i2 + 1 ]
    on error
      No
  [ X1 Y1 ]
  [ X2 Y2 ]

```

Программирование

if for try line

Конструкция try/on error

Вставка матрицы

Строки: 1

Столбцы: 3

Матрицы

Рис. 2.16. Программа сканирования области первых предположений

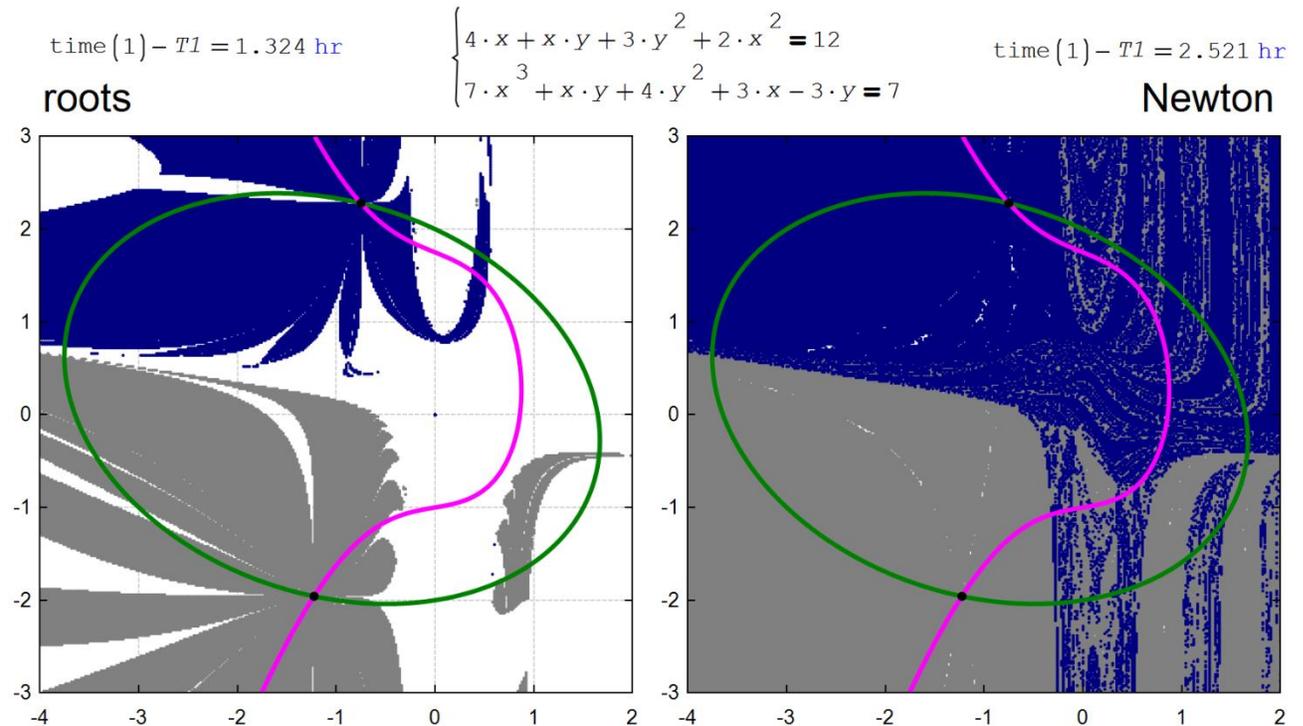


Рис. 2.17. Области притяжения двух корней системы уравнений

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. Пользовательская функция `Newton` работает в два раза медленнее, но качество её 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).

Мы забыли отметить выше, что 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 – её производная. При работе с двумя и более уравнениями деление заменяется на матричное умножение обратной матрицы

## Занятие 2

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 её обратную матрицу, то получится 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 кружочки у висков портрета – это два решения [6, 7].

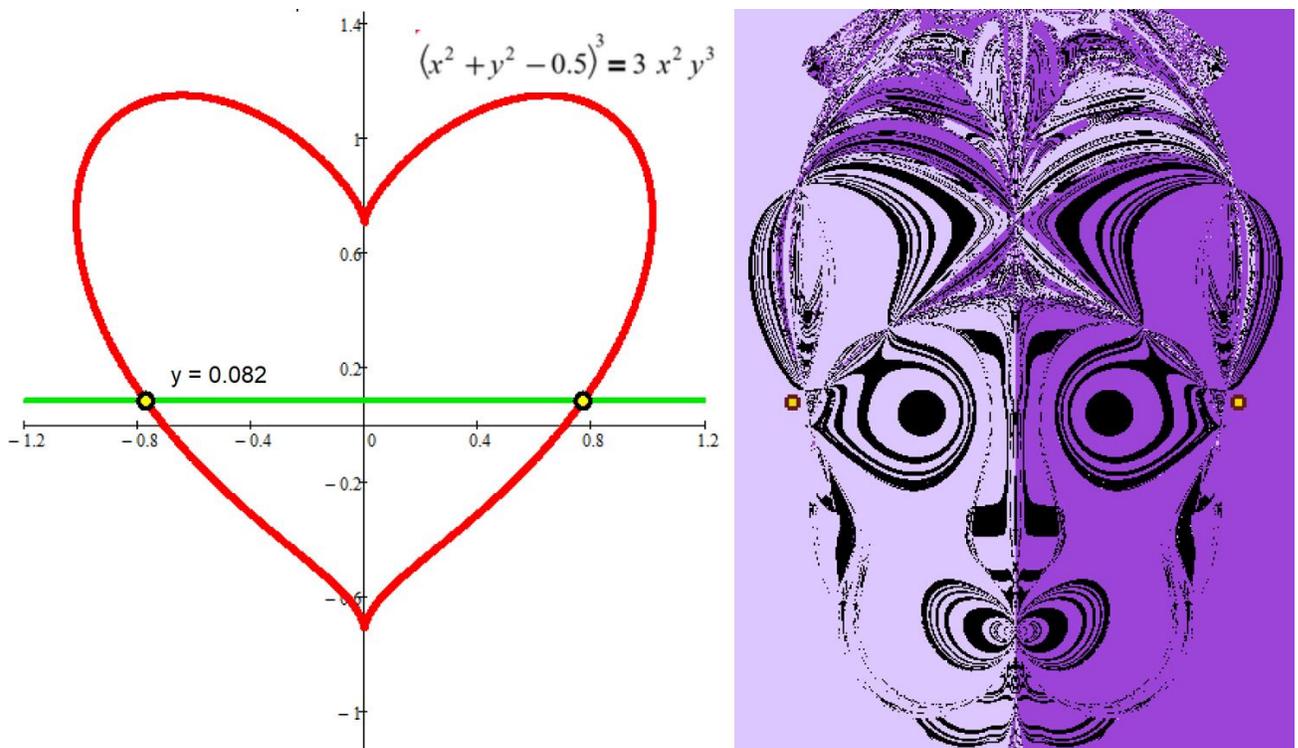


Рис. 2.18. Мистический портрет решения системы двух уравнений

Афина считается богиней математики. Покровительницей же численной математики можно считать богиню (или демона), портрет которой на рис. 2.18.

Но и без раскраски зон притяжения корней 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.

## Занятие 2

Если в системе три уравнения с тремя неизвестными, то ход численного решения такой задачи – рисование линий и точек приближения к решению (см. рис. 2.12 и 2.13) нужно будет делать не на плоскости, а в объеме. Сами же уравнения должны отображаться не линиями, а поверхностями.

Рисунок 2.19 иллюстрирует такой случай – имеется три уравнения: плоскости, сферы и конуса.

Необходимо найти точки их пересечения – решить три уравнения с тремя неизвестными  $x$ ,  $y$  и  $z$ .

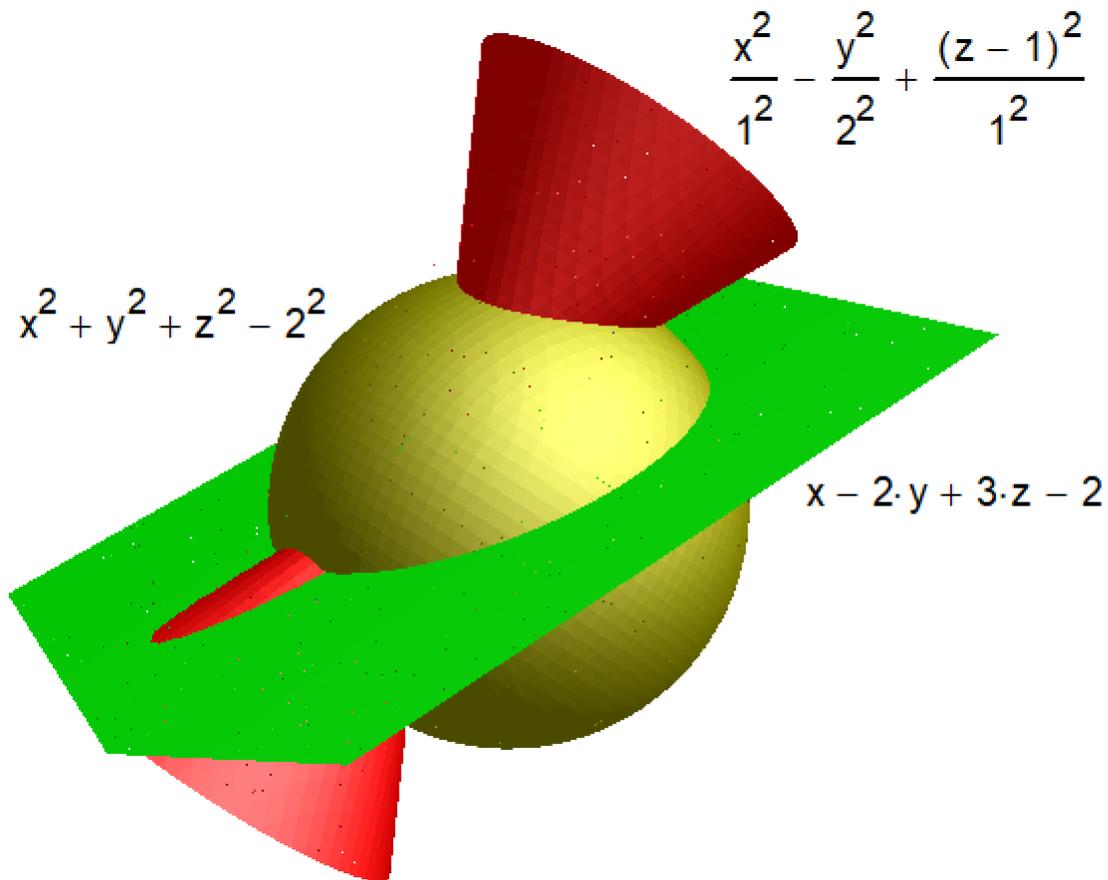


Рис. 2.19. Пересечение шара, конуса и плоскости

Формулы на рисунке 2.19 в расчете должны быть приравнены нулю, чтобы получились уравнения соответствующих поверхностей.

Рассмотрим численное решение методом Ньютона задачи с тремя уравнениями и тремя неизвестными на плоскости.

На рисунке 2.20 показана её схема: заданы координаты вершин треугольника – необходимо вписать в него окружность – найти её радиус  $r$  и координаты центра  $(x; y)$ .

## Занятие 2

Как правило, геометрические задачи решаются без использования единиц расстояния, площади, объема, но мы отойдем от этой традиции и будем работать с размерными величинами. Благо, в среде SMath они есть. Это, во-первых, повысит наглядность решения и, во-вторых, снизит вероятность ошибки при вводе формул.

В задаче требуется найти значения трех неизвестных –  $x$ ,  $y$  и  $r$ . Следовательно, необходимо составить три уравнения с тремя неизвестными. На рисунке 2.20 показано, как эти уравнения формируются в вектор-функции с именем  $F$ . Суть уравнений такова – каждая сторона треугольника (а их длины  $a$ ,  $b$  и  $c$  были подсчитаны заранее) составлена их двух катетов двух прямоугольных треугольников, каковых всего шесть. Из них составлен исходный треугольник. Длина катета согласно теореме Пифагора – это корень квадратный из квадрата гипотенузы за вычетом квадрата второго катета, длина которого равна  $r$ . У всех шести прямоугольных треугольников, из которых составлен исходный треугольник, есть одинаковый по длине катет – это радиус искомой окружности. Напомним, что при касании окружности прямой линией, радиус окружности и данная прямая образуют прямой угол.

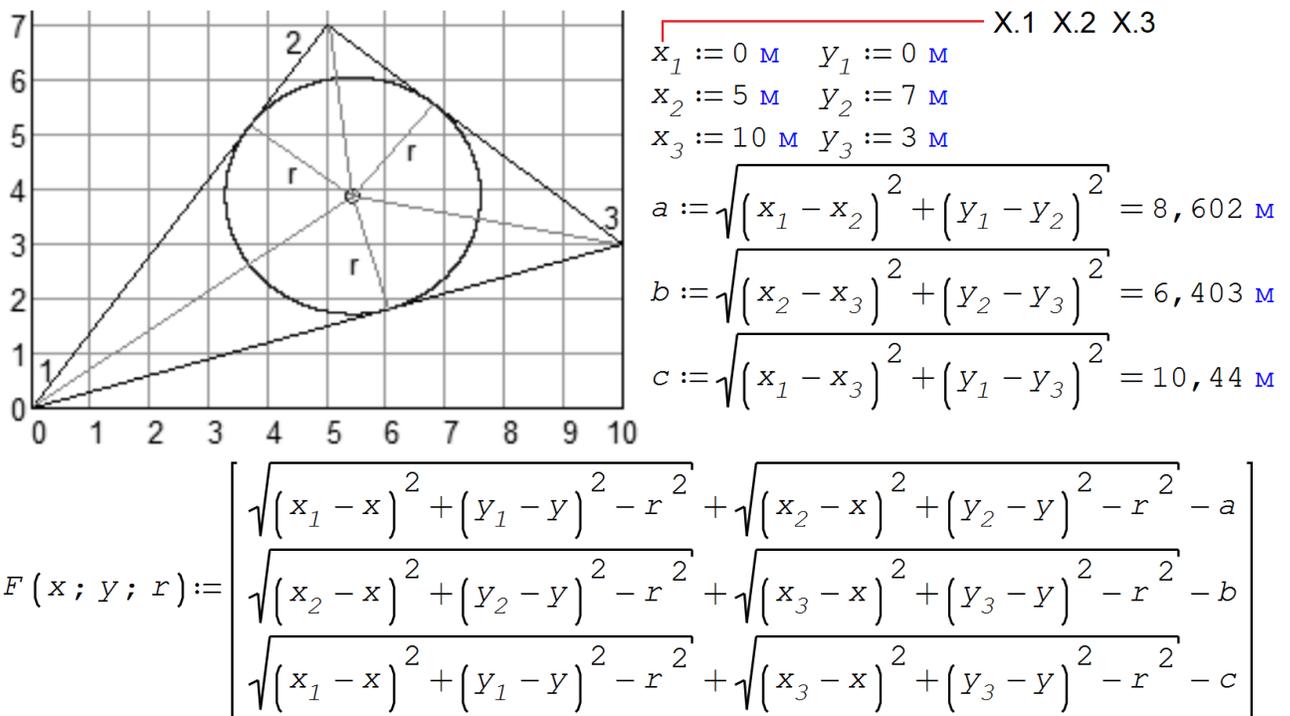


Рис. 2.20. Начало решение задачи об окружности, вписанной в треугольник

Сразу отметим, что функция `roots`, встроенная в SMath и предназначенная для решения систем нелинейных уравнений (а к этому сводится решение нашей задачи), давала сплошные сбои при решении данной трехмерной задачи. Эта функция плохо работала и с двумерной задачей – см. рис.

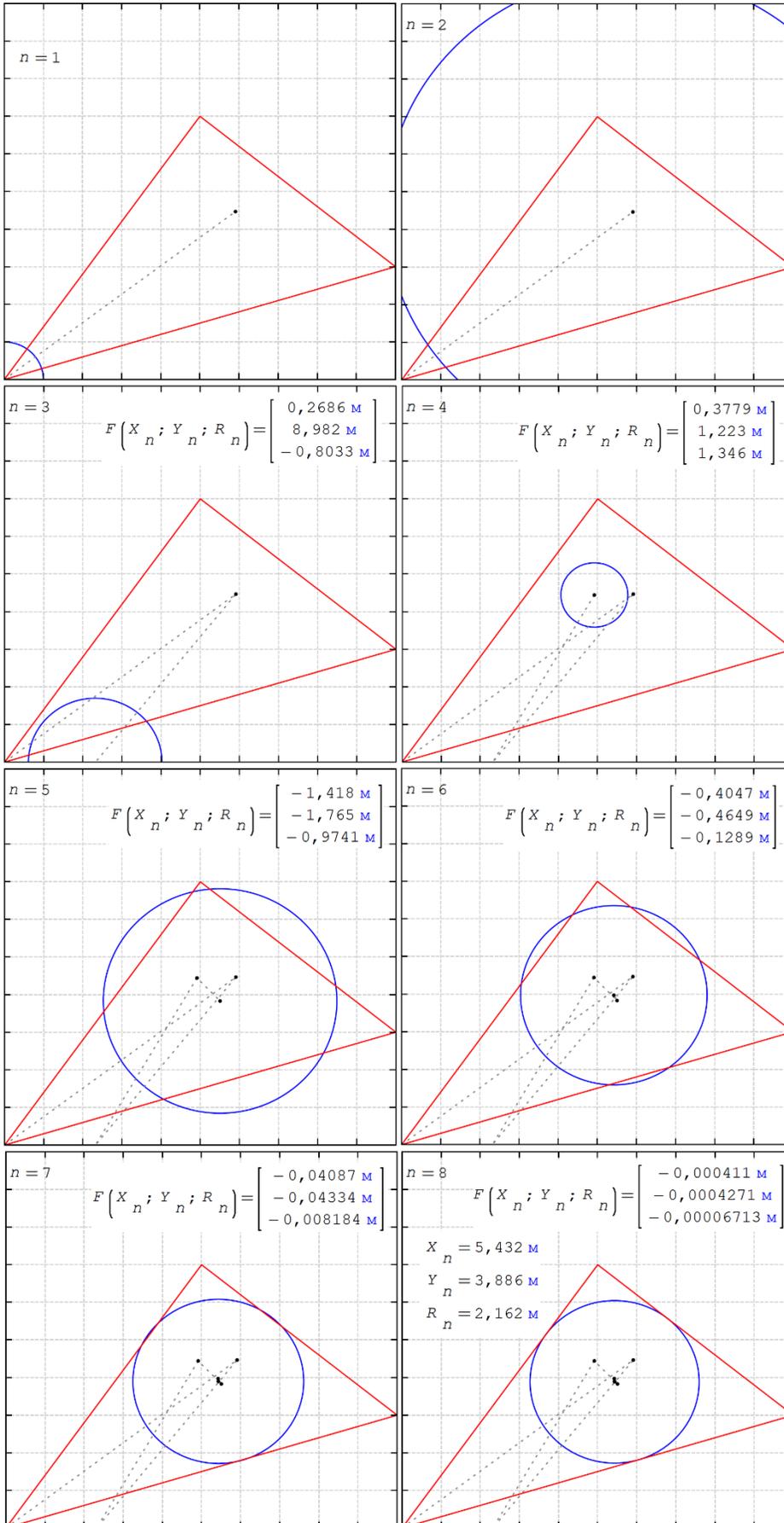
2.17. Поэтому-то и был задействован уже нами описанный метод Ньютона, но не для двухмерного, а для трехмерного случая – см. рис. 2.21.

The image shows a software interface for solving a 3D problem using Newton's method. At the top left, a dialog box titled "Вставка матрицы" (Matrix Insertion) is open, showing "Строки:" (Rows) set to 3 and "Столбцы:" (Columns) set to 1. To its right, a matrix is defined as  $\begin{bmatrix} X \\ Y \\ R \end{bmatrix} := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} M$ , with  $n := 8$ . A "Матрицы" (Matrices) toolbar is visible on the right. Below this, the Jacobian function is defined as  $J(x; y; r) := \text{Jacob} \left( F(x; y; r); \begin{bmatrix} x \\ y \\ r \end{bmatrix} \right)$ . A tooltip for the "Jacob" function explains its usage: "Jacob('1:вектор'; '2:вектор') - Возвращает матрицу Якоби от вектора функции '1:вектор'". The iterative update is shown as  $\begin{bmatrix} X_{i+1} \\ Y_{i+1} \\ R_{i+1} \end{bmatrix} := \begin{bmatrix} X_i \\ Y_i \\ R_i \end{bmatrix} - J(X_i; Y_i; R_i)^{-1} \cdot F(X_i; Y_i; R_i)$ . Finally, the data for plotting is defined as  $\text{Plot} := \begin{bmatrix} \left( x - X_n M^{-1} \right)^2 + \left( y - Y_n M^{-1} \right)^2 - \left( R_n M^{-1} \right)^2 \\ x_1 \ y_1 \\ x_2 \ y_2 \\ x_3 \ y_3 \\ x_1 \ y_1 \\ \text{augment}(X; Y) \end{bmatrix}$ .

Рис. 2.21. Решение задачи об окружности и треугольнике – метод Ньютона

В переменную *Plot* записываются данные для построения графика, иллюстрирующего процесс приближения к решению, показанного отдельными кадрами на рис. 2.22.

## Занятие 2



## Занятие 2

Рис. 2.22. Изменение радиуса и координат центра окружности при приближении к решению методом Ньютона

В первом кадре анимации ( $n = 1$  – см. рис. 2.22) показано, что наша окружность имеет радиус один метр, а её центр совпадает с первой вершиной треугольника – см. верхнюю часть рис. 2.21, где задается начальное приближение к решению. Из последнего кадра ( $n = 8$ ) видно, что задача решена – окружность вписалась в треугольник, а значения функций, входящих в функцию-вектор с именем  $F$ , примерно равны нулю – по модулю меньше, чем  $10^{-3}$  метров (один миллиметр). Из предпоследнего кадра ( $n = 7$ ) видно, что это условие не соблюдено.

Справедливости ради отметим, что нашу задачу с тремя неизвестными, несложно свести к задаче с двумя неизвестными. Для этого достаточно сделать поиск в интернете по ключу «окружность, вписанная в треугольник» и найти формулы, показанные на рис. 2.23 (в частности, формулу Герона):

$$p := \frac{a + b + c}{2}$$
$$S := \sqrt{p \cdot (p - a) \cdot (p - b) \cdot (p - c)} = 27,5 \text{ м}^2$$
$$r := \frac{S}{p} = 2,1615 \text{ м}$$

Рис. 2.23. Прямой расчёт радиуса окружности, вписанной в треугольник

В настоящее время разработано множество сложнейших алгоритмов решения уравнений и их систем. Сейчас уже не понять, для чего это делалось – для повышения точности расчёта 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.

## Занятие 2

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 книги. А в главе 10 мы "смастерили" ещё один "дельтаплан" – написали программу численного решения обыкновенного дифференциального уравнения простейшим методом Эйлера (рис. 10.3).

### Вывод

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 – локализовать нули и корни. Такое *гибридное* решение задачи – это путь к успеху! К символу, числу и графику желательно добавлять и свой интеллект – см. рис. 13.9 и текст около него в главе 13.

### Задание читателю:

1. Напишите программу поиска нуля функции одного аргумента методом половинного деления.
2. Напишите программу поиска корней системы уравнений методом секущих.
3. Подберите такие системы двух алгебраических уравнений, чтобы портрет их корней был занимателен. Задействуете анимацию. Примеры здесь [5].
4. Решите задачу об окружности, описывающей треугольник.

### 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 расчётах: 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>)

## Занятие 2

7. Ochkov, V.; Vasileva, I.; Orlov, K.; Chudova, J.; Tikhonov, A. Visualization in Mathematical Packages When Teaching with Information Technologies. *Mathematics* 2022, 10, 3413 (<https://www.mdpi.com/2227-7390/10/19/3413>)