

Министерство образования и науки Российской Федерации
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Г.В. ТРОШИНА

РЕШЕНИЕ ЗАДАЧ ВЫЧИСЛИТЕЛЬНОЙ
МАТЕМАТИКИ С ИСПОЛЬЗОВАНИЕМ
ЯЗЫКА ПРОГРАММИРОВАНИЯ
ПАКЕТА MathCad

Утверждено Редакционно-издательским советом университета
в качестве учебного пособия

НОВОСИБИРСК
2009

УДК 004.42(075.8)
Т 766

Рецензенты:

А.А. Воевода, д-р техн. наук, профессор,
Ю.А. Котов, канд. физ.-мат. наук, доцент

Работа подготовлена на кафедре вычислительной техники

Трошина Г.В.

Т 766 Решение задач вычислительной математики с использованием языка программирования пакета MathCad : учеб. пособие / Г.В. Трошина – Новосибирск: Изд-во НГТУ, 2009. – 86 с.

ISBN 978-5-7782-1283-1

Представлены основные приемы работы с языком программирования пакета MathCad. Рассмотрены численные алгоритмы типичных задач вычислительной математики и способы их решения с помощью пакета MathCad.

Предназначено для студентов, обучающихся по направлению 230100 – Информатика и вычислительная техника и специальности 230101 – Вычислительные машины, комплексы, системы и сети.

УДК 004.42(075.8)

ISBN 978-5-7782-1283-1

© Трошина Г.В., 2009
© Новосибирский государственный
технический университет, 2009

ПРЕДИСЛОВИЕ

В учебном пособии на многочисленных примерах рассматриваются разнообразные задачи численного анализа (приближенное решение нелинейных и трансцендентных уравнений, решение систем нелинейных уравнений, решение обыкновенных дифференциальных уравнений, численное интегрирование и т. д.) и возможности их решения в пакете MathCad. Текстовый редактор, входящий в состав пакета MathCad, позволяет вводить, редактировать и форматировать как текст, так и математические выражения, а вычислительный процессор позволяет осуществлять расчеты по введенным формулам с использованием численных методов. Пакет MathCad является средой визуального программирования, включает в себя дружественный интерфейс, удобный и для опытных пользователей, и для начинающих. Интерфейс состоит из меню, диалоговых окон и других элементов, которые можно использовать для решения сложных математических задач и оформлять результаты расчетов на высоком профессиональном уровне без особых усилий.

В пособии кратко изложены теоретические основы численных методов, необходимые для освоения соответствующих функций пакета MathCad. Предполагается, что пользователь имеет базовые математические знания и умеет пользоваться пакетом MathCad. После каждой темы даны примеры расчетов. В конце каждого раздела приведены контрольные вопросы для самопроверки. Список литературы включает как учебники по численным методам, так и руководства по пакету MathCad.

1. ЯЗЫК ПРОГРАММИРОВАНИЯ ПАКЕТА MathCad

1.1. Панели инструментов пакета MathCad

После запуска системы MathCad из Windows на экране отображается диалоговое окно (рис. 1.1). Интерфейс системы MathCad организован таким образом, чтобы пользователь, имеющий какие-либо навыки работы с Windows-приложениями, смог сразу начать работать.

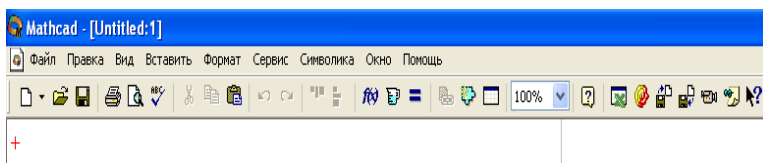


Рис. 1.1. Интерфейс пакета MathCad

Опции главного меню, содержащие основные элементы интерфейса, очень похожи на стандартные опции, принятые в текстовых редакторах Windows. Наборные панели, появляющиеся в окне редактирования документов при активизации кнопок быстрого управления, служат для вывода шаблонов математических знаков (рис. 1.2).

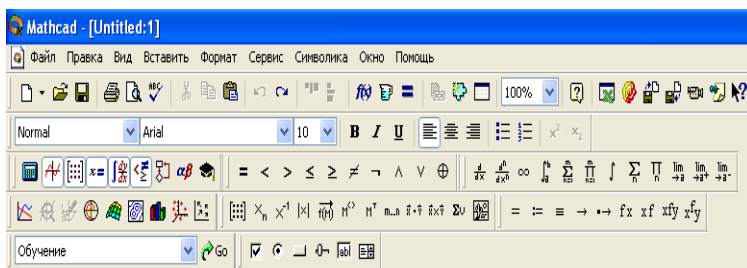


Рис. 1.2. Примеры шаблонов

Последние версии MathCad имеют мощный и элегантный собственный язык программирования, позволяющий решать самые разные задачи. Язык, который применяется для изображения констант, переменных величин, функций и других математических записей, практически полностью совпадает с общепринятым в математике. Символами этого языка являются малые и заглавные буквы латинского и греческого алфавитов, арабские цифры от 0 до 9, знаки математических операций, имена функций и некоторые специальные знаки. На математических инструментальных панелях представлен большой набор стандартных функций: тригонометрических и обратных им, гиперболических и обратных им, показательных и логарифмических, функций комплексного аргумента, специальных (Бесселя, Чебышева, Лежандра, Эрмита и других), статистических, связанных с разнообразными преобразованиями и поиском оптимальных решений. Ввод всех математических символов в программу на языке MathCad осуществляется с помощью клавиатуры или путем обращения к специальным панелям.

Возможны два типа вычислений в среде MathCad: численный и символьный. В первом случае результат получается в виде числа, во втором – в форме математического выражения. Реализация численного метода осуществляется путем обращения, например, к панелям математических инструментов (рис. 1.3).

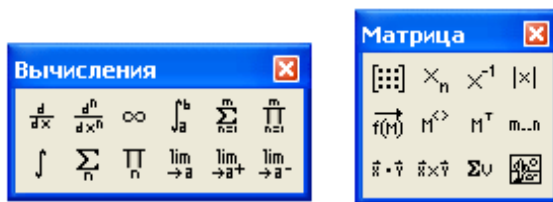


Рис. 1.3. Панели инструментов «Вычисления» и «Матрица»

Преобразование одного математического выражения в другое осуществляется, например, с помощью панели «Символы» (рис. 1.4).

Математическое выражение, подлежащее преобразованию, записывается в рабочей области среды MathCad и с помощью курсора обрамляется рамкой. Далее в зависимости от вида преобразования выбирается соответствующее ключевое слово, например «Series» – при разложении функции в степенной ряд Маклорена по выбранной переменной, «Explan» – при разложении в степенной ряд выражений типа бинома Ньютона, «Complex» – при преобразовании комплексных чи-

сел, « $M^T \rightarrow$ », « $M^{-1} \rightarrow$ », « $|M| \rightarrow$ » – при транспонировании и обращении матрицы и расчете определителя матрицы и т. д.

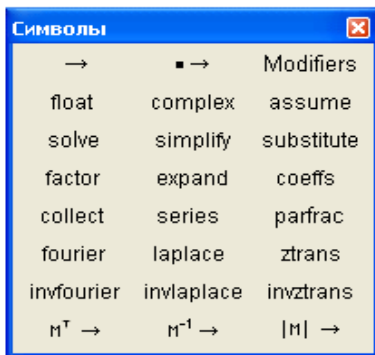



Рис. 1.4. Панель инструментов «Символы»

Встроенные функции можно вызвать с помощью кнопки , находящейся на второй строке стандартной линейки системы MathCad. В открывшемся диалоговом окне в разделе «Категория» выбирается строка, например с надписью «Trigonometric», а в разделе «Имя» – название функции (рис. 1.5).

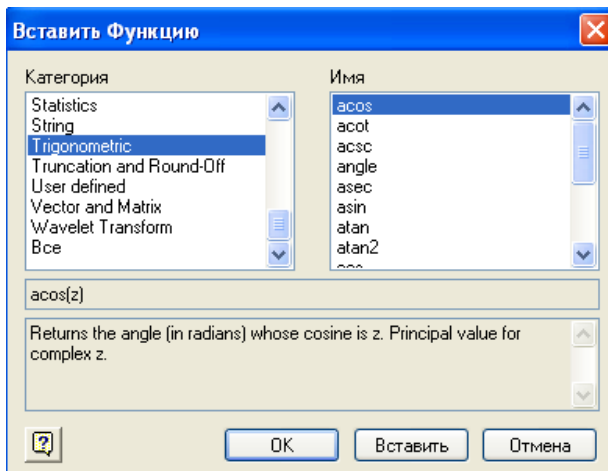



Рис.1.5. Диалоговое окно «Вставить функцию»

Для вставки программного кода в документы предназначена панель инструментов «Программирование», которую можно вызвать на экран из меню панели «Математика» при нажатии кнопки  «Панель программирования» (рис. 1.6).

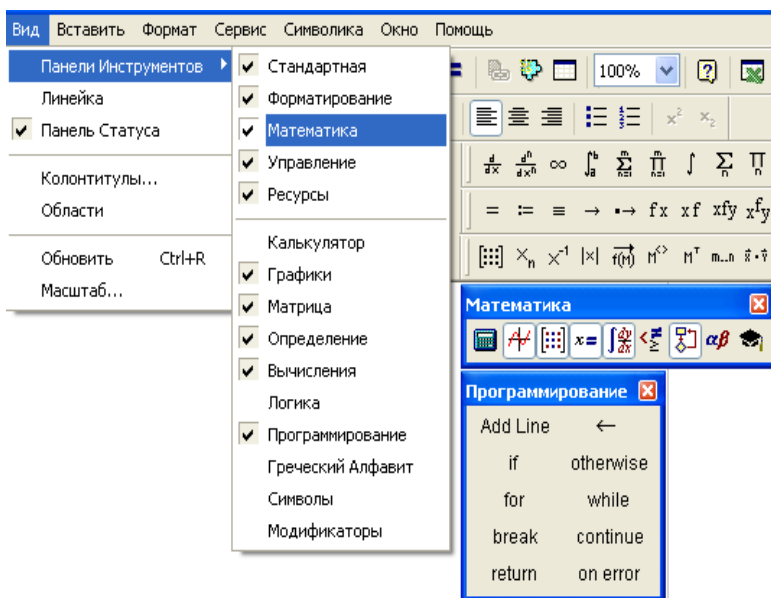


Рис. 1.6. Панель инструментов «Программирование»

1.2. Создание строки программного кода

Рассмотрим составные части языка программирования MathCad и примеры его использования. К основным инструментам относятся математические выражения, переменные и функции. Программный модуль обозначается в MathCad вертикальной чертой, справа от которой последовательно записываются операторы языка программирования. Для того чтобы создать программный модуль, необходимо выполнить следующую последовательность действий.

1. Ввести имя функции $f(x)$ или выражение, которое будет находиться слева от знака присваивания, а затем сам знак присваивания.


2. Нажать на панели инструментов «Программирование» кнопку «Add Line» («Добавить линию»).

3. Создать нужное количество линий, равное количеству строк программного кода, можно повторным нажатием кнопки «Add Line» («Добавить линию») соответствующее число раз.

4. В появившиеся заполнители ввести программный код, используя программные операторы.

Как только программный модуль будет полностью определен, функция может использоваться в численных и символьных расчетах. Вставить строку программного кода в уже созданную программу можно также с помощью кнопки «Add Line» («Добавить линию»). Для этого следует предварительно указать линии ввода. Если линия ввода находится в конце строки, то новая линия появится после строки, а если выделить лишь некоторую часть строки, то это также повлияет на положение в программе новой строки кода. В режиме выполнения программы последовательно вычисляется каждая строка кода, поэтому фрагменты кода одного уровня должны быть сгруппированы в программе с помощью вертикальных линий.

1.3. Локальное присваивание

Присваивание в пределах программы производится с помощью оператора «Local Definition», который вставляется нажатием кнопки с изображением стрелки  на панели инструментов «Программирование» (рис. 1.7).

$$f(x) := \left| \begin{array}{l} y \leftarrow 5 \\ y + x \end{array} \right|$$

$$f(2) = 7$$

Рис. 1.7. Локальное
присваивание

Переменная y существует только внутри программного модуля, за вертикальной линией. Отметим, что если в выражениях программы используется переменная, значение которой не было определено, то по умолчанию она равна нулю. Внутри программного модуля могут присутствовать как внешние, так и внутренние переменные. Значения внешних переменных в программном модуле определяются в соответ-

ствии с общими правилами операций локального и глобального присваивания. Внутренняя переменная определяется с момента присваивания ей числового значения внутри программного модуля. Если идентификаторы внутренней и внешней переменной совпадают, то в пределах программного модуля действует внутренняя переменная. Результатом вычисления программного модуля считается последняя выполняемая в модуле формула. В примере, приведенном на рис. 1.8, переменная a – внешняя, а переменная x – внутренняя.

$a := 5$

$$\left| \begin{array}{l} x \leftarrow a + 4 \rightarrow 81 \\ x^2 \end{array} \right|$$

Рис. 1.8. Внешняя
и внутренняя переменные

Отметим, что, в отличие от правил записи формул на рабочем листе документа MathCad, внутри программного модуля в одной строке можно записать только один оператор или формулу.

1.4. Условный оператор

Условный оператор **if** предназначен для выполнения вычислений в зависимости от условия и состоит из двух частей. При вызове оператора **if** появляется шаблон с двумя заполнителями: **if**. Вначале проверяется логическое условие справа от него. Если условие верно, то выполняются действия, указанные слева от оператора **if**. Если же условие ложно, то выполнение программы продолжается со следующей строки. Для того чтобы вставить условный оператор **if**, необходимо выполнить следующую последовательность действий.

1. Создать строку программного кода, нажав на панели инструментов «Программирование» кнопку «Add Line» («Добавить линию»).
2. Нажать кнопку условного оператора **if** на панели инструментов «Программирование».

3. Справа от оператора **if** определить условие. Для этого можно воспользоваться панелью инструментов «Логика» (рис. 1.9).

4. Слева от оператора **if** ввести выражение, которое будет выполняться в случае, если условие истинно.

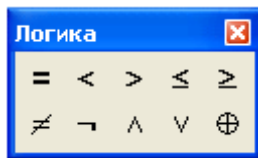


Рис. 1.9. Панель инструментов «Логика»

Оператор **otherwise** используется с одним или несколькими операторами **if** и определяет выражение, которое будет вычисляться, если ни одно из условий не является истинным (рис. 1.10).

$$\text{abs}(x) := \begin{cases} -x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

$$\text{abs}(-6) = 6 \qquad \text{abs}(6) = 6$$

Рис. 1.10. Пример использования оператора **otherwise**

1.5. Функции пользователя

Очень часто программные модули используются для определения функций пользователя. В этом случае в конце программного модуля должна быть указана формула, являющаяся результатом вычисления функции. Например, на рис. 1.8 приведено формирование функции пользователя $f(x) = x^2$. Рассмотрим порядок формирования функции пользователя (xy) с использованием операторов **if** и **otherwise**.

1. Ввести имя функции $f(x)$ или выражение, которое будет находиться слева от знака присваивания, а затем – сам знак присваивания.

2. Нажать на панели инструментов «Программирование» кнопку «Add Line» («Добавить линию»).

3. Установить курсор на верхнюю метку и нажать кнопку условного оператора **if** на панели инструментов «Программирование».

4. Далее поставить курсор на первую метку условного оператора **if** и нажать кнопку «Add Line» («Добавить линию»), для того чтобы образовать блок для оператора **if** (рис. 1.11).

5. Переместить курсор на последнюю метку и кнопкой «Otherwise» панели инструментов «Программирование» ввести оператор **otherwise**.

6. Убедиться в том, что курсор установлен на метке оператора **otherwise**, и нажать кнопку «Add Line» («Добавить линию») для образования блока оператора **otherwise** (рис. 1.12).

7. Перестить курсор таким образом, чтобы выделить блок оператора **otherwise** (см. рис. 1.12) и кнопкой «Add Line» («Добавить линию») ввести новую метку.

8. На месте всех меток разместить выражения согласно рис. 1.13.

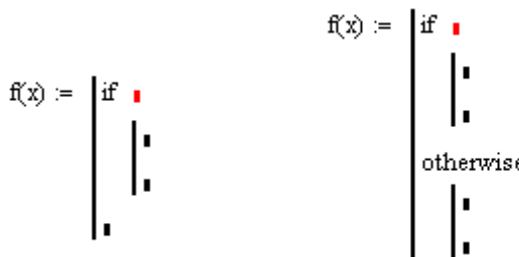


Рис. 1.11. Блок для оператора **if**

Рис. 1.12. Блок для оператора **otherwise**

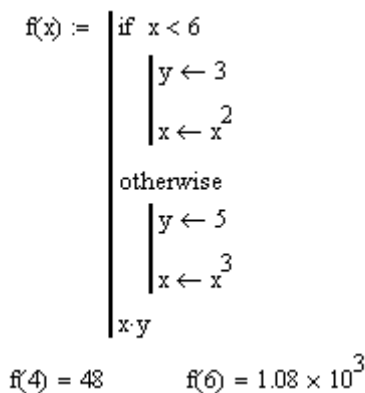


Рис. 1.13. Пример программирования в численных расчетах

1.6. Операторы цикла

В языке программирования пакета MathCad существует два оператора цикла: **while** и **for**. Оператор цикла **for** позволяет сформировать цикл по некоторой переменной, заданной на определенном диапазоне значений. Оператор цикла **while** организует цикл с выходом из него по какому-то логическому условию. Для того чтобы вставить оператор цикла, необходимо выполнить следующую последовательность действий.

1. Создать строку программного кода, нажав на панели инструментов «Программирование» кнопку «Add Line» («Добавить линию»).
2. Нажать кнопку оператора цикла на панели инструментов «Программирование».

```
y := | x ← 0
      | while x < 5
      |   x ← x + 2
```

y = 6

Рис. 1.14. Оператор цикла **while**

3. Если выбран оператор цикла **for**, то необходимо вставить в соответствующий заполнитель имя переменной и диапазон ее значений, а если выбран оператор цикла **while**, то необходимо указать логическое выражение, при котором будет осуществляться выход из цикла (рис. 1.14).

4. В нижнем заполнителе определить тело цикла.

Диапазон значений переменной в условии цикла **for** можно задать с помощью как ранжированной переменной (рис. 1.15), так и вектора (рис. 1.16).

```
y := | x ← 0
      | for k ∈ 0..10
      |   x ← x + k
```

y = 55

Рис. 1.15. Оператор цикла **for** с ранжированной переменной

```
y := | x ← 0
      | for k ∈ {1 2 3 4 5}
      |   x ← x + k
```

y = 15

Рис. 1.16. Оператор цикла **for** с вектором

Пример 1.1. Сформировать программный модуль для нахождения среднего арифметического значения функции

$$f(x) = \begin{cases} \sin x, & \text{если } 0 \leq x \leq \pi/2 \\ \cos x, & \text{если } x > \pi/2 \end{cases}$$

в точках $x_k = 0.25 \cdot k$, $k = 1, 2, \dots, 10$.

Решение. Используя оператор **if** и оператор **otherwise**, сформируем функцию $f(x)$ (рис. 1.17).

$$f(x) := \begin{cases} \sin(x) & \text{if } (x \geq 0) \cdot \left(x \leq \frac{\pi}{2}\right) \\ \cos(x) & \text{otherwise} \end{cases}$$

Рис. 1.17. Программный модуль для функции $f(x)$

Вычисление среднего арифметического осуществляется с помощью оператора цикла **for** (рис. 1.18).

$$g(k) := \begin{cases} s \leftarrow 0 \\ \text{for } i \in 1..k \\ \quad s \leftarrow s + f(0.25 \cdot i) \\ \quad \frac{s}{k} \end{cases}$$

Рис. 1.18. Программный модуль для функции $g(k)$

При $k = 10$ получаем $g(10) = 0.21727$.

Оператор **break** предназначен для досрочного завершения цикла. На рис. 1.19 показано использование оператора **break** совместно с оператором цикла **for**, а на рис. 1.20 – использование оператора **break** совместно с оператором цикла **while**.

```

y :=
  x ← 0
  for k ∈ 0..10
    x ← x + k
    break if k = 6

```

y = 21

Рис. 1.19. Использование оператора **break** внутри цикла **for**

```

y :=
  x ← 0
  while x < 5
    x ← x + 1
    break if x > 3

```

y = 4

Рис. 1.20. Использование оператора **break** внутри цикла **while**

Пример 1.2. Сформировать программный модуль для нахождения значения первого элемента и его позиции в числовом массиве, который находится в интервале $0.6 \leq f_i \leq 0.8$.

Решение. Встроенная функция `rnd(1)` генерирует случайные числа в интервале от 0 до 1. Создадим числовой массив с помощью функции `rnd(1)` (рис. 1.21).

i := 1..10

f_i := rnd(1)

f =

	0
0	0
1	0.54509
2	0.40898
3	0.46556
4	0.15266
5	0.73799
6	0.82669
7	0.87334
8	0.30008
9	0.12723
10	0.78496

Рис. 1.21. Формирование числового массива

Используя оператор цикла **for**, операторы **if** и **break**, определяем функцию для вычисления значения первого элемента и его номера в массиве случайных чисел на интервале $0.6 \leq f_i \leq 0.8$ (рис. 1.22).

$$g(f) := \left| \begin{array}{l} \text{for } i \in 0.. \text{last}(f) \\ \quad \text{break if } (f_i \geq 0.6) \cdot (f_i \leq 0.8) \\ \left(\begin{array}{l} i \\ f_i \end{array} \right) \end{array} \right.$$

Рис. 1.22. Программный модуль для функции $g(f)$

Получаем результат вычислений: $g(f) = \left(\begin{array}{l} 5 \\ 0.73799 \end{array} \right)$.

Оператор **continue** используется для того, чтобы более четко обозначить границы завершения тела цикла. На результат работы программы оператор **continue** не влияет. На рис. 1.23 показано использование оператора **continue** совместно с оператором цикла **for**, а на рис. 1.24 – использование оператора **continue** совместно с оператором цикла **while**.

$$y := \left| \begin{array}{l} x \leftarrow 0 \\ \text{for } k \in (1 \ 2 \ 3 \ 4 \ 5) \\ \quad \left| \begin{array}{l} x \leftarrow x + 1 \\ \text{continue} \end{array} \right. \end{array} \right.$$

$y = 5$

+

$$y := \left| \begin{array}{l} x \leftarrow 0 \\ \text{while } x < 5 \\ \quad \left| \begin{array}{l} x \leftarrow x + 2 \\ \text{continue} \end{array} \right. \end{array} \right.$$

$y = 6$

Рис. 1.23. Оператор цикла **for** с оператором **continue**

Рис. 1.24. Оператор цикла **while** с оператором **continue**

Пример 1.3. Сформировать программный модуль для нахождения минимального и максимального элементов в массиве случайных чисел, который создается с помощью функции $\text{rnd}(3)$.

Решение. Создадим числовой массив с помощью функции `rnd(3)` (рис. 1.25).

`i := 1..10`

`fi := rnd(3)`

$f =$

	0
0	0
1	1.96468
2	0.29494
3	0.14895
4	1.39215
5	2.18358
6	2.03977
7	1.11634
8	0.14995
9	0.50867
10	0.93171

Рис. 1.25. Формирование числового массива

Используя оператор цикла **for**, операторы **if** и **continue**, определяем функцию вычисления минимального и максимального элементов массива (рис. 1.26).

```

h(f) :=
| min ← f1
| max ← f1
| for i ∈ 1..last(f)
|   | if fi < min
|   |   | min ← fi
|   |   | continue
|   | max ← fi if fi > max
| ( min
|   max )

```

Рис. 1.26. Программный модуль для функции $h(f)$

Получаем результат вычислений: $h(f) = \begin{pmatrix} 0.14895 \\ 2.18358 \end{pmatrix}$.

Пример 1.4. Сформировать программный модуль для вычисления суммы $\sum_{k=0}^m \frac{1}{b_k}$, если $b_k \neq 0$, и найти количество нулевых элементов в массиве b_k , $k = 0, 1, \dots, m$.

Решение. Зададим числовой массив в виде: $b_k = (2 \ 0 \ 6 \ 0 \ 12)$. Используя оператор цикла **for**, операторы **if** и **otherwise**, определяем программный модуль для вычисления суммы элементов массива и нахождения количества нулевых элементов (рис. 1.27).

```

m(b) :=
| sum ← 0
| n ← 0
| for i ∈ 0..last(b)
|   | n ← n + 1 if bi = 0
|   | sum ← sum + 1/bi otherwise
| (sum)
| (n)

```

Рис. 1.27. Формирование функции $m(b)$

Получаем результат вычислений: $m(b) = \begin{pmatrix} 0.75 \\ 2 \end{pmatrix}$.

1.7. Оператор return

Оператор **return** используется для выхода из блока и передачи значения из любой точки программного блока. Значение, введенное в заполнитель, указанный после оператора **return**, возвращается в качестве результата и никакой другой код больше не используется. В програм-

му оператор **return** вставляется с помощью одноименной кнопки панели инструментов «Программирование». На рис. 1.28 приведен пример использования цикла **while** и оператора **return** для вычисления факториала. В этом примере определен бесконечный цикл **while**, а принудительный выход из него осуществляется с использованием оператора **return**.

```
f(n) := | f ← n
        | while 1
        |   | f ← f(n - 1)
        |   | n ← n - 1
        |   | return f if n = 1
```

$f(4) = 24$

$f(5) = 120$

Рис. 1.28. Пример использования оператора **return**

1.8. Перехват ошибок

При программировании в MathCad можно выполнить дополнительную обработку ошибок. Если предполагается, что в каком-либо месте программы выполнение программного кода вызывает ошибку, то эту ошибку можно выявить с помощью оператора **on error**. Чтобы вставить оператор **on error** в программный модуль, необходимо ввести линии ввода в код и нажать кнопку «On error» на панели инструментов «Программирование». В результате появится строка с двумя заполнителями и оператором **on error**. В правом заполнителе необходимо ввести выражение, которое должно выполняться в указанной строке программного модуля. В левом заполнителе указывается выражение, которое должно выполняться вместо правого выражения, если при его выполнении возникает ошибка. На рис. 1.29 дан пример перехвата ошибки, возникающей при делении на ноль.

$$f(x) := \left| \begin{array}{l} y \leftarrow x \\ \text{"ошибка пользователя: деление на ноль"} \quad \text{on error } \frac{1}{y} \end{array} \right.$$


$$f(2) \rightarrow \frac{1}{2}$$

$$f(0) \rightarrow \text{"ошибка пользователя: деление на ноль"}$$

Рис. 1.29. Пример перехвата ошибки «деление на ноль»

Отметим, что операторы программирования вставляются в текст программы только с помощью кнопок панели инструментов «Программирование» и их имена нельзя набирать на клавиатуре, поскольку они не будут восприняты корректно.

1.9. Символьные вычисления

Символьные вычисления в пакете MathCad можно осуществлять с помощью команд меню, оператора символьного вывода  и ключевых слов символьного процессора. Символьные преобразования с помощью меню касаются только одного, выделенного в данный момент выражения. Поэтому на них не влияют формулы, находящиеся в документе MathCad выше этого выделенного выражения. Оператор символьного вывода, наоборот, учитывает все предыдущее содержимое документа при выдаче результата. Для символьных вычислений при помощи команд предназначено главное меню «Символика» (рис. 1.30).

Рассмотрим последовательность действий при проведении алгебраических преобразований с помощью команд меню на примере разложения на сомножители выражения $\sin(3x)$.

1. Ввести выражение $\sin(3x)$ и выделить его целиком.
2. Выбрать в главном меню пункт «Символика», а затем пункт «Развернуть» (см. рис. 1.30).
3. Результат разложения появится ниже в виде еще одной строки (рис. 1.31).

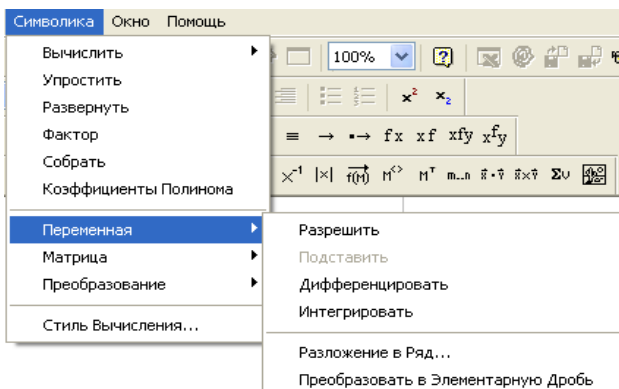


Рис. 1.30. Меню «Символика»

$$\sin(3 \cdot x)$$

$$4 \sin(x) \cdot \cos(x)^2 - \sin(x)$$

Рис. 1.31. Символьное разложение выражения

Отметим, что символьные расчеты с помощью меню возможны лишь над каким-либо объектом. Для того чтобы правильно осуществить аналитическое преобразование, необходимо предварительно выделить тот объект, к которому оно будет относиться. Если же выделить часть формулы, то соответствующее преобразование будет отнесено к выделенной части.

Ниже приведен пример использования программных модулей в символьных расчетах (рис. 1.32).

$$f(n) := \begin{cases} -1 & \text{if } n < 0 \\ x & \text{on error } \frac{d^n}{dx^n} x^{10} & \text{otherwise} \end{cases}$$

$$f(1) \rightarrow 10 \cdot x^9 \qquad f(-3) \rightarrow -1$$

$$f(10) \rightarrow 3628800 \qquad f(2.1) \rightarrow x$$

Рис. 1.32. Программирование в символьных расчетах

Контрольные вопросы

1. Какая панель используется для вставки программного кода в документы?
2. Каким образом обозначается программный модуль в MathCad?
3. Опишите процедуру создания программного модуля.
4. Как можно вставить строку программного кода в уже созданную программу?
5. С помощью какого оператора осуществляется локальное присваивание?
6. Опишите последовательность действий, выполняемую при вставке условного оператора в программный код.
7. Для чего необходим оператор **otherwise**?
8. Какие операторы цикла существуют в языке программирования пакета MathCad?
9. Какой оператор организует цикл с выходом из него по какому-то логическому условию?
10. Какой оператор позволяет сформировать цикл по некоторой переменной, заданной на определенном диапазоне значений?
11. Каким образом можно задать диапазон значений переменной при использовании цикла **for** ?
12. Какой оператор предназначен для досрочного завершения цикла?
13. Для чего используется оператор **continue** ?
14. Опишите процедуру вставки оператора цикла в программу.
15. Каким образом можно выполнить дополнительную обработку ошибок при программировании в MathCad?
16. В каких случаях используется оператор **return**?
17. Опишите последовательность действий при выполнении символьных расчетов с помощью меню.
18. Как определяется функция пользователя?

2. ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ И ТРАНСЦЕНДЕНТНЫХ УРАВНЕНИЙ

2.1. Постановка задачи

Определение корней алгебраических или трансцендентных уравнений и их систем – одна из важнейших задач вычислительной математики. Пусть дано уравнение

$$f(x) = 0, \quad (2.1)$$

где функция $f(x)$ определена и непрерывна в конечном или бесконечном интервале $a < x < b$. Всякое значение ξ , обращающее функцию $f(x)$ в нуль, т. е. $f(\xi) \equiv 0$, называется корнем уравнения (2.1), или нулем функции $f(x)$. Уравнение (2.1) называется алгебраическим, если заданная функция есть полином n -й степени:

$$f(x) = P(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n = 0, \quad a_0 \neq 0.$$

Всякое неалгебраическое уравнение называется трансцендентным.

Отметим, что найти корни уравнения (2.1) точно удастся лишь в очень редких случаях. Обычно уравнение (2.1) содержит коэффициенты, которые известны лишь приближенно. Разработаны многочисленные методы численного решения уравнения (2.1), позволяющие находить приближенные значения корней. Предположим, что уравнение (2.1) имеет лишь изолированные корни, т. е. для каждого корня существует окрестность, не содержащая других корней этого уравнения.

Приближенное нахождение решения уравнения (2.1) складывается из двух этапов:

– отделение корней, т. е. отыскание достаточно тесных интервалов $[a, b]$, в которых содержится один и только один корень уравнения (2.1);

– уточнение приближенных корней до заданной степени точности.

При графическом отделении корней уравнение (2.1) надо преобразовать к виду

$$\varphi_1(x) = \varphi_2(x) \quad (2.2)$$

и построить графики функций

$$y_1 = \varphi_1(x), \quad y_2 = \varphi_2(x). \quad (2.3)$$

Корнями уравнения (2.1) являются абсциссы точек пересечения этих графиков. В общем случае при построении графиков функций (2.3) следует исследовать поведение каждой из функций (2.3), определить точки пересечения этих функций с осями x и y и вычислить ряд их промежуточных, наиболее характерных значений. Полезно также определить точки минимума и максимума функций.

2.2. Метод половинного деления

Пусть функция $f(x)$ в уравнении (2.1) определена и непрерывна на интервале $[a, b]$ и $f(a)f(b) < 0$. Для нахождения корня уравнения делим отрезок пополам. В качестве начального приближения выберем $c = (a + b) / 2$, затем исследуем функцию на концах отрезков $[a, c]$ и $[c, b]$. Выбирается тот отрезок, у которого значение функции на концах имеет противоположные знаки. Новый, суженный отрезок снова делим пополам и проводим такой же анализ и т. д. Процесс продолжается до тех пор, пока не выполнится условие $|b - a| < \varepsilon$, где $\varepsilon > 0$ – сколь угодно малое число. Отметим, что метод половинного деления очень прост и здесь можно обеспечить практически любую точность. Следует обратить внимание на медленную сходимость метода, когда за один шаг интервал, в котором находится корень, сужается всего лишь в два раза.

Пример 2.1. Найти методом половинного деления приближенное решение уравнения $f(x) = x^4 - 13x^2 + 36 - \frac{1}{x}$.

Решение. Для того чтобы решить уравнение, необходимо построить график заданной функции (рис. 2.1).

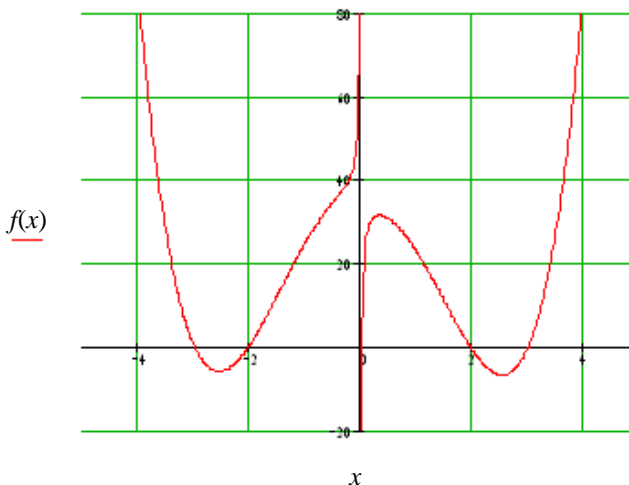


Рис. 2.1. График функции $f(x) = x^4 - 13x^2 + 36 - \frac{1}{x}$

Ниже приведен программный код функции определения одного корня уравнения.

```

Bisect(f,a,b,eps,k) := while |a - b| > 2eps ∧ k ≠ 0
                        |
                        |  ξ ← (a + b) / 2
                        |  break if f(ξ) = 0
                        |  b ← ξ if f(ξ) · f(a) < 0
                        |  a ← ξ otherwise
                        |  k ← k - 1
                        |
                        ξ

```

Рис. 2.2. Программный модуль для вычисления корня уравнения методом половинного деления

Ниже приведены результаты определения всех корней уравнения на заданном интервале и с заданной точностью $\varepsilon = 0.001$, (рис. 2.3– 2.7).

$$a := -5 \quad b := -2.5$$

$$k := \log\left(\frac{|a - b|}{\varepsilon}, 2\right)$$

$$--- k = 11.288 -----$$

$$\text{Bisect}(f, a, b, \varepsilon, k) = -2.99$$

Рис. 2.3. Первый корень

$$a := 0.001 \quad b := 1$$

$$k := \log\left(\frac{|a - b|}{\varepsilon}, 2\right)$$

$$k = 9.964$$

$$\text{Bisect}(f, a, b, \varepsilon, k) = 0.026$$

Рис. 2.5. Третий корень

$$a := -2.5 \quad b := -0.001$$

$$k := \log\left(\frac{|a - b|}{\varepsilon}, 2\right)$$

$$k = 11.287$$

$$\text{Bisect}(f, a, b, \varepsilon, k) = -2.025$$

Рис. 2.4. Второй корень

$$a := 1 \quad b := 2.5$$

$$k := \log\left(\frac{|a - b|}{\varepsilon}, 2\right)$$

$$k = 10.551$$

$$\text{Bisect}(f, a, b, \varepsilon, k) = 1.974$$

Рис. 2.6. Четвертый корень

$$a := 2.5 \quad b := 5$$

$$k := \log\left(\frac{|a - b|}{\varepsilon}, 2\right)$$

$$k = 11.288$$

$$\text{Bisect}(f, a, b, \varepsilon, k) = 3.011$$

Рис. 2.7. Пятый корень

2.3. Метод простых итераций

Пусть корни отделены и $[a, b]$ содержит единственный корень. Уравнение (2.1) приведем к итерационному виду

$$x = \varphi(x), \quad (2.4)$$

где функция $\varphi(x)$ дифференцируема на $[a, b]$. Возьмем приближенное значение корня x_0 и найдем более точный результат при помощи формулы

$$x_{n+1} = \varphi(x_n), \quad n = 0, 1, 2, \dots \quad (2.5)$$

Повторяя этот процесс, можно получить значение корня с любой степенью точности, если выполняется условие

$$|\varphi'(x)| < 1 \quad (2.6)$$

на интервале $[\xi, x_0]$, где $x = \xi$ – точное значение корня. Если же условие (2.6) не выполняется, то функцию $\varphi(x)$ необходимо подбирать в виде

$$\varphi(x) = x + kf(x), \quad (2.7)$$

где k находится из условия

$$|\varphi'(k, x)| = |1 + kf'(x)| < 1. \quad (2.8)$$

Последнее условие гарантирует сходимость последовательности $x_1, x_2, \dots, x_{n-1}, x_n$ к корню ξ . В некоторых случаях непосредственное применение формулы (2.5) приводит к расходящемуся процессу. Этим трудностей можно избежать, если перейти к обратной функции. Например, при $x > 0$ уравнение $2 + x = e^x$ методом итераций решить нельзя, так как при $x > 0$ имеем $\varphi'(x) = (e^x - 2)' = e^x > 1$ и согласно условию (2.6) процесс будет расходящимся. Но в этом случае метод итераций можно применить, если сначала перейти к обратной функции. Тогда получим уравнение $x = \ln(2 + x)$, для которого при $x > 0$ имеем $\varphi'(x) = \frac{1}{2 + x} < 1$. Читателю рекомендуется самостоятельно выяснить геометрическим путем, почему при $|\varphi'(x)| \geq 1$ метод итераций приводит к расходящемуся процессу.

Пример 2.2. Найти методом простых итераций положительный корень уравнения $\cos x - \frac{1}{x} \sin x = 0$.

Решение. Заменяем исходное уравнение эквивалентным ему уравнением $x = \operatorname{tg} x$ и, построив графики функций $y_1 = x$, $y_2 = \operatorname{tg} x$ (это читателю предлагается выполнить самостоятельно), находим, что можно взять $x_0 = 3\pi/2 \approx 4.7$. Но к уравнению $x = \operatorname{tg} x$ непосредственно метод итераций применить нельзя, так как при любом значении x имеем $\varphi'(x) = (\operatorname{tg} x)' = \frac{1}{\cos^2 x} \geq 1$ и условие (2.6) не выполняется. Поэтому перейдем к обратной функции, т. е. к уравнению $x = \operatorname{arctg} x$, для которого $\varphi'(x) = (\operatorname{arctg} x)' = \frac{1}{1+x^2} < 1$ при $x \neq 0$. Уравнение $x = \operatorname{arctg} x$, которое эквивалентно исходному уравнению, решаем при помощи формулы $x_{n+1} = \operatorname{arctg} x_n$, где $x_0 = 4.7$. Итерационный процесс заканчиваем, когда с необходимой точностью совпадут значения x_n и x_{n+1} . Выполнив вычисления, приведенные в таблице, получаем $x = 4.4934$.

Т а б л и ц а

n	0	1	2	3
x_n	4.7	4.5	4.494	4.4934
$\operatorname{arctg} x_n$	4.5	4.494	4.4934	4.4934

При использовании метода простых итераций основным моментом является выбор функции $\varphi(x)$ в уравнении (2.4). Метод итераций широко распространен в математике, его применяют при решении дифференциальных, интегральных, интегродифференциальных уравнений и во многих других вычислительных задачах.

Пример 2.3. Найти методом простых итераций приближенное решение уравнения $\phi(x) = \frac{\sin(x^2) + 1}{6}$.

Решение. Для того чтобы найти решение исходного уравнения, построим график заданной функции (рис. 2.8).

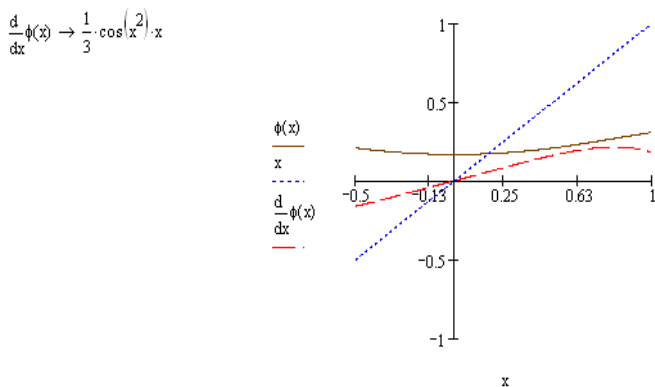


Рис. 2.8. График функции $\phi(x) = \frac{\sin(x^2) + 1}{6}$ и ее производной

Ниже приведены программный код функции определения корня уравнения по методу простых итераций и полученный результат:

```
SimpleIteration(f, x0, eps) :=
  x ← x0
  k ← 0
  while k ≥ 0
    y_k ← f(x)
    break if |y_k - x| ≤ eps
    x ← y_k
    k ← k + 1
  y
```

$\text{SimpleIteration}(\phi, 0.25, \epsilon) = \begin{pmatrix} 0.17707655 \\ 0.17189183 \\ 0.17159042 \\ 0.17157317 \end{pmatrix}$

Рис. 2.9. Программный модуль для вычисления корня уравнения методом простых итераций

2.4. Метод Ньютона (метод касательных)

Пусть корень уравнения (2.1) отделен на интервале $[a, b]$, причем первая и вторая производные $f'(x)$ и $f''(x)$ непрерывны и $f(a)f(b) < 0$, $f(a)f''(a) > 0$ при $a \leq x \leq b$. Найдя какое-нибудь n -е

приближение корня $x_n \approx \xi$ ($a \leq x_n \leq b$), можно уточнить его по методу Ньютона следующим образом:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 1, 2, \dots, K. \quad (2.9)$$

Значение x_0 рекомендуется выбирать из условия выполнения неравенства $f(x_0) \cdot f''(x_0) > 0$. Процесс вычислений заканчивается при выполнении условия $|x_n - x_{n-1}| < \varepsilon$. Для оценки погрешностей можно воспользоваться формулой $|x_n - \xi| \leq \frac{|f(x_n)|}{\mu}$, где $\mu = \min |f'(x)|$ на интервале $[a, b]$. На рис. 2.10 графически проиллюстрирован процесс нахождения корня методом Ньютона.

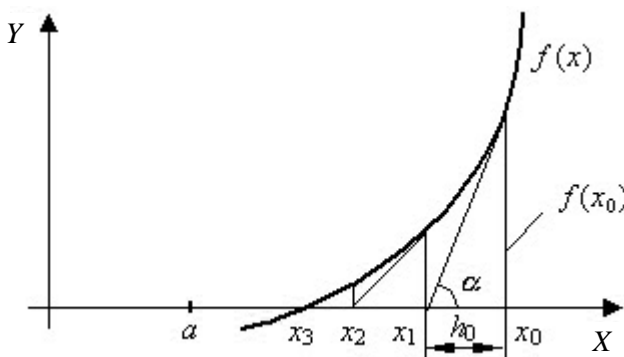


Рис. 2.10. Графическое изображение процесса нахождения корня методом Ньютона

Удобнее пользоваться более простой формулой

$$x_{n+1} = x_n - \alpha f(x_n), \quad n = 1, 2, \dots, K, \quad (2.10)$$

где $x_0 = a$, $\alpha = 1/f'(a)$, которая дает примерно ту же точность, что и формула (2.9).

Пример 2.4. Найти методом Ньютона приближенное решение уравнения $f(x) = \ln(x) - \frac{1}{1+x^2}$.

Решение. Для того чтобы найти приближенное решение заданного уравнения, сначала построим график заданной функции (рис. 2.11).

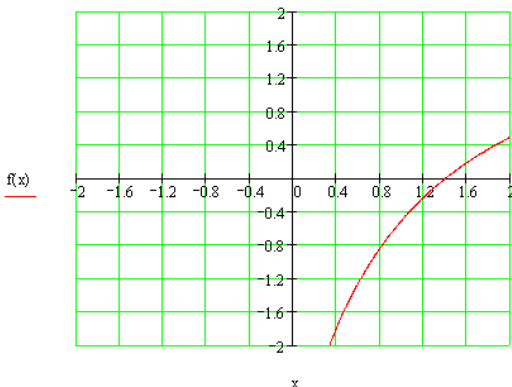


Рис. 2.11. График функции $f(x) = \ln(x) - \frac{1}{1+x^2}$

Далее приведены программный код функции определения корня уравнения по методу Ньютона и полученный результат (рис. 2.12).

$x := 2.5$

```

Newton(f,x0,eps) :=
  k ← 0
  x ← x0
  while k ≥ 0
    y_k ← f(x)
    (break) if  $\left| \frac{y_k}{\frac{d}{dx} f(x)} \right| < \text{eps}$ 
    x ← x -  $\frac{y_k}{\frac{d}{dx} f(x)}$ 
    k ← k + 1
  x
Newton(f,0.5,0.0001) = 1.40132121

```

Рис. 2.12. Программный модуль для вычисления корня уравнения методом Ньютона

2.5. Метод хорд (метод линейной аппроксимации)

Пусть дано уравнение (2.1), где функция $f(x)$ определена и непрерывна на интервале $[a, b]$ и выполняется соотношение $f(a)f(b) < 0$. Пусть для определенности $f(a) < 0$, а $f(b) > 0$. Геометрически этот метод эквивалентен замене кривой $f(x)$ хордой, проходящей через точки $(a, f(a))$ и $(b, f(b))$. Уравнение хорды имеет вид

$$\frac{x-a}{b-a} = \frac{y-f(a)}{f(a)-f(b)}. \quad (2.11)$$

Учитывая, что $y=0$ при $x=x_1$, получаем первое приближение корня:

$$x_1 = a - \frac{f(a)}{f(a)-f(b)}(b-a). \quad (2.12)$$

Для получения второго приближения формулу (2.12) применяем к тому из отрезков $[a, x_1]$ или $[x_1, b]$, на концах которого значения функции $f(x)$ имеют противоположные знаки. Так же строятся и следующие приближения. Если на отрезке $[a, b]$ вторая производная $f''(x)$ сохраняет постоянный знак, метод хорд сводится к двум различным вариантам. В первом варианте в качестве неподвижной выбирается точка a , а точка b приближается к корню ξ , т. е. получаем

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n)-f(a)}(x_n-a). \quad (2.13)$$

Во втором варианте в качестве неподвижной выбирается точка b , а точка a приближается к корню ξ , т. е. имеем уже другую вычислительную формулу

$$x_{n+1} = x_n - \frac{f(x_n)}{f(b)-f(x_n)}(b-x_n). \quad (2.14)$$

Рекомендуется в качестве неподвижной выбирать ту точку, в которой выполняется соотношение $f(x)f''(x) > 0$. Для оценки абсолютной

погрешности приближенного корня x_n можно воспользоваться формулой $|\xi - x_n| \leq \frac{|f(x_n)|}{\mu}$, где $\mu = \min |f'(x)|$ на интервале $[a, b]$.

Контрольные вопросы

1. Какие приближенные методы решения нелинейных уравнений вы знаете?
2. Из каких этапов складывается приближенное нахождение корней нелинейных уравнений?
3. Для чего необходим первый этап – отделение корней?
4. Что можно сказать о точности метода половинного деления, метода касательных, метода простых итераций?
5. По каким параметрам можно провести сравнение методов половинного деления, Ньютона, простых итераций?
6. Всегда ли можно найти решение нелинейного уравнения методами половинного деления, Ньютона, простых итераций?
7. Опишите алгоритм метода касательных (Ньютона). Сформулируйте условие выбора начальной точки.
8. Сформулируйте условия окончания вычислений по методу простых итераций.
9. Какие способы отделения корней вы знаете?
10. Опишите процедуру решения нелинейных уравнений с использованием метода половинного деления.

3. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

3.1. Формула прямоугольников

Задача численного интегрирования функции заключается в вычислении определенного интеграла на основании некоторого ряда значений подынтегральной функции. Рассмотрим способы приближенного вычисления определенных интегралов

$$\int_a^b f(x)dx, \quad (3.1)$$

когда интеграл заменяется конечной суммой

$$\sum_{k=0}^n C_k f(x_k), \quad (3.2)$$

где C_k – числовые коэффициенты, а $x_k \in [a, b]$, $k=0, 1, \dots, n$.

Приближенное равенство

$$\int_a^b f(x)dx \approx \sum_{k=0}^n C_k f(x_k) \quad (3.3)$$

называется квадратурной формулой, а x_k – узлами формулы. Погрешность определяется соотношением

$$R_n = \int_a^b f(x)dx - \sum_{k=0}^n C_k f(x_k). \quad (3.4)$$

В общем случае погрешность зависит от выбора коэффициентов C_k и от расположения узлов x_k . Если на отрезке $[a, b]$ ввести равномерную сетку с шагом h , то $x_i = a + i \cdot h$, где $i=0, 1, \dots, n$, $h = (b - a) / n$, и тогда выражение (3.1) можно представить в виде суммы интегралов по частичным отрезкам:

$$\int_a^b f(x)dx = \sum_{i=1}^n \int_{x_{i-1}}^{x_i} f(x)dx. \quad (3.5)$$

В случае замены подынтегральной функции на частичном отрезке $[x_{i-1}, x_i]$ полиномом Лагранжа нулевого порядка, построенным в точке $x_{i-0.5} = x_i - 0.5 \cdot h$, получаем формулу

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx f(x_{i-0.5})h. \quad (3.6)$$

После подстановки формулы (3.6) в выражение (3.5) имеем формулу средних прямоугольников

$$\int_a^b f(x)dx \approx \sum_{i=1}^n f(x_{i-0.5})h, \quad (3.7)$$

погрешность которой определяется выражением

$$|R_n| \leq M_2 \frac{|b-a|h^2}{24}, \quad (3.8)$$

где $M_2 = \max |f''(x)|$, $x \in [a, b]$.

3.2. Метод трапеций

Общий подход к решению задачи заключается в разбиении отрезка $[a, b]$ на множество отрезков меньших размеров и вычислении интеграла как суммы приближенно вычисленных площадей полосок, полученных при таком разбиении.

Метод трапеций – приближенный, потому что при вычислении площади трапеции кривую заменяем на прямую. Для каждой маленькой трапеции эта погрешность своя. Сумма погрешностей должна давать отклонение полученного значения от истинного.

Итак, предполагается, что отрезок $[a, b]$ разбит на n частей точками x_i , что необходимо при построении полинома Лагранжа $L_n(x)$ первой степени:

$$f(x) \approx L_{1,i}(x) = \frac{1}{h} [(x - x_{i-1})f(x_i) - (x - x_i)f(x_{i-1})].$$

Для равноотстоящих узлов $x_i = x_0 + ih$, $h = (b - a) / n$, $x_0 = a$, $x_n = b$ имеем

$$\int_a^b f(x) dx \approx \int_a^b L_n(x) dx. \quad (3.9)$$

Формула трапеций имеет следующий вид:

$$\int_a^b f(x) dx \approx h \left(\frac{1}{2} (f_0 + f_n) + f_1 + f_2 + \dots + f_{n-1} \right), \quad (3.10)$$

где f_i – значения функции в узлах интерполяции.

Получаем следующую оценку погрешности метода интегрирования по формуле трапеций:

$$|R_n| \leq M_2 \frac{|b - a| h^2}{12}, \quad (3.11)$$

где $M_2 = \max |f''(x)|$, $x \in [a, b]$.

Пример 3.1. Вычислить методом трапеций интеграл $f(x) = \ln(\sin(x) + e^x)$ на интервале $[0, 3]$, $n = 3$.

Решение. Для того чтобы найти приближенное решение заданного уравнения, сначала построим график заданной функции (рис. 3.1).

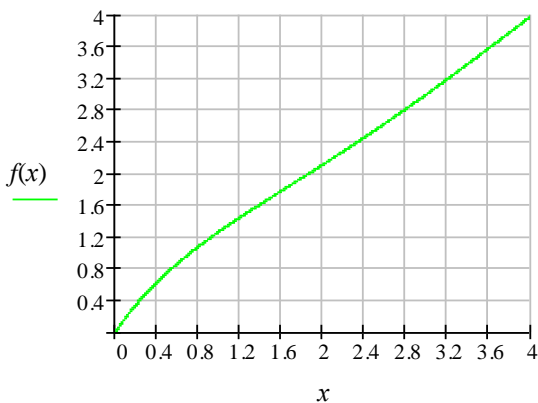


Рис. 3.1. График функции $f(x) = \ln(\sin(x) + e^x)$

Ниже приведен программный модуль, осуществляющий приближенное вычисление интеграла методом трапеций (рис.3.2).

```

Trap(a,b,f,n) :=
  x0 ← a
  h ←  $\frac{b-a}{n}$ 
  for i ∈ 1..n
    xi ← xi-1 + h
    Ji ←  $\frac{h}{2} \cdot (f(x_{i-1}) + f(x_i))$ 
  J ←  $\sum_{i=1}^n J_i$ 
  J

F := Trap(a,b,f,n)          F = 4.94671

```

Рис. 3.2. Программный модуль для вычисления корня уравнения методом трапеций

Далее приведен программный модуль функции для определения погрешности вычислений (рис.3.3).

```

Error(a,b,f,n) :=
  h ←  $\frac{b-a}{n}$ 
  KL ← 0
  for k ∈ 1..n
    ξ ← a + (2k-1) ·  $\frac{h}{2}$ 
    gk ←  $\frac{d^2}{d\xi^2} f(\xi)$ 
    KL ← KL + gk
  KL ←  $\frac{KL}{n} \cdot \left[ \frac{-(b-a) \cdot h^2}{12} \right]$ 

R := Error(a,b,f,n)          R = 0.020182

```

Рис. 3.3. Программный модуль для определения погрешности вычислений

3.3. Формула Симпсона

В этом методе подынтегральная функция аппроксимируется интерполяционным многочленом Лагранжа второй степени:

$$f(x) \approx L_{2,i}(x) = \frac{2}{h^2}[(x - x_{i-1/2})(x - x_i)f(x_{i-1}) - \\ - 2(x - x_{i-1})(x - x_i)f(x_{i-1/2}) + (x - x_{i-1})(x - x_{i-1/2})f(x_i)], \quad (3.12)$$

где $x \in [x_{i-1}, x_i]$, $h = x_i - x_{i-1}$.

Выполнив интегрирование, получим формулу Симпсона (формулу парабол)

$$\int_{x_{i-1}}^{x_i} f(x) dx \approx \int_{x_{i-1}}^{x_i} L_{2,i}(x) dx = \frac{h}{6}(f_{i-1} + 4f_{i-1/2} + f_i). \quad (3.13)$$

Надо отметить, что во многих случаях формула Симпсона оказывается более точной. На отрезке $[a, b]$ формула Симпсона имеет вид

$$\int_a^b f(x) dx \approx \frac{h}{6}(f_0 + f_n + 2(f_1 + f_2 + \dots + f_{n-1}) + \\ + 4(f_{1/2} + f_{3/2} + \dots + f_{n-1/2})). \quad (3.14)$$

Если ввести обозначения $x_i = a + 0.5hi$, где $i = 0, 1, 2, \dots, 2n$, то формула Симпсона преобразуется к виду

$$\int_a^b f(x) dx \approx \frac{h}{6}(f_0 + f_{2n} + 2(f_2 + f_4 + \dots + f_{2n-2}) + \\ + 4(f_1 + f_3 + \dots + f_{2n-1})). \quad (3.15)$$

Если формулу (3.13) перепишем в виде

$$\frac{1}{h} \int_{x_{i-1}}^{x_i+h} f(x) dx \approx \frac{2}{3}f_{i-1/2} + \frac{1}{6}(f_{i-1} + f_i), \quad (3.16)$$

то получим, что формула Симпсона основана на приближенной замене среднего значения \bar{f} функции $f(x)$ на промежутке от x_i до $x_i + h$ на

$$\bar{f} \approx \frac{2}{3} f_{i-1/2} + \frac{1}{6} (f_{i-1} + f_i). \quad (3.17)$$

Формула трапеций дала бы следующий результат (проверьте!):

$$\bar{f} \approx \frac{1}{2} f_{i-1/2} + \frac{1}{4} (f_{i-1} + f_i). \quad (3.18)$$

Замеим, что формула (3.18) точна для линейных функций (вида $y = ax + b$), а формула (3.17) – и для квадратичных функций (вида $y = ax^2 + bx + c$). На самом деле формула (3.17) является точной и для кубических функций (вида $y = ax^3 + bx^2 + cx + d$). Абсолютную погрешность можно определить исходя из формулы

$$|R_n| \leq M_4 \frac{|b-a|h^4}{180}, \quad (3.19)$$

где $M_4 = \max |f^{(4)}(x)|$, $x \in [a, b]$. Для обеспечения заданной точности ε при вычислении интеграла шаг h определяется из неравенства $M_4 \frac{|b-a|h^4}{180} \leq \varepsilon$ и затем округляется в сторону уменьшения так, чтобы $n = (b-a)/h$ было целым числом. Так как определение шага вычислений h и связанного с ним числа n несколько затруднительно, то на практике шаг h определяют грубой прикидкой, а затем, получив результат, удваивают число n , то есть шаг h делят пополам.

Пример 3.2. Найти по формуле трапеций и по формуле Симпсона интеграл $I = \int_0^1 \frac{1}{1+x} dx$.

Решение. Разобьем промежуток интегрирования на части и вычислим интеграл. Найдем необходимые значения подынтегральной функции (табл. 3.1).

Таблица 3.1

x	0	0.5	1
y	1	0.6667	0.5

Формула трапеций дает

$$I = \frac{1}{2} \left(\frac{1+0.5}{2} + 0.6667 \right) = 0.7083 .$$

По формуле Симпсона получаем

$$I = \frac{1}{6} (1 + 4 \cdot 0.6667 + 0.5) = 0.6944 .$$

Так как $I = \ln(1+x)|_0^1 = \ln 2 = 0.6931$, то формула Симпсона дала ошибку примерно 0.2 %, а формула трапеций 2 %.

Пример 3.3. Найти по формуле трапеций и по формуле Симпсона интеграл $I = \int_0^1 \frac{\ln(1+x)}{1+x^2} dx$. Заметим, что интеграл $\int \frac{\ln(1+x)}{1+x^2} dx$ не берется в элементарных функциях.

Решение. Разобьем промежуток на части и найдем значения подынтегральной функции (табл. 3.2).

Т а б л и ц а 3.2

x	0	0.5	1
y	0	0.324	0.346

По формуле трапеций получаем

$$I = \frac{1}{2} \left(\frac{0.346}{2} + 0.324 \right) = 0.248 .$$

Формула Симпсона дает

$$I = \frac{1}{6} (0.324 \cdot 4 + 0.346) = 0.274 .$$

Точное значение этого интеграла 0.272, поэтому ошибка при вычислениях по методу трапеций составила около 10 %, а по формуле Симпсона – менее 1 %.

Преимущество формулы Симпсона особенно сказывается при увеличении числа n интервалов разбиения. Покажите, что при этом

ошибка формулы трапеций убывает обратно пропорционально n^2 , а ошибка формулы Симпсона – обратно пропорционально n^4 .

Пример 3.4. В пакете MathCad вычислить по формуле Симпсона интеграл $\int_1^5 \ln\left(\frac{x}{1+x}\right) dx$ на интервале $[1,5]$ при $n=8$.

Решение. Для того чтобы найти приближенное значение интеграла, сначала построим график подынтегральной функции (рис. 3.4).

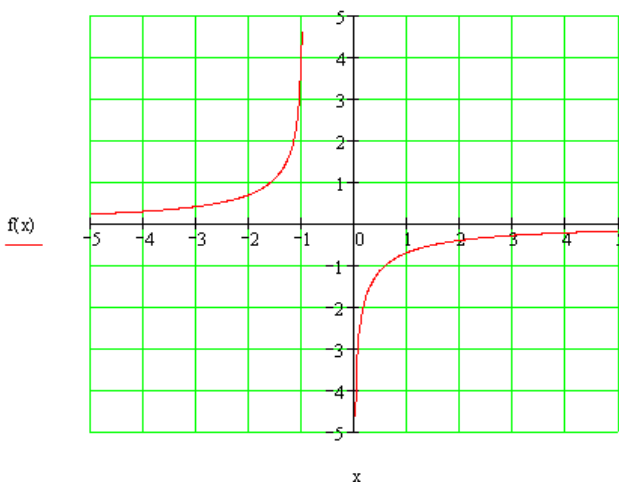


Рис. 3.4. График функции $f(x) = \ln\left(\frac{x}{1+x}\right)$

Ниже приведен программный модуль, осуществляющий приближенное вычисление интеграла по формуле Симпсона. Заметим, что приведенные вычисления показывают, насколько быстро и просто численные методы позволяют находить интегралы. Если для интеграла нужна точная формула, то иногда быстрее и легче найти его методами численного интегрирования (рис. 3.5).

Далее приведен программный модуль функции для определения погрешности вычислений (рис. 3.6).

$$h := \frac{b-a}{n} \quad h = 0.5$$

$$\text{Simps}(a, b, f, n) := \left| \begin{array}{l} x_0 \leftarrow a \\ h \leftarrow \frac{b-a}{n} \\ J \leftarrow f(a) + f(b) \\ j \leftarrow 2 \\ \text{for } k \in 1..(n-1) \\ \quad \left| \begin{array}{l} j \leftarrow 6-j \\ x_k \leftarrow x_{k-1} + h \\ J \leftarrow J + j \cdot f(x_k) \end{array} \right. \\ J \leftarrow \frac{h \cdot J}{3} \\ J \end{array} \right.$$

$$H := \text{Simps}(a, b, f, n) \quad H = -1.3175304$$

Рис. 3.5. Программный модуль для вычисления интеграла по формуле Симпсона

$$\text{Error}(a, b, f, n) := \left| \begin{array}{l} h \leftarrow \frac{b-a}{n} \\ LL \leftarrow 0 \\ \text{for } k \in 1..n \\ \quad \left| \begin{array}{l} \xi \leftarrow a + (2k-1) \cdot \frac{h}{2} \\ g_k \leftarrow \frac{d^4}{d\xi^4} f(\xi) \\ LL \leftarrow LL + g_k \end{array} \right. \\ LL \leftarrow \frac{LL}{n} \cdot \left[\frac{-(b-a) \cdot h^4}{180} \right] \end{array} \right.$$

$$R := \text{Error}(a, b, f, n) \quad R = 5.354 \times 10^{-4}$$

Рис. 3.6. Программный модуль для определения погрешности вычислений

Контрольные вопросы

1. Сформулируйте задачу численного интегрирования.
2. Каким образом осуществляется построение квадратурных формул?
3. В каких случаях осуществляется численное интегрирование?
4. Каковы преимущества формулы парабол по сравнению с формулой трапеций?
5. Предложите и обоснуйте пути повышения точности расчетов при решении задачи численного интегрирования методом трапеций.
6. В каких случаях приближенные формулы трапеций и парабол оказываются точными?
7. Как величина шага влияет на точность численного интегрирования?
8. Верны ли формулы трапеций и Симпсона для неравноотстоящих узлов?

4. ПРИБЛИЖЕННОЕ РЕШЕНИЕ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

4.1. Метод Ньютона

Рассмотрим систему n нелинейных уравнений с n неизвестными

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \quad (4.1)$$

с действительными левыми частями. Систему (4.1) можно представить в матричном виде

$$f(x) = 0. \quad (4.2)$$

Здесь приняты следующие обозначения:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} - \text{вектор аргументов, } f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} - \text{вектор-функция.}$$

Основная идея метода Ньютона состоит в выделении из уравнений системы линейных частей, которые являются главными при малых приращениях аргументов. Для решения системы (4.2) будем использовать метод последовательных приближений. Предположим, что найдено k -е приближение одного из изолированных корней уравнения (4.2). Тогда точный корень уравнения (4.2) можно записать в виде

$$x = x^{(k)} + \varepsilon^{(k)}, \quad (4.3)$$

где $\varepsilon^{(k)} = (\varepsilon_1^{(k)}, \varepsilon_2^{(k)}, K, \varepsilon_n^{(k)})$ – погрешность корня на n -м шаге. Предположим, что функция $f(x)$ непрерывно дифференцируема в некоторой выпуклой области, содержащей x и $x^{(k)}$. Тогда, подставив выражение (4.3) в уравнение (4.2), разложим левую часть полученного уравнения в ряд Тейлора по степеням малого вектора $x^{(k)}$, ограничиваясь при этом линейными членами:

$$f(x^{(k)} + \varepsilon^{(k)}) = f(x^{(k)}) + f'(x^{(k)})\varepsilon^{(k)} = 0. \quad (4.4)$$

Отметим, что под производной $f'(x)$ следует понимать матрицу Якоби системы функций f_1, f_2, K, f_n относительно переменных (x_1, x_2, K, x_n) , т. е.

$$f'(x) = W(x) = \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & K & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & K & \frac{\partial f_2}{\partial x_n} \\ K & K & K & K \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & K & \frac{\partial f_n}{\partial x_n} \end{vmatrix}. \quad (4.5)$$

Выражение (4.5) в краткой форме можно представить в виде

$$f'(x) = W(x) = \left[\frac{\partial f_i}{\partial x_j} \right], \quad (i, j = 1, 2, K, n). \quad (4.6)$$

Подставляя выражение (4.6) в уравнение (4.4), получаем

$$f(x^{(k)}) + W(x^{(k)})\varepsilon^{(k)} = 0. \quad (4.7)$$

Далее, предполагая, что матрица $W(x^{(k)})$ неособенная, записываем

$$\varepsilon^{(k)} = -W^{-1}(x^{(k)})f(x^{(k)}). \quad (4.8)$$

Наконец, подставляя выражение (4.8) в формулу (4.3), окончательно получаем

$$x^{(k+1)} = x^{(k)} - W^{-1}(x^{(k)})f(x^{(k)}), \quad k = 0, 1, K, m. \quad (4.9)$$

В качестве нулевого приближения в вычислительной формуле Ньютона (4.9) можно взять приближенное значение искомого корня. Отметим, что в достаточно малой окрестности корня уравнения итерационный процесс сходится с квадратичной скоростью. Для окончания итерационного процесса можно использовать условие $\|x^{(k-1)} - x^{(k)}\|^2$. Если начальное приближение выбрано удачно, то метод Ньютона сходится очень быстро.

Пример 4.1. Решить систему нелинейных уравнений методом простых итераций

$$\begin{cases} \operatorname{tg}(xy + 0.3) = x^2, \\ 0.9x^2 + 2y^2 = 1. \end{cases}$$

Решение. Для того чтобы решить систему нелинейных уравнений, проведем преобразования исходных функций, входящих в систему нелинейных уравнений (рис. 4.1), а затем построим графики полученных функций (рис. 4.2).

$$f1(x, y) \equiv \tan(xy + 0.3) - x^2 \qquad f2(x, y) \equiv 0.9x^2 + 2y^2 - 1$$

$$f(x, y) := \begin{pmatrix} f1(x, y) \\ f2(x, y) \end{pmatrix} \qquad \varepsilon \equiv 0.0001$$

$$g2(x) := \sqrt{0.5 - 0.45x^2} \qquad g1(x) := \frac{\operatorname{atan}(x^2) - 0.3}{x}$$

Рис. 4.1. Преобразования исходных функций

На рис. 4.3 представлены результаты решения системы нелинейных уравнений по методу простых итераций.

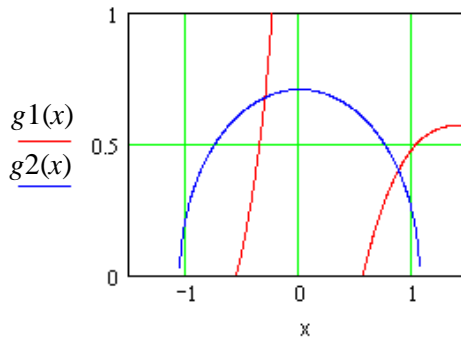


Рис. 4.2. Графики функций $g1(x) = \frac{(\arctg(x^2) - 0.3)}{x}$ и $g2(x) = \sqrt{0.5 - 0.45 \cdot x^2}$

Приближенное значение корней

$$x := \begin{pmatrix} 0.8 \\ -0.3 \end{pmatrix} \quad y := \begin{pmatrix} 0.4 \\ 0.7 \end{pmatrix}$$

Запись уравнений через y, x

$$\Phi(x, y) := \begin{pmatrix} \sqrt{\tan(x \cdot y + 0.3)} \\ \sqrt{\frac{1 - 0.9x^2}{2}} \end{pmatrix} \quad \begin{array}{l} i := 0..50 \\ x2_0 := x_0 \\ y2_0 := y_0 \end{array}$$

$$\begin{pmatrix} x2_{i+1} \\ y2_{i+1} \end{pmatrix} := \Phi(x2_i, y2_i)$$

Вычисление количества итераций:

$$sol_i := \text{if}(|f(x2_i, y2_i)|_1 < \varepsilon, 1, 0) \quad N := \sum_i \text{if}(sol_i = 0, 1, 0) + 1$$

$$N = 50$$

Ответ:

$$\begin{array}{l} x2_N = 0.869603 \\ y2_N = 0.399508 \end{array} \quad f(x2_N, y2_N) = \begin{pmatrix} -7.846301 \times 10^{-5} \\ -1.979554 \times 10^{-4} \end{pmatrix}$$

Рис. 4.3. Решение системы нелинейных уравнений методом простых итераций

Пример 4.2. Найти методом Ньютона положительный корень системы нелинейных уравнений:

$$\begin{cases} x^2 + y^2 - 1 = 0, \\ x^3 - y = 0. \end{cases} \quad (4.10)$$

Решение. Для системы из двух уравнений якобиан в общем виде выглядит следующим образом:

$$f'(x) = W(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix},$$

или его можно записать в виде: $W = \begin{bmatrix} A & C \\ B & D \end{bmatrix}$. Если матрица W неособенная, тогда обратную матрицу вычислим по формуле

$$W^{-1} = \frac{1}{\Delta} \begin{bmatrix} D & -C \\ -B & A \end{bmatrix}, \quad (4.11)$$

где $\Delta = |AD - BC|$. Зададим начальное приближение: $x^{(0)} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$. Найдем $f(x^{(0)})$. После подстановки начального приближения в систему нелинейных уравнений (4.10) получим $f(x^{(0)}) = \begin{pmatrix} -0.5 \\ -0.375 \end{pmatrix}$. Далее воспользуемся формулой (4.9) для выполнения первых трех итераций. Якобиан выглядит следующим образом:

$$W = \begin{bmatrix} 2x & 2y \\ 3x^2 & -1 \end{bmatrix}. \quad (4.12)$$

После подстановки начального приближения в выражение (4.12) получим: $W = \begin{bmatrix} 1 & 1 \\ 0.75 & -1 \end{bmatrix}$. Тогда $\Delta = \det W(x^{(0)}) = -1.75$. Найдем об-

ратную матрицу по формуле (4.11). Получим $W^{-1}(x^{(0)}) = \frac{1}{\Delta} \begin{bmatrix} -1 & -1 \\ -0.75 & 1 \end{bmatrix}$. По формуле (4.9) найдем

$$x^{(1)} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} + \frac{1}{1.75} \begin{bmatrix} -1 & -1 \\ -0.75 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.5 \\ -0.375 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

После подстановки $x^{(1)} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$ в систему нелинейных уравнений (4.10) получим $f(x^{(1)}) = \begin{pmatrix} 0.25 \\ 0.5 \end{pmatrix}$. После подстановки первого прибли-

жения $x^{(1)} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}$ в выражение (4.12) найдем $W = \begin{bmatrix} 2 & 1 \\ 3 & -1 \end{bmatrix}$. Тогда $\Delta = \det W(x^{(1)}) = -5$. Найдем обратную матрицу: $W^{-1}(x^{(1)}) = \frac{1}{\Delta} \begin{bmatrix} -1 & -1 \\ -3 & 2 \end{bmatrix}$. По формуле (4.9) найдем следующее приближение:

$$x^{(2)} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} + \frac{1}{5} \begin{bmatrix} -1 & -1 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.85 \\ 0.55 \end{bmatrix}.$$

Выполнив аналогичные действия, получим

$$x^{(3)} = \begin{pmatrix} 0.85 \\ 0.55 \end{pmatrix} + \frac{1}{4.084} \begin{bmatrix} -1 & -1.1 \\ -2.17 & 2 \end{bmatrix} \begin{bmatrix} 0.025 \\ 0.064 \end{bmatrix} = \begin{bmatrix} 0.827 \\ 0.563 \end{bmatrix}.$$

Для $x^{(3)} = \begin{pmatrix} 0.827 \\ 0.563 \end{pmatrix}$ имеем $f(x^{(3)}) = \begin{pmatrix} 0.0009 \\ 0.0026 \end{pmatrix}$. Другие корни находятся аналогичным образом.

Пример 4.3. Решить методом Ньютона систему нелинейных уравнений

$$\begin{cases} \operatorname{tg}(xy) = x^2, \\ 0.8x^2 + 2y^2 = 1. \end{cases}$$

Решение. Для того чтобы решить систему нелинейных уравнений, проведем преобразования исходных функций, входящих в систему нелинейных уравнений (рис. 4.4).

$$\begin{aligned}
 f1(x, y) &\equiv \tan(x \cdot y) - x^2 & f2(x, y) &\equiv (0.8x^2 + 2y^2 - 1) \\
 f(x, y) &:= \begin{pmatrix} f1(x, y) \\ f2(x, y) \end{pmatrix} & \varepsilon &\equiv 0.00001 \\
 yg(x) &:= \frac{\operatorname{atan}(x^2)}{x} & yh1(x) &:= \sqrt{\frac{(1 - 0.8 \cdot x^2)}{2}} & yh2(x) &:= -\sqrt{\frac{(1 - 0.8 \cdot x^2)}{2}}
 \end{aligned}$$

Рис. 4.4. Преобразования исходных функций

Далее построим графики функций (рис. 4.5).

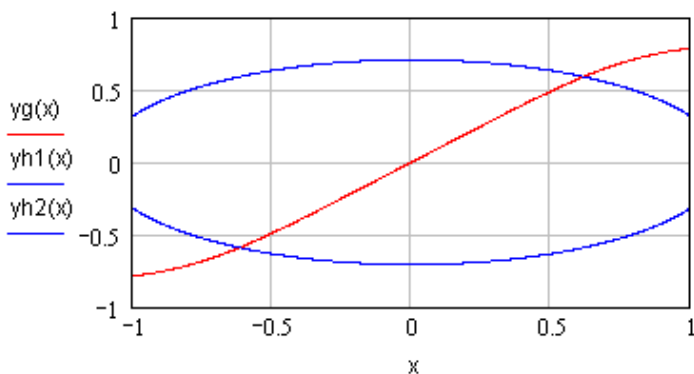


Рис. 4.5. Графики функций $yg(x) = \frac{\operatorname{arctg}(x^2)}{x}$, $yh1(x) = \sqrt{\frac{(1 - 0.8x^2)}{2}}$
и $yh2(x) = -\sqrt{\frac{(1 - 0.8x^2)}{2}}$

Определим начальные приближения $x_0 = 0.612$ и $y_0 = 0.608$. Ниже представлены результаты решения системы нелинейных уравнений по методу Ньютона (рис. 4.6).

$$\delta(x, y) := \begin{pmatrix} \frac{d}{dx} f(x, y)_0 & \frac{d}{dy} f(x, y)_0 \\ \frac{d}{dx} f(x, y)_1 & \frac{d}{dy} f(x, y)_1 \end{pmatrix}^{-1} \quad \begin{array}{l} i := 0 \dots 30 \\ x_0 := x_0 \\ y_0 := y_0 \end{array}$$

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} := \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \delta(x_i, y_i) \cdot \begin{pmatrix} f(x_i, y_i)_0 \\ f(x_i, y_i)_1 \end{pmatrix}$$

Вычисление количества итераций:

$$sol_i := \text{if} \left[\left(\left| f(x_i, y_i)_1 \right| \right)^2 < \varepsilon, 1, 0 \right] \quad N := \sum_i \text{if} (sol_i = 0, 1, 0) + 1$$

$$N = 2$$

Ответ:

$$\begin{array}{l} x_N = 0.617113 \\ y_N = 0.589634 \end{array} \quad f(x_N, y_N) = \begin{pmatrix} -4.484483 \times 10^{-8} \\ 1.504023 \times 10^{-7} \end{pmatrix}$$

Рис. 4.6. Решение системы нелинейных уравнений методом Ньютона

4.2. Метод градиента

Для решения системы (4.1) используется формула

$$x^{(k+1)} = x^{(k)} - \mu_k W_k^T f^{(k)}, \quad (4.13)$$

где $x^{(k)}$ и $x^{(k+1)}$ – векторы неизвестных на k и $k+1$ шагах итераций;
 W_k^T – транспонированная матрица Якоби на k -м шаге;

$$W_k = \left[\frac{\partial f_i^{(k)}}{\partial x_j^{(k)}} \right], \quad (i, j = 1, 2, \dots, n); \quad (4.14)$$

$$\mu_k = \frac{(f^{(k)}, W_k W_k^T f^{(k)})}{(W_k W_k^T f^{(k)}, W_k W_k^T f^{(k)})}. \quad (4.15)$$

Пример 4.4. Методом градиента, ограничиваясь тремя итерациями, найти приближенно положительный корень системы нелинейных уравнений:

$$\begin{cases} yx^2 + y - x + 5 = 0, \\ 2x^2 - yx - 7 = 0. \end{cases} \quad (4.16)$$

Решение. Зададим начальное приближение: $x^{(0)} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$. Найдем $f(x^{(0)})$. После подстановки начального приближения в систему нелинейных уравнений (4.16) получим $f(x^{(0)}) = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$. Далее воспользуемся формулой (4.13) для выполнения первых трех итераций. Матрица Якоби выглядит следующим образом:

$$W = \begin{bmatrix} 2x-1 & x^2+1 \\ 4x-y & -x \end{bmatrix}. \quad (4.17)$$

После подстановки начального приближения в выражение (4.17) получим: $W_0 = \begin{bmatrix} 3 & 5 \\ 9 & -2 \end{bmatrix}$. Далее, используя транспонированную матрицу Якоби на нулевом шаге, найдем

$$W_0^T f(x^{(0)}) = \begin{bmatrix} 3 & 9 \\ 5 & -2 \end{bmatrix} \begin{bmatrix} -2 \\ -1 \end{bmatrix} = \begin{bmatrix} -15 \\ -8 \end{bmatrix}.$$

Для дальнейших преобразований вычислим следующее выражение:

$$W_0 W_0^T f(x^{(0)}) = \begin{bmatrix} 3 & 5 \\ 9 & -2 \end{bmatrix} \begin{bmatrix} -15 \\ -8 \end{bmatrix} = \begin{bmatrix} -85 \\ -119 \end{bmatrix}.$$

Исходя из формулы (4.15) при нулевом приближении получим

$$\mu_0 = \frac{(f^{(0)}, W_0 W_0^T f^{(0)})}{(W_0 W_0^T f^{(0)}, W_0 W_0^T f^{(0)})} = 0.0135.$$

Далее по формуле (4.13) найдем первое приближение:

$$x^{(1)} = x^{(0)} - \mu_0 W_0^T f^{(0)} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} - 0.0135 \begin{bmatrix} -15 \\ -8 \end{bmatrix} = \begin{bmatrix} 2.203 \\ -0.892 \end{bmatrix}.$$

Проведя аналогичные действия, определим $x^{(2)} = \begin{pmatrix} 1.83 \\ -0.86 \end{pmatrix}$,

$x^{(3)} = \begin{pmatrix} 1.637 \\ -0.899 \end{pmatrix}$. Другие корни находят по такой же схеме.

Пример 4.5. Решить систему нелинейных уравнений:

$$\begin{cases} \sin(x + y) - 1.5x = 0.1, \\ x^2 + y^2 = 1. \end{cases}$$

Решение. Для того чтобы решить систему нелинейных уравнений, проведем преобразования исходных функций, входящих в систему нелинейных уравнений (рис. 4.7), а затем построим графики полученных функций (рис. 4.8).

$$f1(x, y) \equiv (\sin(x + y) - 1.5x - 0.1) \quad f2(x, y) \equiv (x^2 + y^2 - 1)$$

$$f(x, y) := \begin{pmatrix} f1(x, y) \\ f2(x, y) \end{pmatrix} \quad \varepsilon \equiv 0.00001$$

$$yg(x) := \arcsin(0.1 + 1.5x) - x$$

$$yh1(x) := \sqrt{1 - x^2}$$

$$yh2(x) := -\sqrt{1 - x^2}$$

Рис. 4.7. Преобразования исходных функций

Начальные приближения $x0 = \begin{pmatrix} 0.59 \\ -0.72 \end{pmatrix}$ и $y0 = \begin{pmatrix} 0.8 \\ 0.78 \end{pmatrix}$ используем в программном модуле, представленном на рис. 4.9. Результаты решения системы нелинейных уравнений методом градиента представим на рис. 4.10.

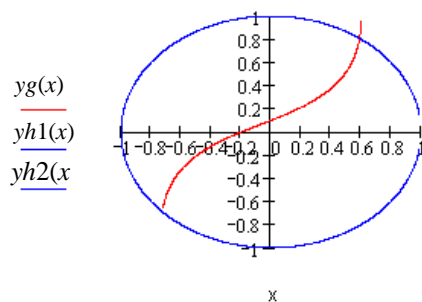


Рис. 4.8. Графики функций $yg(x) = \arcsin(0.1 + 1.5x) - x$,
 $yh1(x) = \sqrt{1 - x^2}$ и $yh2(x) = -\sqrt{1 - x^2}$

```

Minimum(f, xy, grad, k, h, esp) :=
  s ← 0
  a ← f(xy0, xy1)
  for n ∈ 0..k
    d ← grad(f, xy0, xy1)
    for i ∈ 0,1
      | s ← s + (di)2
      | s ← √s
    break if s ≤ esp
    b0 ← f(xy0, xy1)
    for i ∈ 0,1
      xyi ← xyi - di·h
  b1 ← f(xy0, xy1)
  while b0 < b1
    h ← h/2
    for i ∈ 0,1
      xyi ← xyi - di·h
    b1 ← f(xy0, xy1)
  xy

```

Рис. 4.9. Программный модуль для вычисления минимума функции

Новая функция:

$$\Phi(x, y) := f_1(x, y)^2 + f_2(x, y)^2$$

$$\text{grad}(f, x, y) := \begin{pmatrix} \frac{d}{dx} f(x, y) \\ \frac{d}{dy} f(x, y) \end{pmatrix}$$

$$i := 30 \quad X := \begin{pmatrix} x_{20} \\ y_{20} \end{pmatrix}$$

$$q := \text{Minimum}(\Phi, X, \text{grad}, 30, 0.15, \varepsilon)$$

Ответ:

$$q = \begin{pmatrix} 0.590003 \\ 0.807401 \end{pmatrix}$$

Рис. 4.10. Решение системы нелинейных уравнений методом градиента

Контрольные вопросы

1. В чем заключается основная идея метода Ньютона при решении системы нелинейных уравнений?
2. Каким образом можно определить начальное приближение для решения системы нелинейных уравнений: а) методом Ньютона; б) методом градиента?
3. Когда можно закончить итерационный процесс при решении системы нелинейных уравнений: а) методом Ньютона; б) методом градиента?
4. Как строится итерационная последовательность для нахождения решения системы нелинейных уравнений: а) методом Ньютона; б) методом градиента?
5. В чем отличие итерационного процесса метода Ньютона от итерационного процесса методом градиента?
6. Как определяется матрица Якоби?

5. ИНТЕРПОЛИРОВАНИЕ ФУНКЦИЙ

5.1. Интерполяционная формула Лагранжа

Если пары значений $(x_0; y_0), (x_1; y_1), \dots, (x_{n-1}; y_{n-1})$ удовлетворяют заданному уравнению $y = f(x)$, то задача интерполирования заключается в том, чтобы найти приближенное значение y для произвольного x , при этом значение x не выходит за пределы ряда данных значений x_n . Основное применение интерполяции – вычисление значений функции для промежуточных (не узловых) значений аргумента. Для реализации интерполяции надо построить функцию $y = f(x)$, которая должна удовлетворять следующим условиям:

$$y_0 = f(x_0), y_1 = f(x_1), \dots, y_{n-1} = f(x_{n-1}),$$

т. е. интерполяционная функция должна принимать заданные значения y_0, y_1, \dots, y_{n-1} для узловых значений аргумента x_0, x_1, \dots, x_{n-1} . Можно построить множество непрерывных функций, графики которых будут проходить через n заданных точек $(x_0; y_0), (x_1; y_1), \dots, (x_{n-1}; y_{n-1})$, поэтому существуют разные типы интерполяции: полиномиальная, тригонометрическая, экспоненциальная и т. д.

Обычно задача интерполирования формулируется следующим образом: построить многочлен $P(x) = P_n(x)$ степени не выше n , значения которого в точках x_i ($i = 0, 1, 2, \dots, n$) совпадают со значениями заданной функции, т. е. $P(x_i) = y_i$. При этом многочлен $P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ называется интерполяционным, а точки x_i ($i = 0, 1, 2, \dots, n$) – узлами интерполяции. Для определения коэффициентов a_0, a_1, \dots, a_n решают систему уравнений вида

$$a_0 + a_1x_i + a_2x_i^2 + \dots + a_nx_i^n = f(x_i), \quad (5.1)$$

где $i=0, 1, 2, \dots, n$. Решение системы (5.1) можно найти разными способами. Наиболее часто используются запись интерполяционного многочлена в форме Лагранжа и в форме Ньютона.

Так как многочлен $P_n(x)$ содержит $n+1$ параметров, которыми являются коэффициенты, то при его нахождении можно поставить $n+1$ условий. Рассмотрим для простоты многочлен второй степени

$$P(x) = P_2(x) = ax^2 + bx + c. \quad (5.2)$$

Сформулируем три условия. Обычно требуется, чтобы этот многочлен совпадал с функцией $f(x)$ в некоторых трех заданных точках:

$$P(x_1) = f(x_1), \quad P(x_2) = f(x_2), \quad P(x_3) = f(x_3). \quad (5.3)$$

Эти значения считаются известными. Лагранж предложил искать многочлен $P(x)$ в виде

$$P(x) = A(x-x_2)(x-x_3) + B(x-x_1)(x-x_3) + \\ + C(x-x_1)(x-x_2), \quad (5.4)$$

где A, B, C – постоянные. Для выбора постоянных A, B, C воспользуемся условиями (5.3). Заметим, что при подстановке $x=x_1, x_2, x_3$ в правой части формулы (5.4) остается лишь одно слагаемое. Получим

$$f(x_1) = A(x_1-x_2)(x_1-x_3), \quad f(x_2) = B(x_2-x_1)(x_2-x_3), \\ f(x_3) = C(x_3-x_1)(x_3-x_2).$$

Найдя отсюда A, B, C и подставляя их в (5.4), получаем интерполяционную формулу Лагранжа

$$f(x) = P_2(x) = f(x_1) \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + \\ + f(x_2) \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + f(x_3) \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}. \quad (5.5)$$

В общем случае интерполяционный многочлен Лагранжа строится

аналогично:

$$\begin{aligned}
 f(x) = & f(x_0) \frac{(x-x_1)(x-x_2)\mathbf{K}(x-x_n)}{(x_0-x_1)(x_0-x_2)\mathbf{K}(x_0-x_n)} + \\
 & + f(x_1) \frac{(x-x_0)(x-x_2)\mathbf{K}(x-x_n)}{(x_1-x_0)(x_1-x_2)\mathbf{K}(x_1-x_n)} + \mathbf{K} + \\
 & + f(x_k) \frac{(x-x_0)(x-x_1)\mathbf{K}(x-x_{k-1})(x-x_{k+1})\mathbf{K}(x-x_n)}{(x_k-x_0)(x_k-x_1)\mathbf{K}(x_k-x_{k-1})(x_k-x_{k+1})\mathbf{K}(x_k-x_n)} + \\
 & + \mathbf{K} + f(x_n) \frac{(x-x_0)(x-x_1)\mathbf{K}(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\mathbf{K}(x_n-x_{n-1})}, \quad (5.6)
 \end{aligned}$$

или в более краткой записи

$$L_n(x) = \sum_{k=0}^n \frac{\prod_{j \neq k} (x-x_j)}{\prod_{j \neq k} (x_k-x_j)} f(x_k). \quad (5.7)$$

Пример 5.1. Функция задана таблицей. Построить интерполяционный многочлен Лагранжа и найти значение y при $x=0$ (табл. 5.1).

Таблица 5.1

x	-2	1	2
y	25	-8	-15

Решение. Используя формулу (5.6) или (5.7), получим:

$$\begin{aligned}
 L_2(x) = & 25 \frac{(x-1)(x-2)}{(-2-1)(-2-2)} + (-8) \frac{(x+2)(x-2)}{(1+2)(1-2)} + \\
 & + (-15) \frac{(x+2)(x-1)}{(2+2)(2-1)}.
 \end{aligned}$$

Проведя необходимые преобразования, найдем: $L_2(x) = x^2 - 10x + 1$. Отсюда при $x=0$ получим $y=1$.

5.2. Интерполяционная формула Ньютона

Прежде чем перейти к интерполяционной формуле Ньютона, рассмотрим понятие конечной разности – одно из самых важных в прикладной математике. Пусть $y = f(x)$, тогда при заданном h выражение

$$\Delta y = f(x+h) - f(x) \quad (5.8)$$

называется конечной разностью первого порядка функции $f(x)$ с шагом h . Выражение $\frac{1}{h} \Delta y = \frac{f(x+h) - f(x)}{h}$ – первая разделенная разность. Из очевидных свойств разностей отметим, например, следующие:

$$\Delta(y_1 + y_2) = \Delta y_1 + \Delta y_2, \quad \Delta(C y) = C(\Delta y),$$

где $C = \text{const}$. Возьмем вторую разность

$$\begin{aligned} \Delta^2 y &= \Delta(\Delta y) = \Delta(f(x+h) - f(x)) = \\ &= (f(x+2h) - f(x+h)) - (f(x+h) - f(x)) \end{aligned}$$

или, иначе: $\Delta^2 y_0 = \Delta y_1 - \Delta y_0$. Аналогично определим вторую разделенную разность. По такой же схеме можно определить третью разность и третью разделенную разность. Вычисление этих разностей, как правило, удобно, если функция задана табличным способом.

Если расстояние h между соседними значениями x , при которых задается функция $f(x)$, постоянно, то можно воспользоваться более удобной формулой, чем интерполяционный многочлен Лагранжа. Пусть, например, известны значения:

$$f(x_0) = y_0, f(x_1) = y_1, f(x_2) = y_2, f(x_3) = y_3,$$

где $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, $x_3 = x_0 + 3h$. Тогда многочлен $P(x)$, принимающий те же значения при указанных значениях x , будет иметь третью степень. Ньютон предложил искать его в виде

$$P(x) = A + Bq + Cq(q-h) + Dq(q-h)(q-2h), \quad (5.9)$$

где $q = x - x_0$. Согласно условию имеем

$$y_0 = P(x_0) = A, \quad y_1 = P(x_1) = A + Bh, \quad y_2 = P(x_2) = A + B2h + C2h^2,$$

$$y_3 = P(x_3) = A + B3h + C3 \cdot 2h^2 + D3 \cdot 2h^3.$$

Составляя разности для левых и правых частей, получаем

$$\Delta y_0 = Bh, \quad \Delta y_1 = Bh + C2h^2, \quad \Delta y_2 = Bh + C2 \cdot 2h^2 + D3 \cdot 2h^3.$$

Вторично, а затем и в третий раз составляя разности, находим

$$\Delta^2 y_0 = C2h^2, \quad \Delta^2 y_1 = C2h^2 + D3 \cdot 2h^2, \quad \Delta^3 y_0 = D3 \cdot 2h^2.$$

Отсюда находим $A = y_0$, $B = \frac{\Delta y_0}{h}$, $C = \frac{\Delta^2 y_0}{2! h^2}$, $D = \frac{\Delta^3 y_0}{3! h^3}$. Подставляя эти значения в формулу (5.9), получаем формулу Ньютона

$$\begin{aligned} f(x) \approx P(x) = y_0 + \Delta y_0 \frac{q}{h} + \frac{\Delta^2 y_0}{2!} \frac{q}{h} \left(\frac{q}{h} - 1 \right) + \\ + \frac{\Delta^3 y_0}{3!} \frac{q}{h} \left(\frac{q}{h} - 1 \right) \left(\frac{q}{h} - 2 \right), \end{aligned} \quad (5.10)$$

где $q = x - x_0$. Заметив, что если вместо x_0 будем использовать любое табличное значение x_k , получим формулу Ньютона в виде

$$f(x) \approx P(x) = y_k + \Delta y_k \frac{t}{h} + \frac{\Delta^2 y_k}{2!} \frac{t}{h} \left(\frac{t}{h} - 1 \right) + \frac{\Delta^3 y_k}{3!} \frac{t}{h} \left(\frac{t}{h} - 1 \right) \left(\frac{t}{h} - 2 \right),$$

где $t = x - x_k$. Аналогичный вид имеет формула для интерполяционных многочленов других степеней. Для общего случая получаем интерполяционную формулу Ньютона в следующем виде:

$$\begin{aligned} f(x) \approx P(x) = y_0 + s \Delta y_0 + s(s-1) \frac{\Delta^2 y_0}{2!} + K + \\ + (s(s-1)K + (s-n+1)) \frac{\Delta^n y_0}{n!}, \end{aligned} \quad (5.11)$$

где $s = \frac{x - x_0}{h}$ и $\Delta y_0 = y_1 - y_0$, $\Delta^2 y_0 = \Delta y_1 - \Delta y_0$, ... – последовательные конечные разности функции y . При $x = x_i$ ($i = 0, 1, 2, \dots, n$) интерполяционный многочлен Ньютона принимает табличные значения y_i ($i = 0, 1, 2, \dots, n$). Как частные случаи формулы Ньютона получаем при $n=1$ линейное интерполирование, при $n=2$ – квадратичное. Для удобства применения формулы Ньютона рекомендуется предварительно составлять таблицу конечных разностей. Степень интерполяционного многочлена $P(x)$ выбирают, руководствуясь значениями разностей. Например, если третьи разности очень малы, то последний член в формуле (5.10) мал и его можно отбросить, т. е. ограничиться многочленом второй степени. Погрешность формулы (5.11) можно оценить с помощью выражения

$$R_n(x) \approx \frac{\Delta^{n+1} y_0}{(n+1)!} s(s-1) \dots (s-n).$$

Число n следует выбирать так, чтобы разность $\Delta^{n+1} y_0 \approx 0$ в пределах заданной точности.

Пример 5.2. Функция задана таблицей. Построить интерполяционный многочлен Ньютона.

Т а б л и ц а 5.2

x	0	1	3	3
y	1	4	15	40

Решение. Вычислим шаг $h=1$. Далее найдем конечные разности первого порядка:

$$\begin{aligned} \Delta y_0 &= y_1 - y_0 = 4 - 1 = 3, \quad \Delta y_1 = y_2 - y_1 = 15 - 4 = 11, \\ \Delta y_2 &= y_3 - y_2 = 40 - 15 = 25. \end{aligned}$$

Затем определим конечные разности второго и третьего порядков:

$$\begin{aligned} \Delta^2 y_0 &= \Delta y_1 - \Delta y_0 = 11 - 3 = 8, \quad \Delta^2 y_1 = \Delta y_2 - \Delta y_1 = 25 - 11 = 14, \\ \Delta^3 y_0 &= \Delta^2 y_1 - \Delta^2 y_0 = 14 - 8 = 6. \end{aligned}$$

По формуле (5.11) построим интерполяционный многочлен Ньютона:

$$P(x) = 1 + 3x + \frac{x(x-1)}{2!}8 + \frac{x(x-1)(x-2)}{3!}6.$$

Проведя необходимые преобразования, получим $P(x) = 1 + x + x^2 + x^3$.

5.3. Задача обратного интерполирования

Для функции $y = f(x)$, заданной таблично, задача обратного интерполирования состоит в том, чтобы по заданному значению функции y найти соответствующее значение x . Это задача может быть решена, например, с помощью интерполяционного многочлена Лагранжа, если принять переменную y за независимую переменную и переписать формулу (5.7), выразив x как функцию y ;

$$x = \sum_{k=0}^n \frac{\prod_{j \neq k} (y - y_j)}{\prod_{j \neq k} (y_k - y_j)} x_k. \quad (5.12)$$

Аналогично можно получить выражение и с использованием интерполяционной формулы Ньютона. Отметим, что задача обратного интерполирования может быть корректно решена только для взаимно-однозначных функций.

Пример 5.3. Функция задана таблицей. Определить, при каком значении x величина $y = 20$.

Т а б л и ц а 5.3

x	4	6	8
y	11	27	50

Решение. Для решения поставленной задачи обратного интерполирования используем формулу Лагранжа (5.12).

$$x = 4 \frac{(20-27)(20-50)}{(11-27)(11-50)} + 6 \frac{(20-11)(20-50)}{(27-11)(27-50)} + \\ + 8 \frac{(20-11)(20-27)}{(50-11)(50-27)}.$$

Проведя необходимые преобразования, получим: при $x=5.19$ величина $y=20$.

5.4. Численное дифференцирование

Численное дифференцирование производится, как правило, если функция, от которой надо найти производную, задана таблично. Это можно сделать, если заменить рассматриваемую функцию многочленом, от которого и найти производную. Например, из формулы (5.10) получим (это предлагается проверить самостоятельно!)

$$f'(x) \approx \frac{\Delta y_0}{h} + \frac{\Delta^2 y_0}{h} \left(\frac{q}{h} - \frac{1}{2} \right) + \frac{\Delta^3 y_0}{2h} \left[\left(\frac{q}{h} \right)^2 - 2 \left(\frac{q}{h} \right) + \frac{2}{3} \right], \quad (5.13)$$

где $q = x - x_0$. Взяв в формуле (5.13) больше членов, можно получить более точный результат. В частности, полагая, что производная функции вычисляется в точке x_0 , т. е. $x = x_0$ и $q = 0$, получим

$$f'(x_0) \approx \frac{1}{h} \left(\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} \right).$$

Более точная формула имеет вид бесконечного ряда

$$f'(x_0) \approx \frac{1}{h} \left(\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + K \right).$$

Подобным образом можно получить формулы для производных второго и последующих порядков. Если таблица функции получилась в результате эксперимента, то небольшая ошибка в значении функции после деления на малый шаг может привести к большой ошибке в значении производной. То же наблюдается и при вычислении производ-

ных высших порядков. Поэтому желательно, чтобы шаг таблицы был, по крайней мере, на порядок больше, чем возможная ошибка в значении функции, а для вычисления производной второго порядка шаг должен быть на два порядка больше этой ошибки.

Пример 5.4. Пусть задана функция $f(x) = \ln\left(\frac{x}{1+x}\right)$ на интервале $[a, b]$, где $a=1$, $b=5$, $n=16$, $h=(b-a)/n$. Выполнить численное дифференцирование.

Решение. Программный модуль для нахождения конечных разностей на основе интерполяционной формулы Ньютона представлен на рис. 5.1.

```
Delta(a,f,h,n) :=
  for i ∈ 0..n
    | x ← a + h · i
    | Pi ← f(x)
  for i ∈ 0..n-1
    Q0,i ← Pi+1 - Pi
  for j ∈ 1..n-1
    | k ← n-1-j
    | for i ∈ 0..n-1
      | Qj,i ← 0
    | for i ∈ 0..k
      | Qj,i ← Qj-1,i+1 - Qj-1,i
  Q ← QT
  Q
```

Рис. 5.1. Функция для нахождения конечных разностей

Полученные значения приведены на рис. 5.2.

Программный модуль для вычисления первой производной представлен на рис. 5.3.

$D := \text{Delta}(a, f, h, n)$

	0	1	2	3	4	5
0	0.1054	-0.0284	0.0103	-4.4789·10 ⁻³	2.2196·10 ⁻³	-1.2093·10 ⁻³
1	0.077	-0.0181	5.8001·10 ⁻³	-2.2593·10 ⁻³	1.0104·10 ⁻³	-5.0083·10 ⁻⁴
2	0.0588	-0.0123	3.5408·10 ⁻³	-1.2489·10 ⁻³	5.0954·10 ⁻⁴	-2.3195·10 ⁻⁴
3	0.0465	-8.7797·10 ⁻³	2.2919·10 ⁻³	-7.3935·10 ⁻⁴	2.7759·10 ⁻⁴	-1.1692·10 ⁻⁴
4	0.0377	-6.4878·10 ⁻³	1.5525·10 ⁻³	-4.6177·10 ⁻⁴	1.6067·10 ⁻⁴	-6.3006·10 ⁻⁵
5	0.0313	-4.9352·10 ⁻³	1.0908·10 ⁻³	-3.0111·10 ⁻⁴	9.766·10 ⁻⁵	-3.5843·10 ⁻⁵
6	0.0263	-3.8445·10 ⁻³	7.8968·10 ⁻⁴	-2.0344·10 ⁻⁴	6.1817·10 ⁻⁵	-2.1329·10 ⁻⁵
7	0.0225	-3.0548·10 ⁻³	5.8624·10 ⁻⁴	-1.4162·10 ⁻⁴	4.0488·10 ⁻⁵	-1.3184·10 ⁻⁵
8	0.0194	-2.4685·10 ⁻³	4.4462·10 ⁻⁴	-1.0113·10 ⁻⁴	2.7305·10 ⁻⁵	-8.419·10 ⁻⁶
9	0.0169	-2.0239·10 ⁻³	3.4348·10 ⁻⁴	-7.383·10 ⁻⁵	1.8886·10 ⁻⁵	-5.5306·10 ⁻⁶
10	0.0149	-1.6804·10 ⁻³	2.6965·10 ⁻⁴	-5.4945·10 ⁻⁵	1.3355·10 ⁻⁵	-3.7244·10 ⁻⁶
11	0.0132	-1.4108·10 ⁻³	2.1471·10 ⁻⁴	-4.159·10 ⁻⁵	9.6306·10 ⁻⁶	0
12	0.0118	-1.1961·10 ⁻³	1.7312·10 ⁻⁴	-3.1959·10 ⁻⁵	0	0
13	0.0106	-1.0229·10 ⁻³	1.4116·10 ⁻⁴	0	0	0
14	9.6155·10 ⁻³	-8.8178·10 ⁻⁴	0	0	0	0
15	8.7337·10 ⁻³	0	0	0	0	0

Рис. 5.2. Конечные разности

```

dly(a, h, n) :=
  k ← 0
  for j ∈ 0..n-1
    yj,1 ← a + h · j
  for j ∈ 0.. $\frac{n-1}{2}$ 
     $y^1_{j,0} \leftarrow \frac{1}{h} \cdot \sum_{i=0}^{n-1} D_{j,i} \cdot dq(j, i+1)$ 
     $y^1_{j,1} \leftarrow \frac{1}{h} \cdot \sum_{i=0}^{n-1} D_{j,i} \cdot dq(j+1, i+1)$ 
  i ← 0
  j ← 0
  while i < n
    yi,0 ← y1j,0
    yi+1,0 ← y1j,1
    j ← j + 1
    i ← i + 2
  y

```

Рис. 5.3. Функция для вычисления первой производной

Полученные значения приведены на рис. 5.4.

$$G := dly(a, h, n)$$

$G =$

	0	1
0	0.5	1
1	0.3556	1.25
2	0.2667	1.5
3	0.2078	1.75
4	0.1667	2
5	0.1368	2.25
6	0.1143	2.5
7	0.097	2.75
8	0.0833	3
9	0.0724	3.25
10	0.0635	3.5
11	0.0561	3.75
12	0.05	4
13	0.0448	4.25
14	0.0404	4.5

Рис. 5.4. Значения первой производной

Контрольные вопросы

1. В чем заключается задача интерполирования?
2. Запишите интерполяционные формулы Лагранжа и Ньютона.
3. Дайте определение конечных разностей и разделенных разностей.
4. В каких случаях применяется интерполяционная формула Ньютона?
5. Как степень интерполяционного многочлена связана с количеством узлов интерполяции?
6. Как строятся интерполяционные многочлены Лагранжа и Ньютона?

7. В чем особенность приближения таблично заданной функции методом интерполирования?
8. Как используется метод интерполирования для уточнения таблиц функций?
9. Что понимается под обратным интерполированием?
10. Каким образом можно оценить погрешность интерполяционной формулы Ньютона?

6. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

6.1. Метод Рунге–Кутты

Будем рассматривать численные методы решения задачи Коши. Пусть на отрезке $x_0 \leq x \leq b$ требуется найти решение дифференциального уравнения

$$y' = f(x, y), \quad (6.1)$$

удовлетворяющее начальному условию

$$y(x_0) = y_0. \quad (6.2)$$

Предполагается, что условия существования и единственности решения задачи Коши выполнены. На практике найти решение задачи Коши удастся не всегда, поэтому используют приближенные методы решения. Отрезок $[x_0, b]$ разбивают на интервалы с шагом h (как правило, постоянным), полагают $h = x_{n+1} - x_n$ и находят значение $y_{n+1} = y(x_{n+1})$. При этом получают таблицу, состоящую из вектора аргументов $x = (x_0, x_1, \dots, x_n)$ и соответствующего ему вектора функции $y = (y_0, y_1, \dots, y_n)$.

Из общего курса обыкновенных дифференциальных уравнений известен аналитический метод, основанный на использовании ряда Тейлора. При этом приближенное решение исходной задачи ищут в виде

$$y_m(x_{n+1}) \approx \sum_{i=0}^m \frac{h^i}{i!} y^{(i)}(x_n), \quad (6.3)$$

где $x_{n+1} = x_n + h$, $y^{(0)}(x_0) = y(x_0)$, $y^{(1)}(x_0) = y'(x_0) = f(x_0, y_0)$, а значения $y^{(i)}(x_0)$ при $i = 2, 3, \dots, m$ находят по формулам, полученным последовательным дифференцированием уравнения (6.1). В случае $m = 1$ при-

ближенное равенство (6.3) не требует вычисления производных правой части уравнения и позволяет с погрешностью порядка h^2 находить значение $y(x_n)$. Соответствующая формула имеет вид

$$y_{n+1} = y_n + hf'_n \quad (6.4)$$

и носит название формулы Эйлера.

Ниже приводятся формулы Рунге–Кутты для решения уравнения (6.1):

$$y_{i+1} = y_i + \Delta y_i, \quad (6.5)$$

$$\text{где } \Delta y_i = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6},$$

$$k_1 = hf(x_i, y_i), \quad k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right), \quad k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right),$$

$$k_4 = hf(x_i + h, y_i + k_3), \quad i = 0, 1, 2, \dots, m.$$

Метод Рунге–Кутты имеет порядок точности h^4 . Приближенно оценку погрешности метода Рунге–Кутты можно вычислить по формуле

$$\varepsilon = \frac{|y_{2h} - y_h|}{15}, \quad (6.6)$$

где y_{2h} и y_h – результаты вычислений по схеме (6.5) с шагом h и шагом $2h$. Метод Рунге–Кутты применим также для решения системы дифференциальных уравнений вида

$$\begin{cases} y' = f(x, y, z), \\ z' = \phi(x, y, z) \end{cases}$$

с заданными начальными условиями: $y = y_0, z = z_0$ при $x = x_0$.

Одношаговые методы решения задачи Коши имеют один существенный недостаток, заключающийся в том, что при построении этих методов требуется информация о решаемой задаче на отрезке длиной в один шаг. На каждом этапе процесса эта информация должна обновляться, что усложняет вычисление. Можно построить вычислительные методы таким образом, чтобы информация о решаемой задаче могла быть использована на нескольких шагах вычислительного процесса. Такие методы получили название многошаговых.

6.2. Метод Милна

Для решения задачи (6.1) по методу Милна из начальных условий $y = y_0$ при $x = x_0$ находим каким-либо способом последовательные значения

$$y_1 = y(x_1), y_2 = y(x_2), y_3 = y(x_3)$$

исходной функции $y(x)$ (например, применяем метод Рунге–Кутты). Приближения y_i и \bar{y}_i для следующих значений y_i ($i = 4, 5, 6, K, n$) последовательно находятся по формулам:

$$y_i = y_{i-2} + \frac{h}{3}(\bar{f}_i + 4f_{i-1} + f_{i-2}), \quad (6.7)$$

$$\text{где } \bar{y}_i = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1}),$$

$$f_i = f(x_i, y_i), \quad \bar{f}_i = f(x_i, \bar{y}_i).$$

Полагая в формуле (6.7) $i = 3$, находим последовательно $\bar{y}_4, \bar{f}_4, y_4$. Затем, полагая $i = 4$, находим $\bar{y}_5, \bar{f}_5, y_5$ и т. д. Найденные значения y_4, y_5, y_6, \dots и являются приближенными значениями решения $y(x)$ при $x = x_4, x_5, x_6, \dots$, где $x_i = x_0 + ih$. Погрешность при вычислении y_i по данному методу можно определить по формуле

$$\varepsilon_i = \frac{1}{29} |\bar{y}_i - y_i|.$$

Поэтому при вычисления можно попутно проверять, не выходит ли эта погрешность за пределы принятой степени точности вычислений. Если это где-либо произойдет, надо уменьшить шаг. Отметим, что суммарная ошибка данного метода имеет порядок h^4 .

6.3. Метод Адамса

Для решения задачи (6.1) по методу Адамса исходя из начальных условий $y = y_0$ при $x = x_0$ находим каким-либо способом следующие три значения искомой функции $y(x)$:

$$y_1 = y(x_1) = y(x_0 + h), \quad y_2 = y(x_2) = y(x_0 + 2h), \quad y_3 = y(x_3) = y(x_0 + 3h).$$

Эти три значения можно найти, применяя, например, метод Рунге–Кутты. С помощью значений x_0, x_1, x_2, x_3 и y_0, y_1, y_2, y_3 вычисляем величины q_0, q_1, q_2, q_3 , где

$$q_0 = hy'_0 = hf(x_0, y_0), \quad q_1 = hy'_1 = hf(x_1, y_1),$$

$$q_2 = hy'_2 = hf(x_2, y_2), \quad q_3 = hy'_3 = hf(x_3, y_3).$$

Метод Адамса заключается в вычислении разностей с помощью формулы

$$\Delta y_i = q_i + \frac{1}{2} \Delta q_{i-1} + \frac{5}{12} \Delta^2 q_{i-2} + \frac{3}{8} \Delta^3 q_{i-3}, \quad (6.8)$$

где $\Delta q_{i-1} = q_i - q_{i-1} = h(f(x_i, y_i) - f(x_{i-1}, y_{i-1})),$

$$\Delta^2 q_{i-2} = \Delta q_{i-1} - \Delta q_{i-2} = \Delta q_{i-1} - (q_{i-1} - q_{i-2}) =$$

$$= \Delta q_{i-1} - h(f(x_{i-1}, y_{i-1}) - f(x_{i-2}, y_{i-2})),$$

$$\Delta^3 q_{i-3} = \Delta^2 q_{i-2} - \Delta^2 q_{i-3} =$$

$$= h[f(x_i, y_i) - 3f(x_{i-1}, y_{i-1}) + 3f(x_{i-2}, y_{i-2}) - f(x_{i-3}, y_{i-3})].$$

Таким образом, можно записать следующую рекуррентную формулу для нахождения конечных разностей j -го порядка:

$$\Delta^j q_i = \Delta^{j-1} q_{i+1} - \Delta^{j-1} q_i.$$

Используя полученные значения $q_3, \Delta q_2, \Delta^2 q_1, \Delta^3 q_0$, с помощью выражения (6.8), полагая в нем $i=3$, вычисляем Δy_3 по формуле

$$\Delta y_3 = q_3 + \frac{1}{2} \Delta q_2 + \frac{5}{12} \Delta^2 q_1 + \frac{3}{8} \Delta^3 q_0.$$

Найдя значение Δy_3 , определяем $y_4 = y_3 + \Delta y_3$. Зная x_4 и y_4 , далее вычисляем $q_4 = hf(x_4, y_4)$ и конечные разности $\Delta q_3, \Delta^2 q_2, \Delta^3 q_1$. Эта процедура повторяется для вычисления последующих значений $\Delta y_4, y_5, q_5, \Delta q_4, \Delta^2 q_3, \Delta^3 q_2$. Формула Адамса исходит из предположения, что третьи конечные разности $\Delta^3 q_i$ являются постоянными. На практике за значениями третьих конечных разностей следят, выбирая

шаг h столь малым, чтобы соседние разности $\Delta^3 q_i$ и $\Delta^3 q_{i+1}$ отличались между собой незначительно (с заданной точностью). Для повышения точности результата формула Адамса может быть пополнена членами, содержащими четвертые и высшие разности величины q . При этом возрастает число первых значений функции y , которые требуются на начальном этапе вычислений. Формула (6.8) носит название экстраполяции формулы Адамса. Метод Адамса позволяет получить более точные результаты, чем метод Эйлера и его уточнения.

Формулы Адамса–Крылова для решения уравнения (6.1) имеют следующий вид:

$$\begin{cases} \Delta y_0 = q_0 + \frac{1}{2} \Delta q_0 - \frac{1}{12} \Delta^2 q_0 + \frac{1}{24} \Delta^3 q_0, \\ \Delta y_1 = q_1 + \frac{1}{2} \Delta q_0 + \frac{5}{12} \Delta^2 q_0 - \frac{1}{24} \Delta^3 q_0, \\ \Delta y_2 = q_2 - \frac{1}{2} \Delta q_1 + \frac{5}{12} \Delta^2 q_0 + \frac{3}{8} \Delta^3 q_0. \end{cases}$$

Пример 6.1. Решить систему обыкновенных дифференциальных уравнений на интервале $[a, b]$:

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2), \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2). \end{cases}$$

где $f_1 = \frac{x}{\sqrt{1+x^2+y_2^2}}$, $f_2 = \frac{y_2}{\sqrt{1+x^2+y_1^2}}$, $y_1(a) = 0.2$, $y_2(a) = 0.1$.

Решение. Границы отрезка $[a, b]$, на котором ищется решение: $a = -1$, $b = 1$. Зададим шаг $h = 0.1$. Функция определения длины отрезка $[a, b]$ приведен на рис. 6.1.

$$\text{otrz}(a, b) := \begin{cases} g \leftarrow b - a & \text{if } a \geq 0 \wedge b \geq 0 \\ g \leftarrow |a| + b & \text{if } a \leq 0 \wedge b \geq 0 \\ g \leftarrow |a| - |b| & \text{otherwise} \\ g \end{cases}$$

Рис. 6.1. Функция определения длины отрезка

Функция формирования множества экстраполяционных точек на отрезке $[a, b]$ дана на рис. 6.2.

$$\text{Adams}(a, b, y, h, n) := \begin{cases} d \leftarrow \text{rkfixed}\left(y, a, b, \frac{\text{otrz}(a, b)}{h}, f\right) \\ w \leftarrow \begin{pmatrix} d_{2,2} & d_{2,3} & d_{2,1} \\ d_{3,2} & d_{3,3} & d_{3,1} \\ d_{4,2} & d_{4,3} & d_{4,1} \end{pmatrix} \\ as \leftarrow \begin{pmatrix} y_1 & y_2 & a \end{pmatrix} \\ a \leftarrow 3 \cdot h + a \\ \text{for } i \in 4..3 + \frac{\text{otrz}(a, b)}{h} \\ \quad \left| \begin{array}{l} fl \leftarrow f \left[w_{i-1,3}, \begin{pmatrix} w_{i-1,1} \\ w_{i-1,2} \end{pmatrix} \right] \\ f2 \leftarrow f \left[w_{i-2,3}, \begin{pmatrix} w_{i-2,1} \\ w_{i-2,2} \end{pmatrix} \right] \\ f3 \leftarrow f \left[w_{i-3,3}, \begin{pmatrix} w_{i-3,1} \\ w_{i-3,2} \end{pmatrix} \right] \\ \text{for } j \in 1..n \\ \quad w_{i,j} \leftarrow w_{i-1,j} + \frac{h}{12} \cdot (23 fl_j - 16 f2_j + 5 f3_j) \\ a \leftarrow a + h \\ w_{i,3} \leftarrow a \end{array} \right. \\ w \leftarrow \text{augment}(as^T, w^T) \\ w^T \end{cases}$$

Рис. 6.2. Метод Адамса

Для расчета необходимо определить каким-либо способом три приближения искомой функции, задающие «начальный отрезок». Находить эти приближения будем по методу Рунге–Кутты, используя встроенную функцию MathCad **rkfixed**(*y*, *x1*, *x2*, **npoints**, **D**), где *y* –

заданные значения функции (начальные условия), $x1$ – начальное значение интервала, $x2$ – конечное значение интервала, **npoints** – количество рядов в выходной матрице; **D** – обыкновенное дифференциальное уравнение.

Для проверки применим еще функцию, реализующую менее точный метод Эйлера (рис. 6.3).

```

Euler(a,b,y,h,n) :=
  as ← (y1 y2 a)
  for i ∈ 1..(otrz(a,b)/h)
    f ← f(a,y)
    for j ∈ 1..n
      yj ← yj + h·fj
      di,j ← yj
    a ← a + h
    di,3 ← a
  d ← augment(asT, dT)
  dT

```

Рис. 6.3. Метод Эйлера

Сравним значения, полученные по методам Адамса, Рунге–Кутты и Эйлера (рис. 6.4–6.6).

	1	2	3
1	0.2	0.1	-1
2	0.1313426043	0.1074643471	-0.9
3	0.0668380291	0.1159476647	-0.8
4	7.1462829848·10 ⁻³	0.1255997302	-0.7
5	-0.047021729	0.1365778989	-0.6
6	-0.094826884	0.1490548654	-0.5
7	-0.1354260774	0.1632008357	-0.4
8	-0.1679921655	0.1791804071	-0.3
9	-0.1917841661	0.1971463702	-0.2
10	-0.2062287925	0.2172324347	-0.1
11	-0.2109966467	0.2395448561	0
12	-0.2060519328	0.2641523744	0.1
13	-0.1916592898	0.2910750504	0.2
14	-0.1683451873	0.3202744391	0.3
15	-0.1368268356	0.3516487207	0.4
16	-0.0979296439	0.3850357434	0.5
17	-0.0525120071	0.4202244385	0.6
18	-1.4079747253·10 ⁻³	0.4569721416	0.7
19	0.0546098062	0.4950236824	0.8
20	0.1148495275	0.5341283759	0.9
21	0.1787066007	0.5740526269	1

Adam =

Рис.6.4. Результаты, полученные по методу Адамса

	1	2	3
1	-1	0.2	0.1
2	-0.9	0.1313426043	0.1074643471
3	-0.8	0.0668380291	0.1159476647
4	-0.7	7.1462829848·10 ⁻³	0.1255997302
5	-0.6	-0.0469935881	0.1365814985
6	-0.5	-0.0947784348	0.1490608458
7	-0.4	-0.1353718386	0.1632070196
8	-0.3	-0.1679540693	0.1791842397
9	-0.2	-0.1917902667	0.1971448044
10	-0.1	-0.2063090314	0.2172216501
11	0	-0.2111760146	0.2395198995
12	0.1	-0.206343235	0.2641070656
13	0.2	-0.1920588402	0.291002554
14	0.3	-0.1688337441	0.3201685832
15	0.4	-0.1373753014	0.3515056169
16	0.5	-0.0985066404	0.3848549392
17	0.6	-0.0530901435	0.4200090083
18	0.7	-1.9671613018·10 ⁻³	0.4567276908
19	0.8	0.0540818707	0.4947567864
20	0.9	0.1143586185	0.5338451307
21	1	0.1782538622	0.57375775

Rgka =

Рис. 6.5. Результаты, полученные по методу Рунге–Кутты

	1	2	3
1	0.2	0.1	-1
2	0.1294654384	0.1070014004	-0.9
3	0.0627795462	0.1149181801	-0.8
4	5.6005209378·10 ⁻⁴	0.123881019	-0.7
5	-0.0564931206	0.1340297465	-0.6
6	-0.1076062321	0.145509241	-0.5
7	-0.1519535832	0.1584641192	-0.4
8	-0.1886970812	0.1730328617	-0.3
9	-0.2170451714	0.1893421601	-0.2
10	-0.2363272585	0.2075019517	-0.1
11	-0.2460720836	0.227600932	0
12	-0.2460720836	0.2497017404	0.1
13	-0.2364153216	0.2738351005	0.2
14	-0.2174746509	0.2999931404	0.3
15	-0.1898573642	0.3281234632	0.4
16	-0.1543304293	0.3581263732	0.5
17	-0.1117406593	0.3898572909	0.6
18	-0.0629454734	0.4231348684	0.7
19	-8.7623218629·10 ⁻³	0.4577534301	0.8
20	0.0500621903	0.493497131	0.9
21	0.1128667228	0.5301531147	1

Рис. 6.6. Результаты, полученные по методу Эйлера

Результаты сравнения методов для функции f_1 представлены на рис. 6.7, где значения функции: $A1$ – по методу Адамса; $B1$ – по методу Рунге–Кутты; $E1$ – по методу Эйлера.

На рис. 6.8 представлены результаты сравнения методов для функции f_2 , где значения функции: $A2$ – по методу Адамса; $B2$ – по методу Рунге–Кутты; $E2$ – по методу Эйлера.

Поскольку метод Рунге–Кутты дает более точные результаты, то именно он был применен в расчетной схеме метода Адамса для определения начальных приближений. На рис. 6.9 графически представлены результаты сравнения для функции f_1 ; а на рис. 6.10 – для функции f_2 (через Ha обозначена матрица шагов).

$\text{stav}(A1, B1) =$		1	$\text{stav}(A1, E1) =$		1
	1	0		1	0
	2	0		2	$1.877 \cdot 10^{-3}$
	3	0		3	$4.058 \cdot 10^{-3}$
	4	0		4	$6.586 \cdot 10^{-3}$
	5	$-2.814 \cdot 10^{-5}$		5	$9.471 \cdot 10^{-3}$
	6	$-4.845 \cdot 10^{-5}$		6	0.013
	7	$-5.424 \cdot 10^{-5}$		7	0.017
	8	$-3.81 \cdot 10^{-5}$		8	0.021
	9	$6.101 \cdot 10^{-6}$		9	0.025
	10	$8.024 \cdot 10^{-5}$		10	0.03
	11	$1.794 \cdot 10^{-4}$		11	0.035
	12	$2.913 \cdot 10^{-4}$		12	0.04
	13	$3.996 \cdot 10^{-4}$		13	0.045
	14	$4.886 \cdot 10^{-4}$		14	0.049
	15	$5.485 \cdot 10^{-4}$		15	0.053
	16	$5.77 \cdot 10^{-4}$		16	0.056
	17	$5.781 \cdot 10^{-4}$		17	0.059
	18	$5.592 \cdot 10^{-4}$		18	0.062
	19	$5.279 \cdot 10^{-4}$		19	0.063
	20	$4.909 \cdot 10^{-4}$		20	0.065

Рис. 6.7. Результаты сравнения для функции f_1

$\text{stav}(A2, E2) =$		1	$\text{stav}(A2, E2) =$		1
	1	0		1	0
	2	0		2	$4.629 \cdot 10^{-4}$
	3	0		3	$1.029 \cdot 10^{-3}$
	4	0		4	$1.719 \cdot 10^{-3}$
	5	$-3.6 \cdot 10^{-6}$		5	$2.548 \cdot 10^{-3}$
	6	$-5.98 \cdot 10^{-6}$		6	$3.546 \cdot 10^{-3}$
	7	$-6.184 \cdot 10^{-6}$		7	$4.737 \cdot 10^{-3}$
	8	$-3.833 \cdot 10^{-6}$		8	$6.148 \cdot 10^{-3}$
	9	$1.566 \cdot 10^{-6}$		9	$7.804 \cdot 10^{-3}$
	10	$1.078 \cdot 10^{-5}$		10	$9.73 \cdot 10^{-3}$
	11	$2.496 \cdot 10^{-5}$		11	0.012
	12	$4.531 \cdot 10^{-5}$		12	0.014
	13	$7.25 \cdot 10^{-5}$		13	0.017
	14	$1.059 \cdot 10^{-4}$		14	0.02
	15	$1.431 \cdot 10^{-4}$		15	0.024
	16	$1.808 \cdot 10^{-4}$		16	0.027
	17	$2.154 \cdot 10^{-4}$		17	0.03
	18	$2.445 \cdot 10^{-4}$		18	0.034
	19	$2.669 \cdot 10^{-4}$		19	0.037
	20	$2.832 \cdot 10^{-4}$		20	0.041

Рис. 6.8. Результаты сравнения для функции f_2

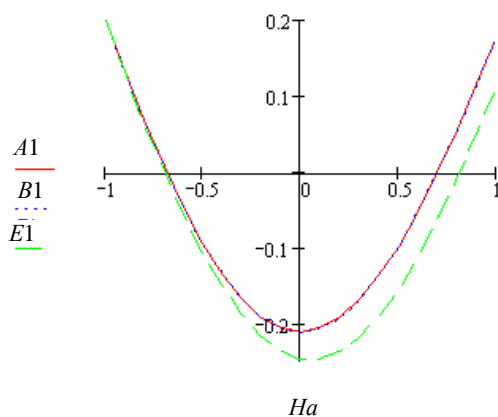


Рис. 6.9. Графическое сравнение результатов для функции f_1

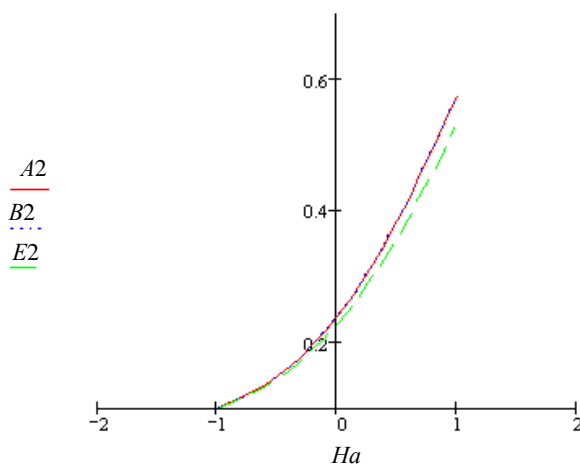


Рис. 6.10. Графическое сравнение результатов для функции f_2

Исследуем влияние шага на точность решения. Зададим шаг, в 10 раз меньший. Сравним значения, полученные по методам Адамса Рунге–Кутты и Эйлера (рис. 6.11–6.13). Выведем первые десять значений – этого будет достаточно, чтобы судить о влиянии шага.

	1	2	3
1	0.2	0.1	-1
2	0.1929644778	0.1007045706	-0.99
3	0.1859651848	0.1014180887	-0.98
4	0.1790026695	0.1021406851	-0.97
Adam = 5	0.172077485	0.1028724914	-0.96
6	0.1651901991	0.1036136421	-0.95
7	0.1583413843	0.1043642723	-0.94
8	0.1515316217	0.1051245189	-0.93
9	0.1447615005	0.10589452	-0.92
10	0.1380316184	0.1066744155	-0.91

Рис. 6.11. Значения, полученные по методу Адамса

	1	2	3
1	-1	0.2	0.1
2	-0.99	0.1929644778	0.1007045706
3	-0.98	0.1859651848	0.1014180887
4	-0.97	0.1790026695	0.1021406851
Rgka = 5	-0.96	0.1720774881	0.102872492
6	-0.95	0.1651902052	0.1036136432
7	-0.94	0.1583413935	0.104364274
8	-0.93	0.151531634	0.1051245212
9	-0.92	0.1447615159	0.1058945229
10	-0.91	0.1380316369	0.106674419

Рис. 6.12. Значения, полученные по методу Рунге–Кутты

	1	2	3
1	0.2	0.1	-1
2	0.1929465438	0.10070014	-0.99
3	0.185929044	0.1014091328	-0.98
4	0.1789480446	0.1021271071	-0.97
Eile = 5	0.1720040979	0.1028541935	-0.96
6	0.1650977642	0.1035905239	-0.95
7	0.1582296122	0.1043362317	-0.94
8	0.1514002185	0.1050914518	-0.93
9	0.1446101682	0.1058563205	-0.92
10	0.1378600547	0.1066309755	-0.91

Рис. 6.13. Значения, полученные по методу Эйлера

Результаты сравнения методов для функции f_1 с шагом $h=0.01$ представлены на рис. 6.14.

$srav(A1,B1) =$		1	$srav(A1,E1) =$		1
	1	0		1	0
	2	0		2	$1.793 \cdot 10^{-5}$
	3	0		3	$3.614 \cdot 10^{-5}$
	4	0		4	$5.462 \cdot 10^{-5}$
	5	$-3.048 \cdot 10^{-9}$		5	$7.339 \cdot 10^{-5}$
	6	$-6.116 \cdot 10^{-9}$		6	$9.243 \cdot 10^{-5}$
	7	$-9.2 \cdot 10^{-9}$		7	$1.118 \cdot 10^{-4}$
	8	$-1.23 \cdot 10^{-8}$		8	$1.314 \cdot 10^{-4}$
	9	$-1.541 \cdot 10^{-8}$		9	$1.513 \cdot 10^{-4}$
	10	$-1.854 \cdot 10^{-8}$		10	$1.716 \cdot 10^{-4}$

Рис 6.14. Результаты сравнения для функции f_1 с шагом $h=0.01$

Результаты сравнения методов для функции f_2 с шагом $h=0.01$ представлены на рис. 6.15.

$srav(A2,B2) =$		1	$srav(A2,E2) =$		1
	1	0		1	0
	2	0		2	$4.431 \cdot 10^{-6}$
	3	0		3	$8.956 \cdot 10^{-6}$
	4	0		4	$1.358 \cdot 10^{-5}$
	5	$-5.692 \cdot 10^{-10}$		5	$1.83 \cdot 10^{-5}$
	6	$-1.142 \cdot 10^{-9}$		6	$2.312 \cdot 10^{-5}$
	7	$-1.714 \cdot 10^{-9}$		7	$2.804 \cdot 10^{-5}$
	8	$-2.286 \cdot 10^{-9}$		8	$3.307 \cdot 10^{-5}$
	9	$-2.856 \cdot 10^{-9}$		9	$3.82 \cdot 10^{-5}$
	10	$-3.425 \cdot 10^{-9}$		10	$4.344 \cdot 10^{-5}$

Рис. 6.15. Результаты сравнения для функции f_2 с шагом $h=0.01$

На рис. 6.16 графически представлены результаты сравнения для функции f_1 , а на рис. 6.17 – для функции f_2 .

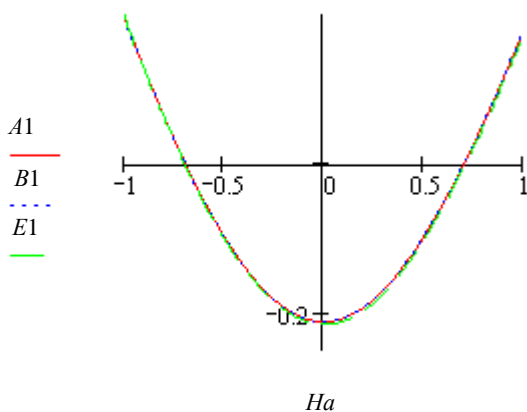


Рис. 6.16. Графическое сравнение результатов для функции f_1 с шагом $h = 0.01$

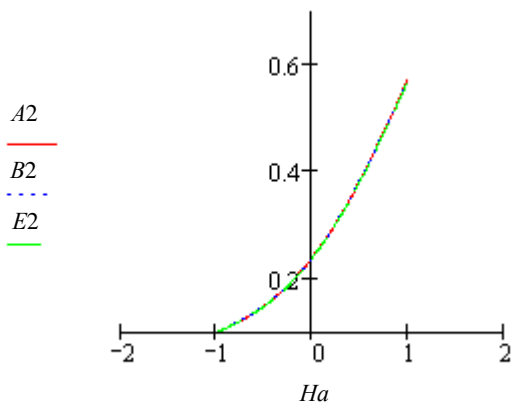


Рис. 6.17. Графическое сравнение результатов для функции f_2 с шагом $h = 0.01$

Из графика и расчетов видно, что при более частом шаге решение системы обыкновенных дифференциальных уравнений становится намного точнее.

Контрольные вопросы

1. Сформулируйте задачу Коши для обыкновенных дифференциальных уравнений первого порядка.
2. Какие методы решения дифференциального уравнения называются многошаговыми? Привести примеры.
3. Как оценить погрешность применяемого метода?
4. Перечислите достоинства и недостатки одношаговых и многошаговых методов решения дифференциальных уравнений.
5. В чем заключается геометрический смысл решения дифференциального уравнения методом Эйлера?
6. Каким образом можно получить решение дифференциального уравнения по методу Рунге–Кутты?
7. В каких случаях применяют экстраполяционную формулу Адамса?
8. Как найти начальные приближения в случае применения метода Милна?

Литература

1. *Воеводин В.В.* Вычислительные основы линейной алгебры. – М.: Наука, 1977.
2. *Волков Е.В.* Численные методы. – М.: Наука, 1987.
3. *Демидович Б.П.* Основы вычислительной математики. – СПб.: Лань, 2007.
4. *Копченкова Н.В., Марон И.А.* Вычислительная математика в примерах и задачах. – М.: Наука, 1972.
5. *Рябенский В.С.* Введение в вычислительную математику. – М.: Физматлит, 2000.
6. *Вержбицкий В.М.* Основы численных методов. – М.: Высш. шк., 2005.
7. *Бахвалов Н.С., Жидков Н.П., Кобельков Г.М.* Численные методы. – М.: Наука, 1987.
8. *Калиткин Н.Н.* Численные методы. – М.: Наука, 1978.
9. *Марчук Г.И.* Методы вычислительной математики. – М.: Наука, 1989.
10. *Самарский А.А., Гулин А.В.* Численные методы: учеб. пособие для вузов. – М.: Наука, 1989.
11. *Демидович Б.П., Марон И.А.* Основы вычислительной математики. – М.: Наука, – 1970.
12. *Исаков В.Н.* Элементы численных методов. – М.: Academia, 2003.
13. *Киреев В.И.* Численные методы в примерах и задачах. – М.: Высш. шк., 2004.
14. *Самарский А.А.* Введение в численные методы. – М.: Наука. – 1997.
15. *Турчак Л.И.* Основы численных методов. – М.: Физматлит, 2005.
16. Описание системы Mathcad 6.0+. – М.: Изд. дом «Филинь», 1996.
17. *Гурский Д., Турбина Е.* Вычисления в MathCad 12. – СПб.: Питер, 2006.
18. *Плис А.И., Сливина Н.А.* Лабораторный практикум по высшей математике: учеб. пособ. для втузов. – М.: Высш. шк., 1994.
19. *Икрамов Х.Д.* Численные методы линейной алгебры. – М.: Знание, 1989. – №2.
20. *Беланов А.А.* Решение алгебраических уравнений методом Лобачевского. – М.: Наука, 1989.

21. *Корн Г.А., Корн Т.М.* Справочник по математике для научных работников и инженеров. – М.: Наука, 1984.
22. *Гельфанд А.О.* Исчисление конечных разностей: учеб. пособие для университетов. – М.: КомКнига, 2006.
23. *Кацман Ю.Я.* Прикладная математика. Численные методы: учеб. пособие. – Томск: Изд. ТПУ, 2000.
24. *Монастырский П.И., Крылов В.И. и др.* Вычислительные методы высшей математики: учеб. пособие. – Минск, 1975.
25. *Лотов А.В.* Введение в экономико-математическое моделирование. – М.: Наука, 1984.
26. *Каганов В.И.* Компьютерные вычисления в средах Excel и Mathcad. – М.: Горячая линия – Телеком, 2003.
27. *Кирьянов Д.В.* Самоучитель Mathcad 12. – СПб.: БХВ-Петербург, 2004.
28. *Черняк А.А., Новиков В.А., Мельников О.И. и др.* Математика для экономистов на базе Mathcad. – СПб.: БХВ-Петербург, 2003.

Оглавление

Предисловие	3
1. Язык программирования пакета MathCad	4
1.1. Панели инструментов пакета MathCad	4
1.2. Создание строки программного кода	7
1.3. Локальное присваивание	8
1.4. Условный оператор	9
1.5. Функции пользователя	10
1.6. Операторы цикла	12
1.7. Оператор return	17
1.8. Перехват ошибок	18
1.9. Символьные вычисления	19
Контрольные вопросы	21
2. Приближенные методы решения нелинейных и трансцендентных уравнений	22
2.1. Постановка задачи	22
2.2. Метод половинного деления	23
2.3. Метод простых итераций	25
2.4. Метод Ньютона (метод касательных)	28
2.5. Метод хорд (метод линейной аппроксимации)	31
Контрольные вопросы	32

3. Численное интегрирование	33
3.1. Формула прямоугольников	33
3.2. Метод трапеций	34
3.3. Формула Симпсона.....	37
Контрольные вопросы.....	42
4. Приближенное решение систем нелинейных уравнений.....	43
4.1.Метод Ньютона.....	43
4.2. Метод градиента.....	50
Контрольные вопросы.....	54
5. Интерполирование функций	55
5.1. Интерполяционная формула Лагранжа.....	55
5.2. Интерполяционная формула Ньютона.....	58
5.3. Задача обратного интерполирования	61
5.4. Численное дифференцирование	62
Контрольные вопросы.....	65
6. Решение обыкновенных дифференциальных уравнений.....	67
6.1. Метод Рунге–Кутты.....	67
6.2. Метод Милна	69
6.3. Метод Адамса	70
Контрольные вопросы.....	81
Литература.....	82

Трошина Галина Васильевна

**РЕШЕНИЕ ЗАДАЧ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ
С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ
ПАКЕТА MathCad**

Учебное пособие

Редактор *Н.В. Городник*
Выпускающий редактор *И.П. Брованова*
Дизайн обложки *А.В. Ладыжская*
Компьютерная верстка *Л.А. Веселовская*

Подписано в печать 14.12.2009. Формат 60 × 84 1/16. Бумага офсетная. Тираж 100 экз.
Уч.-изд. л. 5,11. Печ. л. 5,5. Изд. № 302. Заказ № Цена договорная

Отпечатано в типографии
Новосибирского государственного технического университета
630092, г. Новосибирск, пр. К. Маркса, 20