

**И.А. БЕДАРЕВ
Ю.В. КРАТОВА
Н.Н. ФЕДОРОВА
И.А. ФЕДОРЧЕНКО**

Методы вычислений в пакете MathCAD

НОВОСИБИРСК 2013

СОДЕРЖАНИЕ

Введение.....	4
Глава 1. Методы решения нелинейных алгебраических уравнений.....	6
1.1. Постановка задачи	6
1.2. Приближенные методы	7
1.3. Стандартные функции MathCAD	14
Глава 2. Численные методы линейной алгебры.....	20
2.1. Матрицы и простейшие операции над ними.....	20
2.2. Определитель, ранг, обратная матрица.....	21
2.3. Операции с матрицами в MathCAD	22
2.4. Системы линейных алгебраических уравнений	26
2.5. Точные методы решения СЛАУ	27
2.6. Итерационные методы решения СЛАУ	33
2.7. Решение СЛАУ в пакете MathCAD.....	38
2.8. Собственные значения и собственные векторы.....	41
Глава 3. Интерполяция и приближение функций	47
3.1. Постановка задачи интерполяции	48
3.2. Локальная интерполяция.....	48
3.3. Глобальная интерполяция	53
3.4. Полином Лагранжа	54
3.5. Полином Ньютона	56
3.6. Ряды Фурье.....	58
3.7. Метод наименьших квадратов.....	64
3.9. Заключение	68
Глава 4. Решение задач оптимизации.....	70
4.1. Поиск экстремума функции одной переменной аналитическим методом.....	70
4.2. Поиск экстремума функции с помощью встроенных функций MathCAD	72
Глава 5. Численное дифференцирование и интегрирование ..	77
5.1. Численное дифференцирование	77
5.2. Численное интегрирование.....	84
Глава 6. Решение обыкновенных дифференциальных уравнений.....	96
6.1. Задача Коши	96

6.2. Краевая задача для ОДУ второго порядка.....	115
Глава 7. Решение уравнений в частных производных.....	134
7.1. Основные понятия. Постановка задачи	134
7.2. Параболические уравнения.....	138
7.3. Гиперболические уравнения.....	153
7.4. Приближенные методы решения уравнения Пуассона	158
Заключение	162
Список рекомендованной литературы	163

Введение

В настоящее время компьютерное моделирование завоевывает все новые области приложения в сферах проектирования и создания технических изделий. Это связано как с усложнением объектов производства, так и с ростом возможностей ЭВМ. Распространенное мнение о всемогуществе современных ЭВМ часто порождает впечатление, что математики избавились от всех хлопот, связанных с решением задач, и разработки новых методов для их решения не требуется. В действительности это не так, поскольку практика каждый раз ставит все новые, более сложные цели, возникают комплексные задачи, математическое моделирование проникает в новые области.

Математическое моделирование – это описание на абстрактном математическом языке различных явлений и процессов. Математическая модель объекта или явления – набор формул, таблиц, уравнений, описывающих поведение этого объекта или явления. Этапы математического моделирования включают:

- 1) Уточнение наиболее существенных фактов, свойств описываемого объекта или явления;
- 2) Построение математической модели – системы уравнений: алгебраических, функциональных, в частных производных;
- 3) Нахождение решений, точно или с помощью приближенных методов, которые специально разрабатывают для решения конкретной задачи;
- 4) Проверка адекватности модели: соответствие полученных решений основным фактам, экспериментам, если необходимо – уточнение модели, т.е. введение в модель новых уравнений или добавление новых слагаемых в существующие уравнения;
- 5) Параметрические исследования на основе построенной модели, получение новых сведений об объекте.

В современное время создано и широко применяется множество специальных программных пакетов и комплексов для решения различных физических и технических проблем, включая задачи строительства. Каждый из таких свободно распространяемых или коммерческих продуктов основан на применении

некоторого специфического круга математических моделей и вычислительных методов. Понимание принципов, заложенных в основу используемого программного инструментария, является необходимым требованием к квалификации современного инженера-специалиста. Поэтому основная цель курса - дать представление о многообразии математических моделей и методов, возникающих в процессе научно-исследовательской и проектной деятельности в области строительства; научить подбирать и модифицировать методы прикладной математики для решения поставленной задачи из предметной области и уметь анализировать полученное решение. В качестве помощника в реализации этих целей предлагается использовать систему компьютерной алгебры MathCAD. Данный программный продукт позволяет производить символьные и численные расчеты, используя при этом достаточно понятный интерфейс, не требующий больших затрат времени для знакомства с ним. Кроме стандартных функций, позволяющих решать некоторые задачи, в том числе рассматриваемые в настоящем пособии, область применения пакета существенно расширена за счет возможности использования средств программирования.

Учебное пособие предназначено для использования в учебном процессе студентами НГАСУ (Сибстрин), обучающимся по направлению 270800 «Строительство». Данный курс является неотъемлемой частью математической подготовки в соответствии с требованиями, отраженными в Федеральном Государственном стандарте специальности.

В пособии приведены сведения из линейной алгебры, функционального анализа, дифференциальных уравнений, уравнений в частных производных и численных методов, которые необходимы для понимания и активного освоения методов прикладной математики. Показаны примеры решения некоторых конкретных задач в пакете MathCAD.

Пособие состоит из семи глав, введения и списка литературы. Формулы и рисунки имеют двойную нумерацию с учетом номера главы.

Глава 1. Методы решения нелинейных алгебраических уравнений

1.1. Постановка задачи

Дано нелинейное алгебраическое уравнение

$$F(x)=0 \quad (1.1)$$

Нелинейность уравнения означает, что графиком функции является не прямая линия. Решить уравнение – это найти $x^* \in \mathbb{R}$: $F(x^*)=0$.

Значение x^* называют *корнем уравнения*. Нелинейное уравнение может иметь несколько корней. Геометрическая интерпретация такой ситуации представлена на рис. 1.1. Корнями уравнения (1.1) являются точки x_1^* , x_2^* , x_3^* , в которых функция $F(x)$ пересекает ось x .

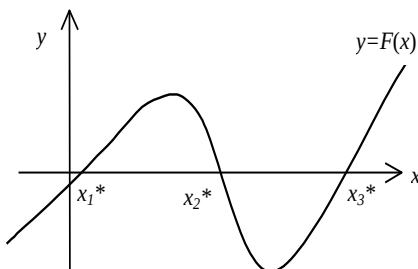


Рис. 1.1. Геометрическая иллюстрация уравнения (1.1)

Необходимое условие существования корня уравнения (1.1) и *достаточное условие единственности* следуют из известной теоремы Больцано-Коши. Пусть $F(x)$ непрерывна и $F(a)F(b)<0$ (т.е. на концах интервала функция имеет разные знаки). Тогда внутри отрезка $[a, b]$ существует корень уравнения $F(x)=0$. Корень будет единственным, если $F(x)$ не меняет знак на отрезке $[a, b]$, т.е. $F(x)$ – монотонная функция.

Методы решения уравнения (1.1) можно разделить на *точные* (аналитические) и *приближенные* (итерационные). В точных методах корень находится за конечное число действий и представляется некоторой алгебраической формулой. В приближенных методах процесс нахождения решения бесконечен. Решением называется бесконечная последовательность $\{x_n\}$, такая, что $\lim_{n \rightarrow \infty} x_n = x^*$. По определению предела, для любого

сколь угодно малого наперед заданного ε найдется такое N , что при $n>N$, $|x_n - x^*|<\varepsilon$. Члены этой последовательности x_n назы-

ваются последовательными приближениями к решению, или *итерациями*. Наперед заданное число ε называют *точностью* метода, а N — это количество итераций, которое необходимо выполнить, чтобы получить решение с точностью ε . Существует различные методы нахождения приближенного решения, т.е.

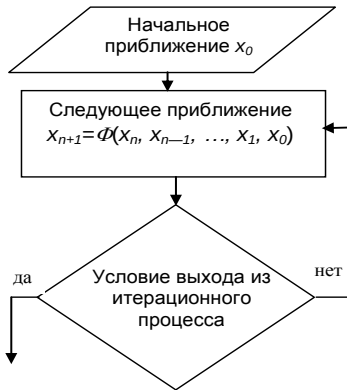


Рис. 1.2. Этапы итерационного процесса

способы построения последовательности итераций $\{x_n\}$, однако все они имеют общие этапы, изображенные на рис. 1.2.

Используются различные критерии остановки итерационного процесса:

- $|x_n - x^*| < \varepsilon$. К сожалению, это условие не всегда возможно проверить, т.к. x^* неизвестно;
- $|F(x_n)| < \varepsilon$, где $F(x_n)$ — невязка метода;
- $|x_{n+1} - x_n| < \varepsilon$, т.е. разница между соседними итерациями стала мала.

1.2. Приближенные методы

Прежде чем использовать приближенный метод, уравнение надо исследовать на наличие корней и уточнить, где эти корни находятся, т.е. найти интервалы изоляции корней. *Интервалом изоляции корня* называется отрезок, на котором корень уравнения существует и единственный. Каждому корню соответствует свой интервал изоляции. Если корней несколько, то для каждого нужно найти интервал изоляции. Существуют различные способы исследования функции: аналитический, табличный, графический. *Аналитический* способ состоит в нахождении экстремумов функции $F(x)$, исследовании ее поведения при $x \rightarrow \infty$, нахождении участков возрастания и убывания функции. *Табличный* способ — это построение таблицы, состоящей из столбца аргумента x и столбца значений функции $F(x)$. О наличии корней свиде-

тельствуют смена знака функции. Чтобы не произошла потеря корней, шаг изменения аргумента должен быть достаточно мелким, а интервал изменения достаточно большим. И, наконец, *графический способ* – это построение графика функции $F(x)$ и определение числа корней по количеству пересечений графика с осью x . Ниже для иллюстрации приведено графическое исследование уравнения $F(x)=0,4 \cdot 2^x - 0,5x - 1=0$.

На рис. 1.3 приведены построенные с помощью MathCAD графики функций $F_1(x)=0,4 \cdot 2^x$ и $F_2(x)=0,5x + 1$. Корнями являются точки, в которых пересекаются два графика. Рисунок показывает, что исходное уравнение имеет два корня, расположенные на интервалах $[-3,0]$ и $[0, 3]$.

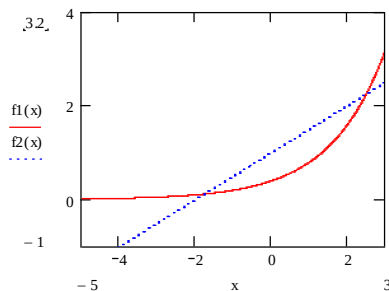


Рис. 1.3. Графический способ нахождения интервалов изоляции

Пусть интервалы изоляции корней известны. Познакомимся с несколькими итерационными методами, позволяющими найти корень на известном интервале изоляции $[a, b]$.

1.2.1. Метод деления отрезка пополам (дихотомии)

Найдем середину отрезка $[a, b]$: $c=(a+b)/2$. Корень остался на одной из частей: $[a, c]$ или $[c, b]$. Если $F(a) \cdot F(c) < 0$, то корень попал на отрезок $[a, c]$, тогда деление отрезка можно повторить, приняв в качестве нового правого конца точку c , т.е. $b=c$. В противном случае корень попал на половину $[c, b]$, и необходимо изменить значение левого конца отрезка: $a=c$. Поскольку корень всегда заключен внутри отрезка, итерационный процесс можно останавливать, если длина отрезка станет меньше заданной точности: $|b - a| < \varepsilon$.

В случае сложных уравнений вычисления приходится проводить с использованием ЭВМ. На рис. 1.4 приведен текст программы MathCAD, реализующей метод дихотомии для решения

уравнения $f(x)=x^2-x-1=0$. Метод реализован в виде функции от аргументов a, b (концы интервала изоляции) и точности метода ε . Вызов этой функции при значениях $a = -1, b = 0, \varepsilon = 0,01$ дает значение корня $-0,617$, при этом невязка уравнения, которую вычисляют, чтобы убедиться в правильности решения, равна $-1.892 \cdot 10^{-3}$.

$$f(x) := x^2 - x - 1$$

$$\text{mdp}(a, b, \varepsilon) := \left| \begin{array}{l} \text{while } |b - a| > \varepsilon \\ \quad c \leftarrow \frac{(a + b)}{2} \\ \quad a \leftarrow c \text{ if } f(a) \cdot f(c) > 0 \\ \quad b \leftarrow c \text{ otherwise} \\ c \end{array} \right.$$

$$\text{mdp}(-1, 0, 0.01) = -0.617 \quad f(\text{mdp}(-1, 0, 0.01)) = -1.892 \times 10^{-3}$$

Рис.1.4. Метод деления отрезка пополам

1.2.2. Метод хорд

В этом методе кривая $F(x)$ заменяется прямой линией — хордой, стягивающей точки $(a, F(a))$ и $(b, F(b))$. В зависимости от знака выражения $F(a)F'(a)$ метод хорд имеет два варианта, изображенных на рис. 1.5а, б.

Пусть $F(a)F'(a) > 0$ (рис. 1.5а). Тогда $x_0 = b$, точка a будет оставаться неподвижной. Следующее приближение x_1 находим как точку пересечения хорды, соединяющей точки $(a, F(a))$ и $(x_0, F(x_0))$ с осью x . Уравнение хорды:

$$y = F(a) + \frac{F(x_0) - F(a)}{x_0 - a}(x - a). \text{ Тогда точка пересечения хорды с}$$

$$\text{осью } x: x_1 = a - \frac{F(a)(x_0 - a)}{F(x_0) - F(a)}.$$

Пусть теперь $F(a)F'(a) < 0$ (рис. 1.5б). Тогда $x_0 = a$, точка b неподвижна. Проведем хорду, соединяющую точки $(b, F(b))$ и

$(x_0, F(x_0))$: $y = F(x_0) + \frac{F(b) - F(x_0)}{b - x_0}(x - x_0)$. Вычисляем точку

пересечения хорды с осью x : $x_1 = x_0 - \frac{F(x_0)(b - x_0)}{F(b) - F(x_0)}$.

На следующей итерации в качестве x_0 надо взять вычисленное значение x_1 . Окончание итерационного цикла в этом методе происходит по условию малости невязки уравнения: $|F(x_1)| < \varepsilon$.

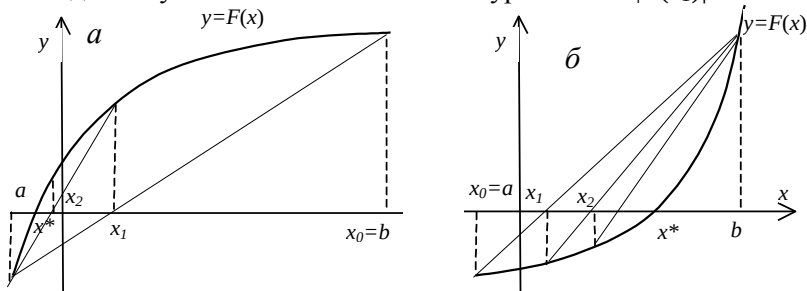


Рис. 1.5. Метод хорд для $F(a)F'(a) > 0$ (а) и $F(a)F'(a) < 0$ (б)

1.2.3. Метод Ньютона (касательных)

Как и предыдущий, этот метод основан на замене исходного нелинейного уравнения (1.1) линейным уравнением, которое можно легко решить. Иллюстрация метода представлена на рис. 1.6. Пусть x_0 – начальное приближение. Построим касательную к функции $y = F(x)$, проходящую через точку $(x_0, F(x_0))$. Найдем пересечение касательной

$y = F(x_0) + F'(x_0)(x - x_0)$ с осью x : $x_1 = x_0 - \frac{F(x_0)}{F'(x_0)}$. На следующей

итерации в качестве x_0 надо взять вычисленное значение x_1 . Окончание итерационного цикла, как и в методе хорд, выполняется по невязке уравнения: $|F(x_1)| < \varepsilon$.

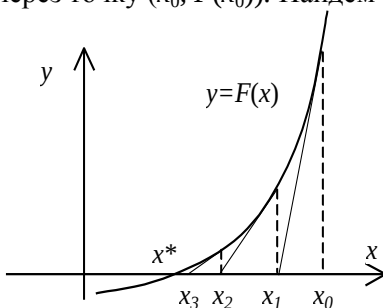


Рис. 1.6. Метод Ньютона

Как показывают практика и теоретические оценки, метод Ньютона позволяет достаточно быстро получить решение. Недостатком метода является то, что для некоторых функций $F(x)$ при неудачном выборе начального приближения метод расходится. Ситуацию легко исправить, если выбрать x_0 ближе к x^* .

На рис. 1.7 показана реализация метода Ньютона в пакете MathCAD. В данном примере: x_0 – начальное приближение, ε – точность, i – количество итераций. В качестве критерия сходимости использовано вычисление невязки. Из примера видно, что для достижения точности 0,001 потребовалось 4 итерации.

$$f(x) := x^2 - x - 1$$

$$df(x) := \frac{d}{dx} f(x) \quad x_0 := 0$$

$$\text{NEWTON}(x, \varepsilon) := \begin{array}{l} i \leftarrow 0 \\ \text{while } |f(x)| > \varepsilon \\ \quad x \leftarrow x - \frac{f(x)}{df(x)} \\ \quad i \leftarrow i + 1 \\ (x \quad i) \end{array}$$

$$\text{NEWTON}(x_0, 0.001) = (-0.618 \quad 4)$$

Рис. 1.7. Метод Ньютона

1.2.4. Метод секущих

Метод используется, когда вычисление производной $F'(x)$ занимает много времени, а также, если функция $F(x)$ задана таблично. Для этого метода необходимо задать два начальных приближения: x_0, x_1 , поэтому этот метод называется двухшаговым (все рассмотренные выше методы были одношаговыми). Значение производной можно вычислить с помощью конечно-

разностного соотношения $F'(x_1) \approx \frac{F(x_1) - F(x_0)}{x_1 - x_0}$ (см. главу 5).

Подставляя это соотношение в формулу метода Ньютона, получим $x_2 = x_1 - \frac{F(x_1)(x_1 - x_0)}{F(x_1) - F(x_0)}$. На следующей итерации в качестве

приближений x_0, x_1 возьмем уже вычисленные значения x_1, x_2 , т.е. $x_0 = x_1, x_1 = x_2$, и будем вычислять новое приближение x_2 . Окончание итерационного цикла производится по невязке уравнения: $|F(x_2)| < \varepsilon$.

1.2.5. Метод простой итерации (МПИ)

Этот метод является обобщением всех описанных выше одношаговых методов. Слово «простой» означает, что для вычисления следующего приближения необходимо знать только одно предыдущее приближение. С помощью эквивалентных преобразований приведем исходное уравнение (1.1) к виду, удобному для применения метода простой итерации: $x = \varphi(x)$. Выберем начальное приближение $x_0 \in [a, b]$. Следующие итерации находим по формуле: $x_{k+1} = \varphi(x_k)$. Итерационный процесс заканчивается, если $|\varphi(x_k) - x_k| < \varepsilon$, или, что тоже самое, $|x_{k+1} - x_k| < \varepsilon$. Иллюстрация метода для $\varphi(x) > 0$ и $\varphi(x) < 0$ представлена на рис. 1.8а, б.

Способ сведения исходного уравнения к виду $x = \varphi(x)$ очень важен, поскольку от вида функции $\varphi(x)$ зависит, будет ли итерационный процесс сходиться или нет.

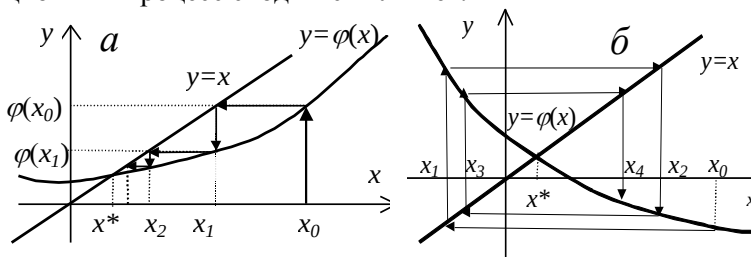


Рис. 1.8. Сходящийся метод простой итерации

Чтобы понять, каким условиям должна удовлетворять порождающая итерации функция $\varphi(x)$, проведем некоторые теоретические оценки. Пусть x_k – известное приближение, отличающееся от искомого корня на величину погрешности $|x_k - x^*|$. Следующее приближение x_{k+1} будет отличаться от корня на величину $|x_{k+1} - x^*| = |\varphi(x_k) - x^*| = |\varphi(x_k) - \varphi(x^*)| = |\varphi'(\xi)| |x_k - x^*|$, где ξ – некоторая средняя точка отрезка (x_k, x^*) . Здесь использована теорема Лагранжа о конечном приращении, а также равенство $x^* = \varphi(x^*)$, которое справедливо, поскольку x^* – корень. В сходящихся методах каждое следующее приближение должно быть ближе к корню, чем предыдущее, а это справедливо, если

$|\varphi'(\xi)| \leq q < 1$. Понятно, что чем меньше число q , тем быстрее сходится итерационный процесс. Можно получить оценку:

$$|x_{k+1} - x^*| < q \cdot |x_k - x^*| < q^2 \cdot |x_{k-1} - x^*| < \dots < q^k \cdot |x_0 - x^*|.$$

Если удастся показать, что $|x_{k+1} - x^*| < q \cdot |x_k - x^*|^\alpha$, $\alpha > 1$, то говорят, что метод сходится с порядком α .

Можно доказать, что *необходимым условием сходимости МПИ* является $|\varphi'(x^*)| < 1$. Это условие трудно проверить, т.к. корень неизвестен. Но можно проверить более сильное *достаточное условие сходимости МПИ* $|\varphi'(x)| < 1$, где x – любая точка некоторого отрезка $[a, b]$, содержащего корень.

Проверим, выполняется ли необходимое условие метода Ньютона, для которого $\varphi(x) = x - \frac{F(x)}{F'(x)}$. Вычислим производную:

$$\varphi'(x^*) = 1 - \frac{(F'(x^*))^2 - F(x^*)F''(x^*)}{(F'(x^*))^2} = \frac{F(x^*)F''(x^*)}{(F'(x^*))^2} = 0 < 1.$$

Следовательно, метод Ньютона всегда сходится в некоторой окрестности корня x^* . Можно показать, что $|x_{k+1} - x^*| < q \cdot |x_k - x^*|^2$, т.е. $\alpha=2$: метод сходится со вторым порядком. Порядок сходимости метода Чебышева еще выше ($\alpha=3$), для метода секущих $\alpha=1,67$, а для метода дихотомии $\alpha=1$.

Если необходимое условие нарушается, то МПИ будет расходиться. Иллюстрации расходящихся итерационных методов приведены на рис. 1.9а, б для $\varphi'(x) > 0$ и $\varphi'(x) < 0$, соответственно.

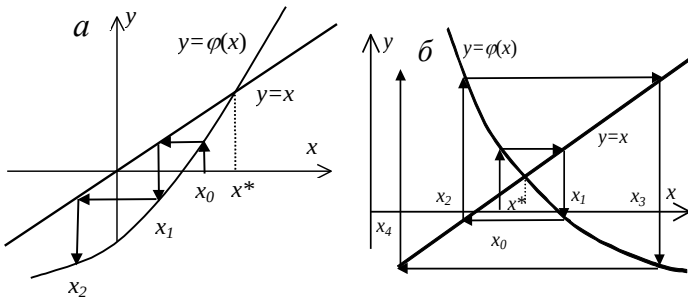


Рис. 1.9. Расходящийся метод простой итерации

1.2.6. Метод релаксации

Метод релаксации – это универсальный вариант МПИ, в котором $\varphi(x) = x - \tau F(x)$. Параметр релаксации $\tau \neq 0$ подберем таким образом, чтобы выполнялось достаточное условие сходимости метода: $|\varphi'(x)| < 1$ для всех $x \in [a, b]$: $|\varphi'(x)| = |1 - \tau F'(x)| < 1$, или $-2 < \tau F'(x) < 0$. Таким образом, при $F'(x) > 0$, $x \in [a, b]$, условием сходимости будет $-2/F'(x) < \tau < 0$. При $F'(x) < 0$ параметр надо выбирать из условия $-2/F'(x) > \tau > 0$. Ясно, что при малых τ МПИ будет сходиться медленно, т.к. расстояние между соседними итерациями $|x_{k+1} - x_k| = |\tau| \cdot |F'(x)|$ мало. Можно ли выбрать такое значение параметра, при котором скорость сходимости будет максимальной? Несложный анализ показывает, что оптимальным значением будет $\tau = 2/(M+m)$, где $M = \max_{x \in [a, b]} F'(x)$,

$m = \min_{x \in [a, b]} F'(x)$, а скорость сходимости в этом случае будет

определяться константой $q = \frac{M - m}{M + m}$.

1.3. Стандартные функции MathCAD

Для решения уравнения (1.1) в MathCAD служит функция `root`, реализующая описанный выше метод секущих. Если $F(x)$ — это полином, то вычислить все его корни можно также с помощью функции `polyroots`.

Встроенная функция `root` в зависимости от типа задачи может иметь либо два аргумента: `root (f(x), x)`, либо четыре аргумента: `root (f(x), x, a, b)`. Здесь $f(x)$ – скалярная функция, определяющая уравнение (1.1); x – скалярная переменная, относительно которой решается уравнение; a, b – границы интервала, внутри которого происходит поиск корня.

Первый тип функции `root` требует дополнительного задания *начального значения* переменной x . Для этого нужно просто предварительно присвоить x некоторое число. Поиск корня будет производиться вблизи этого числа. Таким образом, присвое-

ние начального значения требует априорной информации о примерной локализации корня.

Рассмотрим решение уравнения $\sin(x) = 0$, которое имеет бесконечное количество корней $x_N = N \pi$ ($N = 0, \pm 1, \pm 2, \dots$). Для поиска корня средствами MathCAD требуется его предварительная локализация путем задания начального приближения, например, $x = 0.5$. MathCAD находит с заданной точностью только один корень $x_0 = 0$, лежащий наиболее близко к заданному начальному приближению. Если задать другое начальное значение, например, $x = 3$, то решением будет другой корень уравнения $x_1 = \pi$ и т.д.

На рис. 1.10 приведен пример вызова стандартной функции `root` с двумя аргументами для нахождения корней уравнения $\sin(x) = 0$, график функции $f(x) = \sin(x)$ и положение найденного корня.

$x := 0.5$

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

$s := \text{root}(f(x), x)$

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

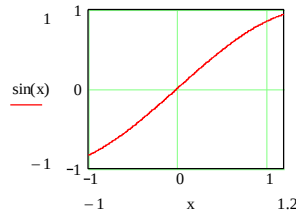


Рис.1.10. Использование стандартной функции `root` для решения нелинейного уравнения $\sin(x) = 0$

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

Иногда удобнее задавать не начальное приближение к корню, а интервал $[a, b]$, внутри которого корень заведомо находится. В этом случае следует использовать функцию `root` с четырьмя аргументами, а присваивать начальное значение x не нужно. Поиск корня осуществляется в промежутке между a и b .

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

$x = 0 \quad \sin(x) = 0$

Рис. 1.11. Поиск корня алгебраического уравнения в заданном интервале

При этом явный вид функции $f(x)$ может быть определен непосредственно в теле функции `root`. На рис. 1.11 приведен листинг программы с использованием этого варианта функции `root`.

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

- внутри интервала $[a, b]$ не должно находиться более одного корня, иначе будет найден один из них, заранее неизвестно, какой именно;

- значения $f(a)$ и $f(b)$ должны иметь разный знак, иначе будет выдано сообщение об ошибке.

Если уравнение не имеет действительных корней, но имеет мнимые, то их также можно найти. Ниже приведен пример, в котором уравнение $x^2 + 1 = 0$, имеющее два чисто мнимых корня, решается два раза с разными начальными значениями.

Для решения этого уравнения второй вид функции `root` (с четырьмя аргументами) неприменим, поскольку $f(x)$ является положительно-определенной, и указать интервал, на границах которого она имела бы разный знак, невозможно.

Отметим, что $f(x)$ может быть функцией не одного, а любого количества аргументов.

$x := 0.5$

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

$x := -0.5$

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

Рис. 1.12. Поиск мнимого корня

Эта возможность проиллюстрирована на рис. 1.13 на примере

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

$$x := 1$$

$$y := 0$$

$$\text{root}(f(x, y), x) = -1.732i$$

$$\text{root}(f(x, y), y) = 2$$

функции двух переменных $f(x, y) = x^2 - y^2 + 3$. В самой функции `root` необходимо определить, относительно какого из аргументов следует решить уравнение. Затем уравнение $f(x, 0) = 0$ решается относительно переменной x , а потом – другое уравнение $f(1, y) = 0$ относительно переменной y .

Рис. 1.13. Поиск корня уравнения, заданного функцией двух переменных. При численном решении уравнений относительно одной из переменных необходимо предварительно определить значения остальных переменных. Иначе попытка вычислить уравнения приведет к появлению ошибки **“This variable or function is not defined above”**, в данном случае говорящей о том, что другая переменная ранее не определена. Конечно, можно указать значение других переменных непосредственно внутри функции `root`.

Если функция $f(x)$ является полиномом, то все его корни можно определить, используя встроенную функцию `polyroots(v)`, где v – вектор, составленный из коэффициентов полинома. Поскольку полином N -й степени имеет ровно N корней (некоторые из них могут быть кратными), вектор v дол-

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

$$\text{polyroot}(v) = \begin{pmatrix} 0.992 \\ 1.004 + 7.177i \times 10^{-3} \\ 1.004 - 7.177i \times 10^{-3} \\ 3 \end{pmatrix}$$

Рис. 1.14. Поиск корня полинома

уравнения $f(x) = (x - 13)(x - 1)^3 = x^4 - 6x^3 + 12x^2 - 10x + 3 = 0$.

жен состоять из $N+1$ элемента. Результатом действия функции `polyroots` является вектор, составленный из N корней рассматриваемого полинома. На рис. 1.14 рассмотрен пример решения

Коэффициенты полинома записаны в виде вектора в первой строке примера. Первым в векторе должен идти свободный член полинома, вторым – коэффициент при x^1 и т.д. Последним, $N+1$ элементом вектора должен быть коэффициент при старшей степени x^N . Во второй строке показано действие функции `polyroots`. При этом численный метод вместо двух действительных единичных корней дает одинаковые мнимые числа. Однако малая мнимая часть этих корней находится в пределах погрешности, определяемой константой `TOL`, и не должна вводить пользователей в заблуждение. Необходимо помнить, что корни полинома могут быть комплексными, и ошибка вычислений может сказываться как на действительной, так и на комплексной части искомого корня.

В следующем примере, представленном на рис. 1.15, показано вычисление трех действительных корней полинома $f(x)=6 - 7x + x^3$ с понижением порядка полинома.

В этом примере используется вариант функции `root` с двумя аргументами. Приведенный на рисунке график функции $f(x)$ показывает, что уравнение имеет три действительных корня. Задавая начальное приближение $z = -2$, находим один из корней полинома: $x_1 = -3$. Затем исходный полином делится на $(z - x_1)$, и отыскивается второй корень $x_2 = 1$. Далее функция `root` еще раз вызывается для нахождения корня полинома первого порядка, получаемого делением исходного полинома на $(z - x_1)$ и $(z - x_2)$. Для каждого из найденных корней производится проверка – вычисляется невязка уравнения.

$$a_1 := 6 \quad a_2 := -7 \quad a_3 := 0 \quad a_4 := 1$$

$$f(y) := a_1 + a_2 \cdot y + a_3 \cdot y^2 + a_4 \cdot y^3$$

$$z := -2 \quad x_1 := \text{root}(f(z), z) \quad x_1 = -3 \quad |f(x_1)| = 2.463 \times 10^{-5}$$

$$z := 0.5 \quad x_2 := \text{root}\left(\frac{f(z)}{z - x_1}, z\right) \quad x_2 = 1 \quad |f(x_2)| = 6.078 \times 10^{-4}$$

$$z := 3 \quad x_3 := \text{root}\left[\frac{f(z)}{(z - x_1) \cdot (z - x_2)}, z\right] \quad x_3 = 2 \quad |f(x_3)| = 1.904 \times 10^{-4}$$

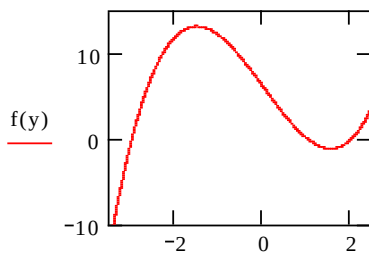


Рис. 1.15. Поиск корня полинома с понижением порядка

Глава 2. Численные методы линейной алгебры

Задачи линейной алгебры, успешно решаемые с помощью инструментов MathCAD, можно условно разделить на следующие группы:

- простейшие матричные операции (транспонирование, сложение, умножение на константу, перемножение матриц), которые сводятся к определенным арифметическим действиям над элементами матриц;
- вычисление определителей матриц, нахождение обратных матриц;
- решение систем линейных алгебраических уравнений (СЛАУ);
- вычисление собственных значений и собственных векторов.

2.1. Матрицы и простейшие операции над ними

Вещественной матрицей размерности $m \times n$ назовем таблицу из m строк и n столбцов, составленную из вещественных чисел:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} = \{x_{ij}\}_{i=1, \dots, m}^{j=1, \dots, n}. \quad (2.1)$$

Пусть $A = \{a_{ij}\}_{i=1, \dots, m}^{j=1, \dots, n}$, $B = \{b_{ij}\}_{i=1, \dots, m}^{j=1, \dots, n}$ – матрицы одинаковой размерности. Для них можно определить **операцию сложения**:

$$C = A + B, \quad C = \{c_{ij}\}_{i=1, \dots, m}^{j=1, \dots, n}, \quad c_{ij} = a_{ij} + b_{ij}, \quad i = 1, \dots, m, \quad j = 1, \dots, n,$$

умножения на числовую константу:

$$C = \lambda \cdot A, \quad C = \{c_{ij}\}_{i=1, \dots, m}^{j=1, \dots, n}, \quad c_{ij} = \lambda \cdot a_{ij}, \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

Операцию умножения матриц можно определить для матриц

$$A = \{a_{ij}\}_{i=1, \dots, m}^{j=1, \dots, n}, B = \{b_{ij}\}_{i=1, \dots, n}^{j=1, \dots, l}, \text{ тогда}$$

$$C = A \cdot B, \quad C = \{c_{ij}\}_{i=1, \dots, m}^{j=1, \dots, l}, \quad c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}. \quad (2.2)$$

Транспонирование - это операция, переводящая матрицу размера $m \times n$ в матрицу размера $n \times m$. При этом строки исходной

матрицы становятся столбцами транспонированной и наоборот. Транспонированную матрицу обычно обозначают A' . Элементы транспонированной матрицы A' связаны с элементами исходной матрицы A простым соотношением: $a'_{ij} = a_{ji}$.

Можно показать, что для матричных операций справедливо:

- $A + B = B + A$
- $(A + B) + C = A + (B + C)$
- $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- существование нулевой матрицы O : $A + O = A$
- существование обратной по отношению к операции сложения матрицы $-A$, такой, что $A + (-A) = O$
- существование единичной матрицы I , $A \cdot I = I \cdot A = A$,

$$I = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

- $A \cdot B \neq B \cdot A$ (нет коммутативности операции умножения)

Нормой матрицы A называется скалярная величина, удовлетворяющая условиям:

- 1) $\|A\| \geq 0$, и если $\|A\| = 0$, то A - нулевая матрица
- 2) $\|\alpha \cdot A\| = |\alpha| \cdot \|A\| \quad \forall$ вещественного α
- 3) $\|A + B\| \leq \|A\| + \|B\|$ для любых матриц A, B .

В какой-то степени норму A можно образно представлять как величину, характеризующую все элементы матрицы. В качестве нормы матрицы используют различные величины, например:

$$\|A\| = \max_j \sum_{i=1}^n |a_{ij}|. \text{ Возможны и другие определения нормы [1].}$$

2.2. Определитель, ранг, обратная матрица

Рассмотрим несколько важных операций линейной алгебры, которые также реализованы в MathCAD в виде стандартных функций. Однако их применение требует гораздо большего

внимания пользователя, чем операции, описанные в предыдущем параграфе.

Определителем (детерминантом) квадратной матрицы A порядка n называется многочлен порядка n , в котором каждое слагаемое содержит по одному элементу из каждой строки и по одному элементу из каждого столбца матрицы, причем члены ряда суммируются с учетом индексов элементов, входящих в сомножитель. Суммирование идет по всем возможным перестановкам порядка n , поэтому число слагаемых в ряде равно $n!$.

Определители первого, второго и третьего порядков вычисляются по простым формулам:

$$\begin{aligned}\Delta_1 &= a_{11} \\ \Delta_2 &= a_{11} \cdot a_{22} - a_{21} \cdot a_{12} \\ \Delta_3 &= a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} + a_{21} \cdot a_{32} \cdot a_{13} - \\ &\quad - a_{31} \cdot a_{22} \cdot a_{13} - a_{21} \cdot a_{12} \cdot a_{33} - a_{11} \cdot a_{23} \cdot a_{32}\end{aligned}\quad (2.3)$$

Определитель порядка n вычисляется по формуле

$$\Delta_n = \sum_{\alpha_1, \alpha_2, \dots, \alpha_n} (-1)^{N(\alpha_1, \alpha_2, \dots, \alpha_n)} \cdot a_{\alpha_1 1} \cdot a_{\alpha_2 2} \cdot \dots \cdot a_{\alpha_n n}, \quad (2.4)$$

где $\alpha_1, \alpha_2, \dots, \alpha_n$ - перестановки чисел от 1 до n , $N(\alpha_1, \alpha_2, \dots, \alpha_n)$ - число инверсий в перестановке.

Понятие определителя играет важную роль в вычислении ранга матрицы и обратной матрицы. **Рангом** матрицы называется наибольшее натуральное число k , для которого существует отличный от нуля определитель k -ого порядка подматрицы, составленной из любого пересечения k строк и k столбцов исходной матрицы. Если ранг квадратной матрицы A порядка n совпадает с ее порядком, т.е. $\Delta_n \neq 0$ то матрица называется невырожденной, и для нее можно определить так называемую **обратную матрицу** A^{-1} , обладающую следующим свойством: $A \cdot A^{-1} = A^{-1} \cdot A = I$, где I – единичная матрица.

2.3. Операции с матрицами в MathCAD

Описанные выше операции реализованы в MathCAD в виде операторов, запись которых максимально приближена к математическим выражениям. При этом для матричных операций за-

частую используются те же знаки математических операций, что и для скаляров, однако вычисления проводятся по приведенным выше формулам.

Заметим, что векторы являются частным случаем матриц размерности $N \times 1$ (вектор-столбец). Кроме стандартных матричных операций к векторам можно применять ряд специфических операций (модуль вектора, скалярное и векторное произведение и др.). Для того чтобы аналогичные действия выполнить с вектор - строкой, т.е. матрицей размерности $1 \times N$, ее нужно сначала транспонировать.

Для работы с матрицами и векторами предназначена панель инструментов **Matrix (Матрица)**, показанная на рис. 2.1. В таблице 2.1. приведен список операций и функций с матрицами и векторами, а на рис. 2.2 и 2.3. – некоторые примеры их использования.

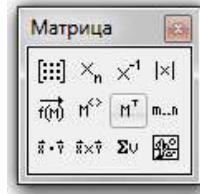
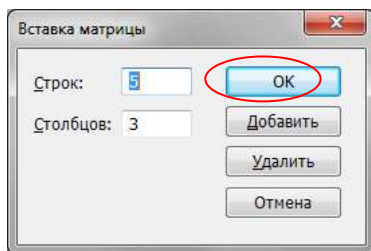


Рис. 2.1. Панель инструментов «Матрица»

Таблица 2.1. Операции с матрицами и векторами

Название операции	Вызов/обозначение в MathCAD	Горячие клавиши
Создание матриц	Меню: Insert – Matrix Панель инструментов:	<Ctrl>+<M>
Обращение к элементу матрицы	Панель инструментов:	<[>
Выделение столбца	Панель инструментов:	<Ctrl>+<6>
Создание единичной матрицы	Identity (N), N - размерность матрицы	-
Создание матрицы из вектора	Diag (V) , V – вектор главной диагонали	-
Создание матриц на основе функ-	Matrix (M, N, f) , M, N – размерность матрицы,	-

ции	$f(i, j)$ – функция, задающая элемент матрицы	
Размерность матрицы A / вектора v	rows(A) – число строк A cols(A) – число столбцов A length(v) – число элементов вектора last(v) – индекс последнего элемента вектора	-
Выделение подматрицы из матрицы A	Submatrix(A, ir, jr, ic, jc) – часть матрицы между строками ir, jr и столбцами ic, jc	-
Слияние матриц	augment(A, B, C, ...) – новая матрица формируется слиянием матриц аргументов слева – направо; stack(A, B, C, ...) – новая матрица формируется слиянием матриц аргументов сверху – вниз	-
Ранг матрицы	rank(A)	-
Определитель матрицы	Панель инструментов: $ x $	<Shift>+< >
Обратная матрица	Панель инструментов: x^{-1}	
Умножение матриц A, B	A*B	
Транспонирование матрицы	Панель инструментов: m^T	<Ctrl>+<1>
Норма матрицы A	norm1(A), norm2(A), norme(A), normi(A)	-



a

Сложение двух матриц *б*

$$\begin{pmatrix} 1 & -3 \\ 4 & 6 \\ 2 & -5 \end{pmatrix} + \begin{pmatrix} 3 & 4 \\ -3 & 0 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ 1 & 6 \\ 4 & -2 \end{pmatrix}$$

Умножение матриц

$$\begin{pmatrix} 1 & 3 \\ 6 & -2 \\ 5 & 7 \\ -2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 3 & -4 & 5 \\ 8 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 27 & -7 & 8 \\ 2 & -22 & 28 \\ 71 & -27 & 32 \\ 26 & 4 & -6 \end{pmatrix}$$

в

Определитель и ранг квадратной матрицы

$$A := \begin{pmatrix} 1 & 1 & 1 \\ 3 & 0 & 4 \\ 0 & 2 & 6 \end{pmatrix} \quad |A| = -20$$

$$\text{rank}(A) = 3$$

Обратная матрица

$$B := A^{-1} = \begin{pmatrix} 0.4 & 0.2 & -0.2 \\ 0.9 & -0.3 & 0.05 \\ -0.3 & 0.1 & 0.15 \end{pmatrix}$$

г

Рис. 2.2. Фрагменты рабочего листа MathCAD, представляющие создание матрицы (*a*), сложение матриц (*б*), произведение матриц (*в*), нахождение определителя и обратной матрицы (*г*)

Умножение матриц неправильной размерности

$$\begin{pmatrix} 1 & 3 \\ 6 & -2 \\ 5 & 7 \\ -2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 9 & 0 \\ 3 & -2 & 4 \\ 5 & 6 & 2 \end{pmatrix} = \mathbf{I}$$

Эти измерения массивов не соответствуют

Рис. 2.3. Фрагмент рабочего листа MathCAD, представляющий умножение матриц несоответствующих размерностей

2.4. Системы линейных алгебраических уравнений

Пусть задана система линейных алгебраических уравнений (СЛАУ) размерности m , которую можно записать в скалярном:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \dots + a_{1m} \cdot x_m = f_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \dots + a_{2m} \cdot x_m = f_2 \\ \dots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \dots + a_{mm} \cdot x_m = f_m \end{cases} \quad (2.5)$$

векторном:

$$\begin{pmatrix} a_{11} \\ a_{21} \\ \dots \\ a_{m1} \end{pmatrix} x_1 + \begin{pmatrix} a_{12} \\ a_{22} \\ \dots \\ a_{m2} \end{pmatrix} x_2 + \dots + \begin{pmatrix} a_{1m} \\ a_{2m} \\ \dots \\ a_{mm} \end{pmatrix} x_m = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_m \end{pmatrix} \quad (2.5')$$

или матричном виде:

$$A\vec{x} = \vec{f}, \quad (2.5'')$$

где $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{pmatrix}$, $\vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_{m-1} \\ f_m \end{pmatrix}$, $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{m-1} \\ x_m \end{pmatrix}$.

Здесь A – матрица системы, $\vec{f}$ – вектор правых частей, $\vec{x}$ – вектор неизвестных. Назовем расширенной матрицей системы матрицу A' , дополненную столбцом вектора правых частей:

$$A' = \begin{pmatrix} a_{11} & \dots & a_{1m} & f_1 \\ a_{21} & \dots & a_{2m} & f_2 \\ \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{mm} & f_m \end{pmatrix}.$$

Система имеет решение, если $\det A \neq 0$. По определению решения, подставив вектор $\vec{x}$ в СЛАУ, получим m тождественных уравнений.

Эффективность способов решения системы (2.5) во многом зависит от структуры и свойств матрицы A : размерности, обусловленности, симметричности, заполненности (т.е. соотношения между числом ненулевых и нулевых элементов) и др.

Все методы решения СЛАУ можно разбить на два класса: **прямые** (точные) и **итерационные** (приближенные).

Прямые методы позволяют получить решение за конечное число арифметических операций. Если операции реализуются точно, то и решение будет точным (поэтому класс прямых методов еще называют точными методами). В итерационных методах решением является предел некоторой бесконечной последовательности единообразных действий.

2.5. Точные методы решения СЛАУ

При небольшой размерности системы m можно использовать формулы Крамера, которые позволяют находить неизвестные в виде дробей, знаменателем которых является определитель матрицы системы, а числителем – определители матриц A_i , полученных из A заменой столбца коэффициентов при вычисляемом неизвестном столбцом вектора правых частей:

$$x_i = \frac{\det A_i}{\det A} \quad (i = 1, 2, \dots, m).$$

Размерность системы (т.е. число m) является главным фактором, из-за которого **формулы Крамера** не могут быть использованы для численного решения СЛАУ большого порядка. При непосредственном раскрытии определителей решение системы с m неизвестными требует порядка $m!m$ арифметических операций. Таким образом, для решения системы, например, из $m = 100$ уравнений потребуется совершить 10^{158} операций, что не под силу даже самым мощным современным ЭВМ.

Для решения небольших систем используют метод **обратной матрицы**. Если $\det A \neq 0$, то существует обратная матрица A^{-1} . По определению обратной матрицы: $AA^{-1} = A^{-1}A = I$, где I – единичная матрица. Если обратная матрица известна, то, умножая на нее СЛАУ слева, получим:

$$A^{-1}A\vec{x} = A^{-1}\vec{f}, \quad I\vec{x} = A^{-1}\vec{f}, \quad \vec{x} = A^{-1}\vec{f}.$$

Следовательно, решение СЛАУ свелось к умножению известной обратной матрицы на вектор правых частей. Таким образом, задача решения СЛАУ и задача нахождения обратной матрицы

связаны между собой, поэтому часто решение СЛАУ называют задачей обращения матрицы. Проблемы применения этого метода те же, что и при использовании метода Крамера: нахождение обратной матрицы – трудоемкая операция.

Наиболее популярным точным способом решения линейных систем вида (2.5) является **метод Гаусса**, или последовательного исключения неизвестных. Метод состоит из двух этапов: прямого и обратного. На первом этапе исходная система с помощью эквивалентных преобразований сводится к системе с треугольной матрицей, на втором этапе решается система с треугольной матрицей. Эквивалентными преобразованиями будут следующие преобразования расширенной матрицы A' :

- перестановка строк;
- умножение строк на ненулевую константу;
- сложение строк.

Используя эти преобразования, перепишем исходную систему так, чтобы один из коэффициентов столбца был равен 1, а все коэффициенты, стоящие ниже, были равны 0.

Пусть в исходной системе уравнений

$$\begin{cases} a_{11}^{(0)} \cdot x_1 + a_{12}^{(0)} \cdot x_2 + \dots + a_{1m}^{(0)} \cdot x_m = f_1^{(0)} \\ a_{21}^{(0)} \cdot x_1 + a_{22}^{(0)} \cdot x_2 + \dots + a_{2m}^{(0)} \cdot x_m = f_2^{(0)} \\ \dots \\ a_{m1}^{(0)} \cdot x_1 + a_{m2}^{(0)} \cdot x_2 + \dots + a_{mm}^{(0)} \cdot x_m = f_m^{(0)} \end{cases}$$

первый элемент $a_{11}^{(0)} \neq 0$. Назовем его ведущим элементом первой строки. Поделим все элементы этой строки на $a_{11}^{(0)}$ и исключим x_1 из всех последующих строк, начиная со второй, путем вычитания первой (преобразованной), умноженной на коэффициент при x_1 в соответствующей строке. Получим

$$\begin{cases} x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + \dots + a_{1m}^{(1)} \cdot x_m = f_1^{(1)} \\ a_{22}^{(1)} \cdot x_2 + a_{23}^{(1)} \cdot x_3 + \dots + a_{2m}^{(1)} \cdot x_m = f_2^{(1)} \\ \dots \\ a_{m2}^{(1)} \cdot x_2 + a_{m3}^{(1)} \cdot x_3 + \dots + a_{mm}^{(1)} \cdot x_m = f_m^{(1)} \end{cases}$$

Если $a_{22}^{(1)} \neq 0$, то, продолжая аналогичное исключение, приходим к системе уравнений с верхней треугольной матрицей

$$\begin{cases} x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + \dots + a_{1m}^{(1)} \cdot x_m = f_1^{(1)} \\ \quad x_2 + a_{23}^{(2)} \cdot x_3 + \dots + a_{2m}^{(2)} \cdot x_m = f_2^{(2)} \\ \quad \quad x_3 + \dots + a_{3m}^{(3)} \cdot x_m = f_3^{(3)} \\ \quad \quad \quad \dots \quad \quad \quad \dots \\ \quad \quad \quad \quad \quad \quad \quad x_m = f_m^{(m)} \end{cases}$$

Из нее в обратном порядке находим все значения x_i :

$$\begin{cases} x_m = f_m^{(m)} \\ x_{m-1} = f_{m-1}^{(m-1)} - a_{m-1m}^{(m-1)} \cdot x_m \\ \dots \quad \dots \quad \dots \\ x_1 = f_1^{(1)} - a_{12}^{(1)} \cdot x_2 - a_{13}^{(1)} \cdot x_3 - \dots - a_{1m}^{(1)} \cdot x_m \end{cases}$$

Процесс приведения к системе с треугольной матрицей называется прямым ходом, а нахождения неизвестных – обратным. В случае если один из ведущих элементов равен нулю, изложенный алгоритм метода Гаусса неприменим. Кроме того, если какие-либо ведущие элементы малы, то это приводит к увеличению ошибок округления и ухудшению точности счета. Поэтому обычно используется другой вариант метода Гаусса – схема Гаусса с выбором главного элемента. Путем перестановки строк, а также столбцов с соответствующей перенумерацией коэффициентов и неизвестных, добиваются выполнения условия:

$$|a_{ii}^{(0)}| \geq |a_{ij}^{(0)}|, \quad i, j = 1, 2, \dots, m,$$

т.е. осуществляется выбор первого главного элемента. Разделив первую строку на главный элемент, как и прежде, исключают x_1 из остальных уравнений. Затем для оставшихся столбцов и строк выбирают второй главный элемент и т.д.

Метод Гаусса – Жордано использует аналогичный прием с исключением элементов в столбцах матрицы таким образом, чтобы перевести исходную систему $A\vec{x} = \vec{f}$ к $A'\vec{x} = \vec{f}'$, где A' – диагональная матрица. Этот метод можно использовать для нахождения обратной матрицы, для чего в расширенную матрицу надо включить единичную матрицу. Преобразования строк,

приводящие исходную матрицу к диагональному виду, будут переводить единичную матрицу в обратную, а вектор правых частей – в решение СЛАУ.

Пример 2.1. Решить СЛАУ

$$\begin{cases} x_1 + x_2 - x_3 = 2 \\ -2x_1 + x_2 + x_3 = 3 \\ x_1 + x_2 + x_3 = 6 \end{cases}$$

методом Гаусса – Жордано и найти обратную матрицу. Записываем расширенную матрицу и с помощью эквивалентных преобразований переводим матрицу системы в единичную:

A				I		f
1	1	-1	1	0	0	2
-2	1	1	0	1	0	3
1	1	1	0	0	1	6

Прибавляем ко второй строке первую, умноженную на 2, а из третьей вычитаем первую:

1	1	-1	1	0	0	2
0	3	-1	2	1	0	7
0	0	2	-1	0	1	4

Делим вторую строку на 3 и вычитаем ее из первой:

1	0	$-\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{3}$	0	$-\frac{1}{3}$
0	1	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	0	$\frac{7}{3}$
0	0	2	-1	0	1	4

Делим последнюю строку на 2 и прибавляем ее с коэффициентом $\frac{1}{3}$ к первой строке и с коэффициентом $-\frac{1}{3}$ – ко второй:

1	0	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	1
0	1	0	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{6}$	3
0	0	1	$-\frac{1}{2}$	0	$\frac{1}{2}$	2

I A^{-1} $\bar{x}$

Выполняя проверку $A \cdot A^{-1} = I, A\bar{x} = \bar{f}$, убеждаемся, что решение найдено правильно.

Часто возникает необходимость в решении СЛАУ, матрицы которых являются слабо заполненными, т.е. содержат много нулевых элементов. В то же время эти матрицы имеют определенную структуру. Среди таких систем выделим системы с матрицами ленточной структуры, в которых ненулевые элементы располагаются на главной диагонали и на нескольких побочных диагоналях. Для решения систем с ленточными матрицами коэффициентов вместо метода Гаусса можно использовать более эффективные методы, например, **метод прогонки**.

Рассмотрим наиболее простой случай: систему с трехдиагональной матрицей коэффициентов, к которой сводится решение ряда численных задач (сплайн-интерполяция таблично заданной функции, дискретизация краевых задач для дифференциальных уравнений методами конечных разностей и др.). В этом случае СЛАУ имеет вид:

$$\begin{cases} -c_1 x_1 + b_1 x_2 = f_1 \end{cases} \quad (2.6)$$

$$\begin{cases} a_i x_{i-1} - c_i x_i + b_i x_{i+1} = f_i, i = 2, 3, \dots, m-1 \end{cases} \quad (2.7)$$

$$\begin{cases} a_m x_{m-1} - c_m x_m = f_m \end{cases} \quad (2.8)$$

Матрица системы (2.6)–(2.8) имеет трехдиагональную структуру, что хорошо видно из следующего эквивалентного векторно-матричного представления:

$$\begin{bmatrix} -c_1 & b_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & -c_2 & b_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & -c_3 & b_3 & 0 & \dots & 0 & 0 \\ 0 & 0 & a_4 & -c_4 & b_4 & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & a_{m-1} & -c_{m-1} & b_{m-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & a_m & -c_m \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{m-1} \\ f_m \end{bmatrix}.$$

Если при этом выполняется условие $|c_i| \geq |b_i| + |a_i|$, то говорят, что матрица данной системы имеет диагональное преобладание.

Предположим, что существуют такие наборы чисел α_i и β_i , $i = 2, 3, \dots, m$, при которых

$$x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}. \quad (2.9)$$

Записав уравнение (1.6) в виде (2.9):

$$x_1 = \frac{b_1}{c_1}x_2 - \frac{f_1}{c_1},$$

получим формулы для определения α_2, β_2 :

$$\alpha_2 = \frac{b_1}{c_1}, \quad \beta_2 = -\frac{f_1}{c_1}. \quad (2.10)$$

Уменьшим в (2.9) индекс на единицу: $x_{i-1} = \alpha_i x_i + \beta_i$, и подставим полученное выражение в (2.7):

$$a_i \alpha_i x_i + a_i \beta_i - c_i x_i + b_i x_{i+1} = f_i,$$

откуда

$$x_i = \frac{b_i}{c_i - a_i \alpha_i} x_{i+1} + \frac{a_i \beta_i - f_i}{c_i - a_i \alpha_i}.$$

Данное равенство совпадает с (2.9), если при всех $i = 1, 2, \dots, m-1$ выполняются рекуррентные соотношения

$$\alpha_{i+1} = \frac{b_i}{c_i - a_i \alpha_i}, \quad \beta_{i+1} = \frac{a_i \beta_i - f_i}{c_i - a_i \alpha_i}. \quad (2.11)$$

Они позволяют получить все остальные коэффициенты α_i, β_i .

При $i = m-1$ из (2.9) получим $x_{m-1} = \alpha_m x_m + \beta_m$. Подставляя это выражение в (2.8) и разрешая полученное выражение относительно x_m , записываем:

$$x_m = \frac{a_m \beta_m - f_m}{c_m - a_m \alpha_m}, \quad (2.12)$$

где α_m и β_m известны. Далее по формулам (2.9) последовательно находятся $x_{m-1}, x_{m-2}, \dots, x_1$.

Для успешного применения метода прогонки нужно, чтобы в процессе вычислений не возникало ситуаций с делением на нуль, а при больших размерностях систем не должно быть быстрого роста погрешностей округления.

Будем называть прогонку корректной, если знаменатели прогоночных коэффициентов в формуле (2.11) не обращаются в нуль, и устойчивой, если $|\alpha_i| < 1$ при $i = 1, 2, 3, \dots, m$.

Теорема. Пусть коэффициенты a_i, b_i уравнения (2.7) отличны от нуля и пусть $|c_i| \geq |b_i| + |a_i|$ при $i = 1, 2, 3, \dots, m$. Тогда прогонка корректна и устойчива.

Условия этой теоремы, которые во многих приложениях выполняются автоматически, являются достаточными условиями корректности и устойчивости прогонки. Если эти условия не выполняются, то можно организовать выбор главного элемента аналогично схеме Гаусса.

2.6. Итерационные методы решения СЛАУ

Рассмотрим систему линейных алгебраических уравнений (2.5). Итерационные методы, или методы последовательных приближений, дают возможность построить последовательность векторов $\vec{x}^{(0)}, \vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(k)}, \dots$, пределом которой должно быть точное решение $\vec{x}^* = \lim_{k \rightarrow \infty} \vec{x}^{(k)}$.

На практике построение последовательности обрывается, как только достигается желаемая точность. Чаще всего для достаточно малого значения $\varepsilon > 0$ контролируется выполнение оценки $|\vec{x}^* - \vec{x}^{(k)}| < \varepsilon$.

Метод последовательных приближений может быть построен по следующей схеме. Эквивалентными преобразованиями приведем систему (2.5) к виду

$$\vec{x} = C \vec{x} + \vec{d}, \quad (2.13)$$

где $\vec{x}$ – тот же самый вектор, а C и $\vec{d}$ – некоторые новые матрица и вектор соответственно.

При решении методом последовательных приближений необходимо выбрать начальное (нулевое) приближение. За нулевое приближение можно принять столбцы правых частей $\vec{f}, \vec{d}$

системы (2.13) или нулевой вектор. Следующее приближение $\bar{x}^{(1)}$ определяется рекуррентным равенством

$$\bar{x}^{(1)} = C \bar{x}^{(0)} + \vec{d}.$$

Далее находим $\bar{x}^{(2)}$:

$$\bar{x}^{(2)} = C \bar{x}^{(1)} + \vec{d},$$

и т.д. Для k -й итерации получаем

$$\bar{x}^{(k+1)} = C \bar{x}^{(k)} + \vec{d}, \quad k=0, 1, 2, \dots \quad (2.14)$$

Такой итерационный процесс будем называть **одношаговым итерационным методом**.

Изучим вопрос о сходимости итерационного процесса, т.е. определим, какие нужно предъявить требования к виду матрицы C , чтобы последовательность $\{\bar{x}^{(k)}\}$ при $k \rightarrow \infty$ имела пределом $\bar{x}^*$, т.е. была решением системы (2.13), эквивалентной исходной системе (2.5): $\lim_{k \rightarrow \infty} \bar{x}^{(k)} = \bar{x}^*$.

Достаточным условием сходимости итерационного метода (2.13) к решению системы (2.5) при любом начальном векторе $\bar{x}^{(0)}$ является требование $\|C\| < 1$, где $\|C\|$ – норма матрицы C .

В общем виде одношаговый итерационный метод можно записать в так называемой канонической форме:

$$B^{(k+1)} \frac{\bar{x}^{(k+1)} - \bar{x}^{(k)}}{\tau^{(k+1)}} + A\bar{x}^{(k)} = \vec{f}.$$

Здесь $B^{(k+1)}$ – матрица, задающая итерационный метод, τ^{k+1} – итерационный параметр. Если $B^{(k+1)} = E$ (где E – единичная матрица), то метод называют **явным**, а в противном случае – **неявным**. Если матрица $B^{(k+1)}$ и итерационный параметр $\tau^{(k+1)}$ не зависят от номера итерации ($B^{(k+1)} = B$, $\tau^{(k+1)} = \tau$), то метод называют **стационарным**, и **нестационарным** – в противном случае.

Использование неявных методов сопровождается обращением матрицы B^{k+1} , поэтому для сохранения эффективности алго-

ритма эта матрица должна быть легко обратима (например, B^{k+1} – диагональная, треугольная, трехдиагональная или ортогональная).

Приведем некоторые **примеры**. Для этого представим матрицу A в виде $A = A_1 + D + A_2$, где D – диагональная матрица, A_1 – левая треугольная, A_2 – правая треугольная. A_1 и A_2 имеют нулевую главную диагональ. Ненулевые элементы всех трех матриц совпадают с соответствующими элементами матрицы A .

- ✓ *Метод релаксации (простой итерации).* Здесь $B^{(k+1)} = E$, а $\tau^{(k+1)} = \tau$.
- ✓ *Итерационный метод Рундсона.* Здесь $B^{(k+1)} = E$, а $\tau^{(k+1)}$ – переменный параметр.
- ✓ *Метод Якоби.* $B^{(k+1)} = D$, а $\tau^{(k+1)} = 1$.
- ✓ *Метод верхней релаксации.* $B^{(k+1)} = D + \omega A_1$, $\tau^{k+1} = \omega$, $0 < \omega < 2$, где ω – заданный числовой параметр. Для $\omega = 1$ как частный случай получается *метод Гаусса – Зейделя*. Для симметричных положительно определенных матриц A условие $0 < \omega < 2$ является условием сходимости метода.

Для некоторых из вышеназванных методов рассмотрим более детально способы получения матрицы C .

Предположим, что диагональные элементы матрицы A исходной системы (2.5) не равны 0 ($a_{ii} \neq 0, i = 1, 2, \dots, m$). Разрешим первое уравнение системы (2.5) относительно x_1 , второе относительно x_2 и т.д. Получим следующую эквивалентную систему, записанную в скалярном виде:

$$\left\{ \begin{array}{l} x_1 = C_{12}x_2 + C_{13}x_3 + ... + C_{1n}x_n + d_1 \\ x_2 = C_{21}x_1 + C_{23}x_3 + ... + C_{2n}x_n + d_2 \\ \\ x_n = C_{n1}x_1 + C_{n2}x_2 + ... + C_{n,n-1}x_{n-1} + d_n \end{array} \right.$$

что совпадает с формулой (2.13), в которой матрица C и вектор $\vec{d}$ определены по следующим формулам:

$$C_{ij} = \begin{cases} -\frac{a_{ij}}{a_{ii}}, & i \neq j, \\ 0, & i = j \end{cases}, \quad d_i = \frac{f_i}{a_{ii}}, \quad i = 1, 2, \dots, n. \quad (2.15)$$

Метод, основанный на таком приведении системы (2.5) к виду (2.13), называют **методом Якоби**. Теперь, задав нулевое приближение, по рекуррентным соотношениям (2.14) можем выполнять итерационный процесс.

Условие сходимости $\|C\| < 1$ в методе Якоби равносильно условию диагонального преобладания для исходной матрицы A :

$$|a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^m |a_{ij}|, \quad i = 1, 2, \dots, m.$$

Действительно, пусть для матрицы A выполняется условие диагонального преобладания. Разделим обе части данного неравенства на $|a_{ii}|$, получим неравенство

$$1 > \sum_{\substack{j=1 \\ i \neq j}}^m \frac{|a_{ij}|}{|a_{ii}|}, \quad i = 1, 2, \dots, m.$$

С учетом (2.15) можно перейти к следующему неравенству:

$$1 > \sum_{j=1}^m |C_{ij}|, \quad i = 1, 2, \dots, m.$$

То есть сумма элементов любой строки матрицы C меньше 1. Выполнение такого условия равносильно выполнению условия

сходимости метода $\|C\| = \max_{1 \leq i \leq m} \sum_{j=1}^m |C_{ij}| < 1$.

Под методом **Гаусса – Зейделя** обычно понимается такое видоизменение одношагового итерационного метода (2.15) решения СЛАУ, в котором для подсчета i -й компоненты $(k+1)$ -го приближения $x_i^{(k+1)}$ к искомому вектору $\vec{x}^*$ используются уже вычисленные на этом, т.е. $(k+1)$ -м шаге, значения первых $i-1$ компонент $x_{i-1}^{(k+1)}$. Это означает, что если система (2.5) тем или иным способом сведена (например, с помощью метода Якоби) к

точностью. Поэтому важной проблемой является вопрос о способе остановки итерационного процесса при достижении точности. Наиболее простой способ – это сравнение между собой соответствующих неизвестных с двух соседних итераций: $(k + 1)$ и (k) . Если максимальная из всех разностей становится меньше заданной точности ε , то итерационный процесс останавливается:

$$\max_{1 \leq i \leq m} |x_i^k - x_i^{k+1}| < \varepsilon.$$

2.7. Решение СЛАУ в пакете MathCAD

Описанные выше алгоритмы решения СЛАУ реализованы в MathCAD в виде функции `lsolve`, основанной на прямом методе Гаусса, и в виде вычислительного блока `Given/Find`, в основе которого лежит итерационный алгоритм.

Для применения функции `lsolve` система должна быть записана в матричном виде (2.5"). Пример использования функции `lsolve` показан на рис. 2.4.

Использование блока `Given/Find` может быть связано как со скалярной, так и с матричной формой записи СЛАУ. Как уже было отмечено, данный метод основан на итерационной схеме решения, поэтому всем неизвестным сначала должны быть присвоены начальные значения. В случае если СЛАУ имеет единственное решение, выбор начального приближения не влияет на решение. После этого необходимо привести ключевое слово `Given` (Дано) и записать СЛАУ в виде логического выражения, т.е. с использованием знака логического равенства `<=>`, который можно найти на панели инструментов `Boolean` (Логическая). Затем вызвать встроенную функцию `Find`, аргументами которой являются искомые величины. На рис. 2.5 показано решение СЛАУ, записанной в матричной форме, с помощью вычислительного блока `Given/Find`.

Используя матричные функции, описанные выше, СЛАУ небольшой размерности можно решать с помощью метода обратной матрицы и метода Крамера. Соответствующие примеры показаны на рис. 2.6 и 2.7.

$$A := \begin{pmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ 7 & 8 & 9 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix} \quad x := \text{lsolve}(A, b) = \begin{pmatrix} 0.2 \\ 0.6 \\ -0.133 \end{pmatrix} \quad A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Рис. 2.4. Решение СЛАУ с помощью функции lsolve

Исходные данные: $A := \begin{pmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ 7 & 8 & 9 \end{pmatrix}$ $b := \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}$ Начальное приближение: $x := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

Given $A \cdot x = b$ Find(x) = $\begin{pmatrix} 0.2 \\ 0.6 \\ -0.133 \end{pmatrix}$



Рис. 2.5. Решение СЛАУ с помощью блока Given/Find

Решение СЛАУ методом обратной матрицы

$$A := \begin{pmatrix} 5 & 2 & 3 \\ 2 & 6 & 1 \\ 3 & 1 & 7 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad x := A^{-1} \cdot b = \begin{pmatrix} 0.104 \\ 0.119 \\ 0.081 \end{pmatrix} \quad A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Рис. 2.6. Решение СЛАУ методом обратной матрицы

Решение СЛАУ методом Крамера

$$A := \begin{pmatrix} 5 & 2 & 3 \\ 2 & 6 & 1 \\ 3 & 1 & 7 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad A1 := \begin{pmatrix} 1 & 2 & 3 \\ 1 & 6 & 1 \\ 1 & 1 & 7 \end{pmatrix} \quad A2 := \begin{pmatrix} 5 & 1 & 3 \\ 2 & 1 & 1 \\ 3 & 1 & 7 \end{pmatrix} \quad A3 := \begin{pmatrix} 5 & 2 & 1 \\ 2 & 6 & 1 \\ 3 & 1 & 1 \end{pmatrix}$$

$$\det := |A| \quad \det1 := |A1| \quad \det2 := |A2| \quad \det3 := |A3|$$

$$x := \begin{pmatrix} \det1 \\ \det2 \\ \det3 \end{pmatrix} \cdot \frac{1}{\det} = \begin{pmatrix} 0.104 \\ 0.119 \\ 0.081 \end{pmatrix} \quad A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$


Рис. 2.7. Решение СЛАУ методом Крамера

В некоторых случаях для решения СЛАУ можно использовать пользовательские процедуры-функции, реализующие прямые или итерационные методы решения СЛАУ. Ниже показаны фрагменты рабочих листов MathCAD, на которых приведены примеры решения СЛАУ с трехдиагональной матрицей методом прогонки (рис. 2.8) и итерационным методом Якоби (рис. 2.9). Для того чтобы использовать формулы (2.9) – (2.12), в программе установлено начальное значение индекса матричных переменных $ORIGIN = 1$ (по умолчанию, значение $ORIGIN = 0$).

The screenshot shows a MathCAD worksheet with the following content:

Function Definition:

$$\text{prog}(a, b, c, f) := \begin{cases} n \leftarrow \text{length}(f) \\ \text{alf}_2 \leftarrow \frac{b_1}{c_1} \\ \text{bet}_2 \leftarrow \frac{-f_1}{c_1} \\ \text{for } i \in 2..n-1 \\ \quad \left| \begin{aligned} \text{alf}_{i+1} &\leftarrow \frac{b_i}{c_i - a_i \cdot \text{alf}_i} \\ \text{bet}_{i+1} &\leftarrow \frac{a_i \cdot \text{bet}_i - f_i}{c_i - a_i \cdot \text{alf}_i} \end{aligned} \right. \\ x_n \leftarrow \frac{a_n \cdot \text{bet}_n - f_n}{c_n - a_n \cdot \text{alf}_n} \\ \text{for } i \in n-1..1 \\ \quad x_i \leftarrow \text{alf}_{i+1} \cdot x_{i+1} + \text{bet}_{i+1} \\ x \end{cases}$$

Initial Value:

$$ORIGIN := 1$$

Matrix and Vector Definitions:

$$A := \begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 0 \\ 0 & 2 & -6 & -2 & 0 \\ 0 & 0 & 1 & 5 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad f := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Intermediate Calculations:

$$a := \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \\ 1 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 1 \\ -2 \\ 3 \\ 0 \end{pmatrix} \quad \underline{c}_n := \begin{pmatrix} -2 \\ 4 \\ 6 \\ -5 \\ -1 \end{pmatrix}$$

Final Calculation:

$$x := \text{prog}(a, b, c, f)$$

Result:

$$x = \begin{pmatrix} 0.488 \\ 0.024 \\ 1.61 \\ -6.303 \\ 11.305 \end{pmatrix} \quad A \cdot x - f = \begin{pmatrix} 0 \\ 0 \\ 1.776 \times 10^{-15} \\ 0 \\ 0 \end{pmatrix}$$

Рис. 2.8. Решение СЛАУ методом прогонки

$$\mathbf{A} = \begin{pmatrix} 8 & 4 & 2 \\ 3 & 5 & 1 \\ 3 & -2 & 10 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 10 \\ 5 \\ 4 \end{pmatrix},$$

$$\alpha := \begin{pmatrix} 0 & \frac{-1}{2} & \frac{-1}{4} \\ \frac{-3}{5} & 0 & \frac{-1}{5} \\ \frac{-3}{10} & \frac{1}{5} & 0 \end{pmatrix}, \quad \beta := \begin{pmatrix} \frac{10}{8} \\ 1 \\ \frac{4}{10} \end{pmatrix}$$

```

yakobi(ε) :=
| err ← 100
| x ← β
| while |err| > ε
|   | x1 ← α · x + β
|   | err ← x1 - x
|   | x ← x1
| x

```

$$\text{yakobi}(0.001) = \mathbf{x}$$

Рис. 2.9. Решение СЛАУ итерационным методом Якоби

2.8. Собственные значения и собственные векторы

Пусть A – матрица размерности $n \times n$. Числовая константа λ называется собственным значением, а ненулевой вектор $\bar{x}$ – соответствующим λ собственным вектором матрицы A , если

$$A\bar{x} = \lambda\bar{x}. \quad (2.17)$$

Равенство (2.17) можно записать также в виде

$$B\bar{x} = (A - \lambda I)\bar{x} = \bar{0},$$

где I – единичный (тождественный оператор). Последнее равенство означает, что вектор $\bar{x}$ принадлежит ядру линейного оператора $B = A - \lambda I$. Чтобы найти ненулевое решение, необходимо, чтобы определитель матрицы был равен нулю:

$$\det(A - \lambda I) = 0. \quad (2.18)$$

Следовательно, чтобы найти собственные значения и собственные векторы матрицы, необходимо решить **характеристическое (вековое) уравнение** (2.18), а затем подставить полученные значения λ в уравнение (2.17) и, решив систему линейных алгебраических уравнений (СЛАУ), найти компоненты собственного вектора. Поскольку матрица СЛАУ вырождена, ее собственный вектор находится с точностью до константы.

При нахождении собственных значений могут возникнуть различные ситуации, например, вековое уравнение не имеет вещественных корней, или некоторые из корней кратные. Кратным собственным значениям может соответствовать один, два или больше собственных векторов. Рассмотрим несколько примеров.

Пример 2.2. Найдем собственные значения матрицы

$$A = \begin{pmatrix} 5 & 2 \\ 3 & 6 \end{pmatrix}.$$

Запишем вековое уравнение:

$$\det(A - \lambda I) = \begin{vmatrix} 5-\lambda & 2 \\ 3 & 6-\lambda \end{vmatrix} = (5-\lambda) \cdot (6-\lambda) - 6 = -\lambda^2 + 11\lambda + 24 = 0.$$

Решая квадратное уравнение, получим

$$\lambda_{1,2} = \frac{11 \pm \sqrt{121 - 4 \cdot 24}}{2} = \frac{11 \pm 5}{2}; \quad \lambda_1 = 8, \quad \lambda_2 = 3.$$

Подставляем λ_1 в (1.3), получим СЛАУ

$$\begin{pmatrix} 5-8 & 2 \\ 3 & 6-8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{cases} -3x_1 + 2x_2 = 0 \\ 3x_1 - 2x_2 = 0 \end{cases},$$

решением которой будут векторы, где $3x_1 = 2x_2$, например, $\bar{x} = (2, 3)$. Найдем второй собственный вектор $\bar{y}$, отвечающий собственному значению $\lambda = 3$:

$$\begin{pmatrix} 5-3 & 2 \\ 3 & 6-3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{cases} 2y_1 + 2y_2 = 0 \\ 3y_1 + 3y_2 = 0 \end{cases} \Rightarrow y_1 = -y_2.$$

Вторым собственным вектором будет $\bar{y} = (1, -1)$.

Пример 2.3. Пусть $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. Вековое уравнение

$(1-\lambda)^2 = 0$, значит, $\lambda_1 = \lambda_2 = 1$. Ищем собственный вектор:

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

откуда $x_2 = 0, x_1$ — произвольно, например, $x_1 = 1$. Второго собственного вектора нет.

Пример 2.4. Найти собственные значения и собственные векторы матрицы $A = \begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix}$.

$$\det(A - \lambda I) = 0 \Rightarrow \begin{vmatrix} 1-\lambda & -3 & 3 \\ 3 & -5-\lambda & 3 \\ 6 & -6 & 4-\lambda \end{vmatrix} = 0.$$

Раскладываем определитель по первой строке:

$$\begin{aligned} & (1-\lambda) \cdot \begin{vmatrix} -5-\lambda & 3 \\ -6 & 4-\lambda \end{vmatrix} - (-3) \cdot \begin{vmatrix} 3 & 3 \\ 6 & 4-\lambda \end{vmatrix} + 3 \cdot \begin{vmatrix} 3 & -5-\lambda \\ 6 & -6 \end{vmatrix} = \\ & = (1-\lambda)[(-5-\lambda)(4-\lambda) + 18] + 3[3(4-\lambda) - 18] + 3[6(5+\lambda) - 18] = \\ & = (1-\lambda) \cdot [-2 + \lambda - \lambda^2] + 3 \cdot [-6 - 3\lambda] + 3 \cdot [12 + 6\lambda] = \\ & = -2 + 2\lambda + \lambda - \lambda^2 + \lambda^2 - \lambda^3 - 18 - 9\lambda + 36 + 18\lambda = \\ & = 16 + 12\lambda - \lambda^3 = 0 \end{aligned}$$

Ищем целочисленные решения кубического уравнения, проверяя значения $\lambda = \pm 1, \pm 2, \dots$. В результате подстановки находим первый корень $\lambda_1 = -2$. Выделяя множитель $(\lambda + 2)$ из кубического полинома, получим квадратное уравнение $\lambda^2 - 2\lambda - 8 = 0$ с действительными корнями $\lambda_2 = -2$, $\lambda_3 = 4$. Обозначим собственные векторы, отвечающие найденным собственным значениям $\lambda_1 = \lambda_2 = -2$, $\lambda_3 = 4$, как $\bar{x}, \bar{y}, \bar{z}$ соответственно.

Подставляя $\lambda_3 = 4$ в (2.18), получим систему для определения собственного вектора $\bar{z}$:

$$\begin{cases} -3z_1 - 3z_2 + 3z_3 = 0 \\ 3z_1 - 9z_2 + 3z_3 = 0, \\ 6z_1 - 6z_2 = 0 \end{cases}$$

в которой после несложных преобразований останется два независимых уравнения:

$$\begin{cases} z_1 - z_2 = 0 \\ -2z_2 + z_3 = 0 \end{cases}.$$

Полагая $z_2 = 1$, получим собственный вектор $\bar{z} = (1, 1, 2)$.

Ищем собственные векторы, соответствующие кратному собственному значению $\lambda_1 = \lambda_2 = -2$. После преобразований остается одно независимое уравнение $x_1 - x_2 + x_3 = 0$, которое задает целую плоскость собственных векторов. Полагая $x_2 = 1, x_3 = 0$, $y_2 = 0, y_3 = 1$, получим два собственных вектора $\bar{x} = (1, 1, 0)$ и $\bar{y} = (-1, 0, 1)$.

Если матрица имеет полную систему собственных векторов, то она может быть приведена к диагональному виду, причем на диагонали будут стоять собственные значения матрицы. Пусть A – исходная матрица, имеющая полную систему собственных векторов. Преобразуем исходную матрицу по формуле

$$A' = C^{-1} \cdot A \cdot C, \quad (2.19)$$

где C – матрица, составленная из собственных векторов матрицы A , C^{-1} – обратная к C матрица.

Пример 2.5. Приведем к диагональному виду матрицу из Примера 2.2. Запишем найденные собственные векторы в матрицу преобразования $C = \begin{pmatrix} 2 & 1 \\ 3 & -1 \end{pmatrix}$ и найдем обратную

$$C^{-1} = \frac{1}{5} \begin{pmatrix} 1 & 1 \\ 3 & -2 \end{pmatrix}.$$

$$\text{Проверка: } C \cdot C^{-1} = \frac{1}{5} \begin{pmatrix} 2+3 & 2-2 \\ 3-3 & 3+2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I.$$

Применяя (2.19), получим

$$A' = C^{-1} \cdot A \cdot C = \frac{1}{5} \begin{pmatrix} 2 & 2 \\ 3 & -4 \end{pmatrix} \cdot \begin{pmatrix} 5 & 2 \\ 3 & 6 \end{pmatrix} \cdot \begin{pmatrix} 2 & 1 \\ 3 & -1 \end{pmatrix} = \begin{pmatrix} 8 & 0 \\ 0 & 3 \end{pmatrix},$$

т.е. в результате получена диагональная матрица, на главной диагонали которой стоят собственные значения.

Пример 2.6. Для матрицы из Примера 2.4 найдены собственные векторы $\bar{x} = (1, 1, 0)^T$, $\bar{y} = (-1, 0, 1)^T$, $\bar{z} = (1, 1, 2)^T$, из которых образуем матрицу C и найдем обратную к ней:

$$C = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \quad C^{-1} = \frac{1}{2} \begin{pmatrix} -1 & 3 & -1 \\ -2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}.$$

(Проверить, что $C \cdot C^{-1} = C^{-1} \cdot C = I$.)

Приводим к диагональному виду: $A' = C^{-1} \cdot A \cdot C =$

$$= \frac{1}{2} \begin{pmatrix} -1 & 3 & -1 \\ -2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & -3 & 1 \\ 2 & -2 & 0 \\ 2 & -2 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 4 \end{pmatrix}.$$

В MathCAD для нахождения собственных значений и собственных векторов используются функции `eigenvals`, `eigenvecs`, `eigenvec`.

Функция `eigenvals(A)` вычисляет вектор, компонентами которого являются собственные значения квадратной матрицы A . Функция `eigenvecs(A)` вычисляет матрицу, столбцы которой являются нормированными собственными векторами матрицы A . Функция `eigenvec(A, λ)` вычисляет нормированный собственный вектор матрицы A , соответствующий собственному значению λ . На рисунке 2.10 приведены фрагменты рабочего листа MathCAD, на котором найдены собственные значения и собственные вектора для матриц из примеров 2.2 и 2.4. Как уже отмечено выше, значения собственных векторов определяются с точностью до константы, поэтому в MathCAD получены так называемые нормированные собственные вектора, у которых норма (евклидова длина) равна единице: $\|x\| = 1$. Найденные собственные вектора использованы для приведения исходной матрицы к диагональному виду.

Для нахождения собственных значений можно использовать инструменты MathCAD, позволяющие проводить символьные вычисления. На рис. 2.11 показан фрагмент рабочего листа, на котором вековые уравнения для примеров 2.2. и 2.4. получены в символьном виде и далее решены с помощью стандартной функции MathCAD `polyroots`, позволяющей находить корни полинома по вектору его коэффициентов.

Пример 2.2.

$$A := \begin{pmatrix} 5 & 2 \\ 3 & 6 \end{pmatrix} \quad \text{eigenvals}(A) = \begin{pmatrix} 3 \\ 8 \end{pmatrix} \quad \text{eigenvec}(A, 3) = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix} \quad \text{eigenvec}(A, 8) = \begin{pmatrix} 0.555 \\ 0.832 \end{pmatrix}$$

$$C := \text{eigenvecs}(A) = \begin{pmatrix} -0.707 & 0.555 \\ 0.707 & 0.832 \end{pmatrix} \quad C^{-1} \cdot A \cdot C = \begin{pmatrix} 3 & 0 \\ 0 & 8 \end{pmatrix} +$$

Пример 2.4.

$$B := \begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix} \quad \text{eigenvals}(B) = \begin{pmatrix} 4 \\ -2 \\ -2 \end{pmatrix} \quad \text{eigenvec}(B, 4) = \begin{pmatrix} 0.408 \\ 0.408 \\ 0.816 \end{pmatrix} \quad \text{eigenvec}(B, -2) = \begin{pmatrix} -0.713 \\ -0.011 \\ 0.701 \end{pmatrix}$$

$$D := \text{eigenvecs}(B) = \begin{pmatrix} 0.408 & 0.408 & -0.12 \\ 0.408 & -0.408 & -0.76 \\ 0.816 & -0.816 & -0.639 \end{pmatrix} \quad D^{-1} \cdot B \cdot D = \begin{pmatrix} 4 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

Рис. 2.10. Листинг MathCAD с примерами нахождения собственных значений, собственных векторов матриц и приведения матрицы к диагональному виду

$$\left| \begin{pmatrix} 5-\lambda & 2 \\ 3 & 6-\lambda \end{pmatrix} \right| \rightarrow \lambda^2 - 11\lambda + 24 \quad v := \begin{pmatrix} 24 \\ -11 \\ 1 \end{pmatrix} \quad \text{polyroots}(v) = \begin{pmatrix} 3 \\ 8 \end{pmatrix} +$$

$$\left| \begin{pmatrix} 1-\lambda & -3 & 3 \\ 3 & -5-\lambda & 3 \\ 6 & -6 & 4-\lambda \end{pmatrix} \right| \rightarrow 12\lambda - \lambda^3 + 16 \quad u := \begin{pmatrix} 16 \\ 12 \\ 0 \\ -1 \end{pmatrix} \quad \text{polyroots}(u) = \begin{pmatrix} -2 \\ -2 \\ 4 \end{pmatrix}$$

Рис. 2.11. Листинг MathCAD с примерами получения векового уравнения в символьном виде

Глава 3. Интерполяция и приближение функций

Приближение функций – это раздел теории функций, посвященный изучению вопросов приближённого представления функций. Приближение позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются или свойства которых уже известны). Приближение функций — нахождение для данной функции f другой функции F из некоторого определённого класса (например, среди алгебраических многочленов заданной степени), в том или ином смысле близкой к f , дающей её приближённое представление.

Интерполирование функций — частный случай задачи приближения, когда требуется, чтобы в определённых точках (узлах интерполирования) совпадали значения функции f и приближающей её функции F , а в более общем случае — и значения некоторых их производных. Слово «интерполяция» в переводе означает «между точками». Задачи интерполяции часто возникают в инженерных и других практических приложениях. Допустим, что в результате экспериментальных измерений получена таблица значений некоторой функции. Требуется найти промежуточные значения этой функции, а также производные, определяющие скорость ее изменения. Это так называемая *задача о восстановлении функции*. Кроме того, при проведении расчетов сложные функции удобно заменять алгебраическими многочленами или другими элементарными функциями, которые достаточно просто вычисляются (*задача о приближении функции*). Интерполяцию используют для приближенного вычисления интегралов (*построение квадратурных формул*). Из математического анализа известны, например, многочлены (ряды) Тейлора, которые используются для вычисления значений гладких (т.е. достаточное число раз дифференцируемые) функций. В математике и технике часто используют разложение функций в тригонометрические ряды. Каждый метод имеет свою погрешность, определяемую тем, насколько различаются значения исходной и интерполирующей функций. Существуют

ли другие способы интерполяции и приближения функций? Когда и какой способ лучше использовать? Какова точность (погрешность) используемых методов интерполяции? Об этом мы узнаем, изучив следующую тему.

3.1. Постановка задачи интерполяции

На интервале $[a, b]$ задана система узлов интерполяции x_i , $i=0, 1, \dots, N$; $a \leq x_i \leq b$, и значения неизвестной функции в этих узлах f_i , $i=0, 1, \dots, N$. Могут быть поставлены задачи:

1. Найти функцию $F(x)$, принимающую в точках x_i заданные значения: $F(x_i)=f_i$, $i=0, 1, \dots, N$ (условия интерполяции).
2. Для заданного значения $z \in [a, b]$ найти $F(z)$.
3. Для заданного значения $z \in [a, b]$ найти $F'(z)$.

Задача имеет много решений: через заданные точки (x_i, f_i) , $i=0, 1, \dots, N$, можно провести бесконечно много кривых, каждая из которых будет графиком функции, для которой выполнены все условия интерполяции. Если известна исходная функция $g(x)$, то можно оценить погрешность метода в произвольной точке $z \in [a, b]$: $r(z)=|g(z)-F(z)|$. Кроме того, можно оценивать так называемую равномерную r_1 и среднеквадратичную r_2 погрешности: $r_1 = \max_{z \in [a, b]} r(z)$, $r_2 = \int_a^b (r(z))^2 dz$.

Нас будет интересовать поведение погрешности метода при увеличении числа узлов интерполяции. Будем говорить, что *метод сходится*, если при $N \rightarrow \infty$ погрешность $r \rightarrow 0$.

Все методы интерполяции можно разделить на локальные и глобальные. В случае *локальной интерполяции* на каждом интервале $[x_{i-1}, x_i]$ строится своя (локальная) функция. В случае *глобальной интерполяции* отыскивается одна (глобальная) функция на всем интервале $[a, b]$. Далее приведены примеры различных способов интерполяции.

3.2. Локальная интерполяция

Кусочно–постоянная интерполяция. На каждом локальном отрезке $[x_{i-1}, x_i]$, $i=1, 2, \dots, N$, интерполирующая функция является постоянной и равна левому: $F_i(z)=f_i$ или правому: $F_i(z)=f_i$ зна-

чению. Легко понять, что условия интерполяции в этом случае выполняются. Построенная функция является разрывной (см. рис. 3.1 а), что ограничивает ее применение. Кроме того, в случае малого числа точек такая интерполяция дает большую погрешность.

Кусочно–линейная интерполяция. На каждом интервале $[x_{i-1}, x_i]$ функция является линейной $F_i(z)=k_i z+l_i$. Значения коэффициентов находятся из выполнения условий интерполяции в концах отрезка: $F_i(x_{i-1})=f_{i-1}$, $F_i(x_i)=f_i$. Получаем систему уравнений: $k_i x_{i-1}+l_i=f_{i-1}$, $k_i x_i+l_i=f_i$, откуда находим $k_i = \frac{f_i - f_{i-1}}{x_i - x_{i-1}}$,

$l_i = f_i - k_i x_i$. Итоговая функция будет непрерывной, но производная будет разрывной в каждом узле интерполяции. Погрешность такой интерполяции будет меньше, чем в предыдущем случае. Иллюстрация кусочно–линейной интерполяции приведена на рис. 3.1б.

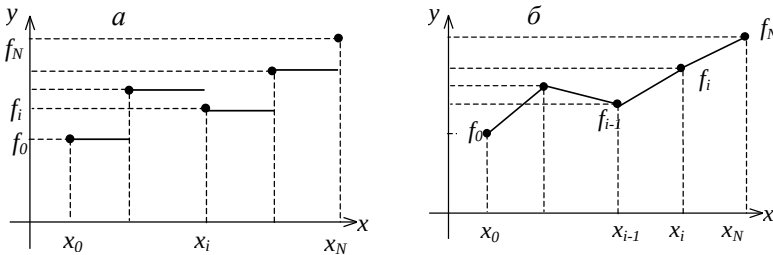


Рис. 3.1. Левая кусочно–постоянная (а) и кусочно–линейная (б) интерполяции

Пример 3.1. Заданы значений некоторой функции:

x	0	2	3	3.5
f	–1	0.2	0.5	0.8

Требуется найти значение функции при $z=1$ по кусочно–постоянной и кусочно–линейной интерполяции.

Решение. Точка $z=1$ принадлежит первому локальному отрезку $[0, 2]$, следовательно, по формулам левой кусочно–постоянной интерполяции $F(1) = f_0 = -1$, по формулам правой

кусочно–постоянной интерполяции $F(1)=f_1=0.2$. Воспользуемся формулами кусочно–линейной интерполяции:

$$k_1 = \frac{f_1 - f_0}{x_1 - x_0} = 0.6, \quad l_1 = f_1 - k_1 x_1 = -1, \quad F(1) = -1 + 0.6 \cdot 1 = -0.4.$$

Для построения линейной интерполяции в пакете MathCAD служит встроенная функция `linterp`.

- `linterp(x, f, t)` – функция, аппроксимирующая данные векторов x и f кусочно-линейной зависимостью, где:
- x – вектор действительных данных аргумента; элементы вектора должны быть определены в порядке возрастания;
- f – вектор действительных значений заданной табличной функции в узлах интерполяции x_i (размерность f должна совпадать с размерностью вектора x);
- t – значение аргумента, для которого вычисляется интерполирующая функция.

На рис. 3.2 приведен листинг программы для решения примера 3.1 с помощью данной функции.

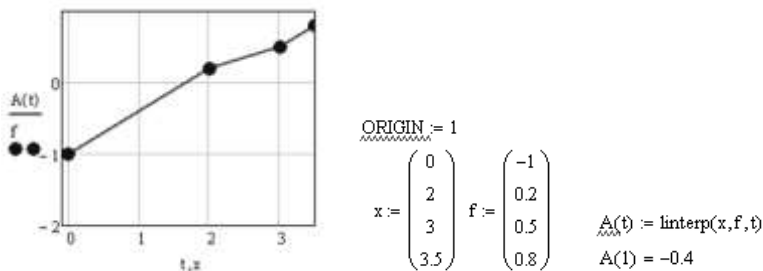


Рис. 3.2. Линейная интерполяция. Решение в MathCAD

В большинстве практических приложений желательно соединить экспериментальные точки не ломаной линией, а гладкой кривой. Лучше всего для этих целей подходит интерполяция сплайнами, т.е. отрезками полиномов (чаще всего парабол).

Кусочно–параболическая интерполяция. На каждом i -м интервале $[x_{i-1}, x_i]$, $i=1, 2, \dots, N$, функция является параболой вида $F_i(z)=a_i+b_i(z-x_i)+c_i/2(z-x_i)^2$. Значения коэффициентов a_i , b_i , c_i

находятся из условий интерполяции, непрерывности функции и ее первой производной в узлах интерполяции.

Кубический интерполяционный сплайн. Название метода происходит от английского слова *spline*. Так называлась гибкая линейка, использовавшаяся корабельными инженерами вместо лекал. Форма этого универсального лекала на каждом отрезке описывается кубической параболой. Сплайны широко используются в инженерных приложениях, в частности, в компьютерной графике. Итак, на каждом i -м $[x_{i-1}, x_i]$, $i=1, 2, \dots, N$, решение будем искать в виде полинома третьей степени:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2/2 + d_i(x - x_i)^3/6.$$

Неизвестные коэффициенты a_i, b_i, c_i, d_i , $i=1, 2, \dots, N$, находим из:

- условий интерполяции: $S_i(x_i) = f_i$, $i=1, 2, \dots, N$; $S_1(x_0) = f_0$,
- непрерывности функции $S_i(x_{i-1}) = S_{i-1}(x_{i-1})$, $i=2, 3, \dots, N$,
- непрерывности первой и второй производной:

$$S'_i(x_{i-1}) = S'_{i-1}(x_{i-1}), S''_i(x_{i-1}) = S''_{i-1}(x_{i-1}), i=2, 3, \dots, N.$$

Для определения $4N$ неизвестных получаем систему $4N-2$ уравнений:

$$\begin{aligned} a_i &= f_i, i=1, 2, \dots, N, \\ b_i h_i - c_i h_i^2/2 + d_i h_i^3/6 &= f_i - f_{i-1}, i=1, 2, \dots, N, \\ b_i - b_{i-1} &= c_i h_i - d_i h_i^2/2, i=2, 3, \dots, N, \\ d_i h_i &= c_i - c_{i-1}, i=2, 3, \dots, N. \end{aligned}$$

где $h_i = x_i - x_{i-1}$. Недостающие два уравнения выводятся из дополнительных условий: $S''(a) = S''(b) = 0$. Из системы можно исключить неизвестные b_i, d_i , получив систему $N+1$ линейных уравнений (СЛАУ) для определения коэффициентов c_i :

$$c_0 = 0, c_N = 0, \\ h_i c_{i-1} + 2(h_i + h_{i+1})c_i + h_{i+1}c_{i+1} = 6 \left(\frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}} \right), i=1, 2, \dots, N-1.$$

В случае постоянной сетки $h_i = h$ эта система уравнений упрощается. Данная СЛАУ имеет трехдиагональную матрицу и решается методом прогонки. После этого высчитываются коэффициенты

$$b_i, d_i: d_i = \frac{c_i - c_{i-1}}{h_i}, b_i = \frac{c_i h_i}{2} - \frac{d_i h_i^2}{6} + \frac{f_i - f_{i-1}}{h_i}, i=1, 2, \dots, N.$$

Для вычисления значения $S(x)$ в произвольной точке отрезка $z \in [a, b]$ необходимо решить систему уравнений на коэффициенты c_i , $i=1,2,\dots,N-1$, затем найти все коэффициенты b_i , d_i . Далее, необходимо определить, на какой интервал $[x_{i_0}, x_{i_0+1}]$ попадает эта точка, и, зная номер i_0 , вычислить значение сплайна и его производных в точке z

$$S(z) = a_{i_0} + b_{i_0}(z - x_{i_0}) + c_{i_0}(z - x_{i_0})^2/2 + d_{i_0}(z - x_{i_0})^3/6$$

$$S'(z) = b_{i_0} + c_{i_0}(z - x_{i_0}) + d_{i_0}(z - x_{i_0})^2/2, \quad S''(z) = c_{i_0} + d_{i_0}(z - x_{i_0}).$$

Кубический интерполяционный сплайн имеет достаточно хорошую точность, и в то же время имеет простую и экономичную реализацию (решение полученного СЛАУ методом прогонки).

Для построения кубической интерполяции в пакете MathCAD служит встроенная функция `interp(s, x, f, t)` - функция, аппроксимирующая данные векторов $\mathbf{x}$ и $\mathbf{f}$ кубическими сплайнами;

$\mathbf{x}$ - вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;

$\mathbf{f}$ - вектор действительных значений заданной табличной функции в узлах интерполяции $\mathbf{x}_i$ (размерность совпадает с размерностью вектора $\mathbf{x}$);

$\mathbf{t}$ - значение аргумента, для которого вычисляется интерполирующая функция.

$\mathbf{s}$ - вектор вторых производных на граничных точках $\mathbf{x}_0$ и $\mathbf{x}_N$, созданный одной из сопутствующих функций `cspline`, `pspline` или `lspline`;

`lspline(x, f)` - вектор значений коэффициентов, найденных с помощью линейного сплайна;

`pspline(x, f)` - вектор значений коэффициентов, найденных с помощью квадратичного сплайна;

`cspline(x, f)` - вектор значений коэффициентов, найденных с помощью кубического сплайна.

Пример 3.2. Задана дискретная функция, представим ее аргументы и значения в виде векторов $\mathbf{x}$ и $\mathbf{y}$. Задача - с помощью кубического сплайна найти значения заданной функции во всех точках интервала $[x_0, x_N]$. Сначала вычисляется вспомога-

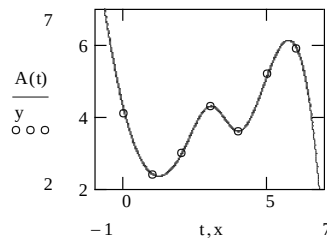
ный вектор s с помощью функции `cspline`, служащий входным параметром для функции `interp`.

$x := (0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6)^T$

$y := (4.1 \ 2.4 \ 3 \ 4.3 \ 3.6 \ 5.2 \ 5.9)^T$

$s := \text{cspline}(x, y)$

$A(t) := \text{interp}(s, x, y, t)$



3.3. Глобальная интерполяция

Будем искать интерполирующую функцию в виде полинома (многочлена) m -ой степени $P_m(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_mx^m$. Какова должна быть степень многочлена, чтобы удовлетворить всем условиям интерполяции? Допустим, что заданы две точки: (x_0, f_0) и (x_1, f_1) . Через эти точки можно провести единственную прямую, т.е. интерполирующей функцией будет полином первой степени $P_1(x) = a_0 + a_1x$. Через три точки можно провести параболу $P_2(x) = a_0 + a_1x + a_2x^2$ и т.д. Рассуждая таким способом, можно предположить, что искомый полином должен иметь степень N

Для того, чтобы доказать это, выпишем систему уравнений на коэффициенты. Уравнения системы представляют собой условия интерполяции при каждом $x = x_i$:

$$\begin{cases} P_N(x_1) = a_0 + a_1x_1 + a_2x_1^2 + a_3x_1^3 + \dots + a_Nx_1^N = f_1 \\ P_N(x_2) = a_0 + a_1x_2 + a_2x_2^2 + a_3x_2^3 + \dots + a_Nx_2^N = f_2 \\ \dots \\ P_N(x_N) = a_0 + a_1x_N + a_2x_N^2 + a_3x_N^3 + \dots + a_Nx_N^N = f_N \end{cases}$$

Данная система является линейной относительно искомых коэффициентов $a_0, a_1, a_2, \dots, a_N$. Известно, что СЛАУ имеет решение, если ее определитель отличен от нуля. Определитель данной системы

$$\Delta = \begin{vmatrix} 1 & x_1 & \dots & x_1^N \\ 1 & x_2 & \dots & x_2^N \\ \dots & & & \dots \\ 1 & x_N & & x_N^N \end{vmatrix} = \prod_{0 \leq k < m \leq N} (x_k - x_m)$$

носит имя Вандермонда. Из курса математического анализа известно, что он отличен от нуля, если $x_k \neq x_m$ (т.е. все узлы интерполяции различные). Таким образом, доказано, что система имеет решение.

Мы показали, что для нахождения коэффициентов $a_0, a_1, a_2, \dots, a_N$ надо решить СЛАУ, что является сложной задачей. Но есть другие способы построения полинома N -й степени, который не требует решения такой системы.

3.4. Полином Лагранжа

Решение ищем в виде $L_n(z) = \sum_{i=0}^N f_i l_i(z)$, где $l_i(z)$ – базисные полиномы N -й степени, для которых выполняется условие:

$$l_i(x_k) = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases}. \text{ Убедимся в том, что если такие полиномы}$$

построены, то $L_N(x)$ будет удовлетворять условиям интерполяции:

$$L_n(x_i) = \sum_{k=0}^N f_k l_k(x_i) = f_0 l_0(x_i) + f_1 l_1(x_i) + \dots + f_i l_i(x_i) + \dots + f_N l_N(x_i) = f_i$$

Каким образом построить базисные полиномы? Определим

$$l_i(z) = \frac{(z - x_0)(z - x_1) \dots (z - x_{i-1})(z - x_{i+1}) \dots (z - x_N)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_N)}, \quad i=0, 1, \dots, N.$$

Легко понять, что

$$l_0(z) = \frac{(z - x_1)(z - x_2) \dots (z - x_N)}{(x_0 - x_1)(x_0 - x_2) \dots (x_0 - x_N)}, \quad l_1(z) = \frac{(z - x_0)(z - x_2) \dots (z - x_N)}{(x_1 - x_0)(x_1 - x_2) \dots (x_1 - x_N)}$$

и т.д.

Функция $l_i(z)$ является полиномом N -й степени от z и для нее выполняются условия «базисности»:

$$l_i(x_k) = \frac{(x_k - x_0)(x_k - x_1) \dots (x_k - x_{i-1})(x_k - x_{i+1}) \dots (x_k - x_N)}{(x_k - x_0)(x_k - x_1) \dots (x_k - x_{i-1})(x_k - x_{i+1}) \dots (x_k - x_N)} = 0, i \neq k;$$

$$l_i(x_i) = \frac{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_N)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_N)} = 1.$$

Таким образом, нам удалось решить задачу о построении интерполирующего полинома N -й степени, и для этого не нужно решать СЛАУ. Полином Лагранжа можно записать в виде компактной формулы:

$$L_N(z) = \sum_{i=0}^N f_i l_i(z) = \sum_{i=0}^N f_i \prod_{i \neq k} \frac{(z - x_k)}{(x_i - x_k)}.$$

Погрешность этой формулы можно оценить, если исходная функция $g(x)$ имеет производные до $N+1$ порядка:

$$r(z) = \frac{g^{(N+1)}(\xi)}{(N+1)!} \prod_{i=0}^{N+1} (z - x_i), \quad \xi \in [a, b].$$

Из этой формулы следует, что погрешность метода зависит от свойств функции $g(x)$, а также от расположения узлов интерполяции и точки z . Как показывают расчетные эксперименты, полином Лагранжа имеет малую погрешность при небольших значениях $N < 20$. При больших N погрешность начинает расти, что свидетельствует о том, что метод Лагранжа не сходится (т.е. его погрешность не убывает с ростом N).

Для вычисления значений полинома нужно выполнить количество действий, пропорциональное N^2 . В случае небольшого числа N эти действия можно выполнить на калькуляторе. В случае если N велико, для вычислений будем использовать ЭВМ.

Полином Лагранжа удобно использовать, если в одних и тех же узлах надо вычислить значения нескольких функций. В том случае, если функция одна, а узлы, в которых она задана, изменяются (или добавляются новые узлы), использование формулы Лагранжа нецелесообразно. В этом случае надо использовать интерполяционную формулу Ньютона.

Пример 3.3. Для исходных данных, использованных в примере 3.1, требуется найти значение полинома Лагранжа при $z=1$.

Для этого случая $N=3$, т.е. полином Лагранжа имеет третий порядок. Вычислим значения базисных полиномов при $z=1$:

$$\begin{aligned}
 l_0(1) &= \frac{(1-2)(1-3)(1-3.5)}{(0-2)(0-3)(0-3.5)} = 0.238 \\
 l_1(1) &= \frac{(1-0)(1-3)(1-3.5)}{(2-0)(2-3)(2-3.5)} = 1.667 \\
 l_2(1) &= \frac{(1-0)(1-2)(1-3.5)}{(3-0)(3-2)(3-3.5)} = -1.667 \\
 l_3(1) &= \frac{(1-0)(1-2)(1-3)}{(3.5-0)(3.5-2)(3.5-3.0)} = 0.762 \\
 L_3(1) &= \sum_{i=0}^3 f_i l_i(1) = -0.129.
 \end{aligned}$$

На рис. 3.3 показан листинг программы для нахождения значения полинома Лагранжа для конкретного значения z . А также приведен график сравнения с исходной дискретной функцией $f(x_i)$.

3.5. Полином Ньютона

Полином Ньютона есть еще одна форма записи глобального интерполирующего полинома N -го порядка. Она имеет вид:

$$\begin{aligned}
 P_N(z) = & f_0 + F(x_0, x_1)(z-x_0) + F(x_0, x_1, x_2)(z-x_0)(z-x_1) + \dots \\
 & + F(x_0, x_1, x_2, \dots, x_N)(z-x_0)(z-x_1) \dots (z-x_{N-1}).
 \end{aligned}$$

Здесь $F(x_0, x_1)$, $F(x_0, x_1, x_2)$, ..., $F(x_0, x_1, x_2, \dots, x_N)$ – разделенные разности первого, второго, ..., N -ого порядков, которые вычисляются по формулам:

$$\begin{aligned}
 F(x_i, x_j) &= \frac{f_i - f_j}{x_i - x_j}, \quad F(x_i, x_j, x_k) = \frac{F(x_i, x_j) - F(x_j, x_k)}{x_i - x_k}, \dots, \\
 F(x_0, x_1, \dots, x_N) &= \frac{F(x_1, x_2, \dots, x_N) - F(x_0, x_1, \dots, x_{N-1})}{x_N - x_0}.
 \end{aligned}$$

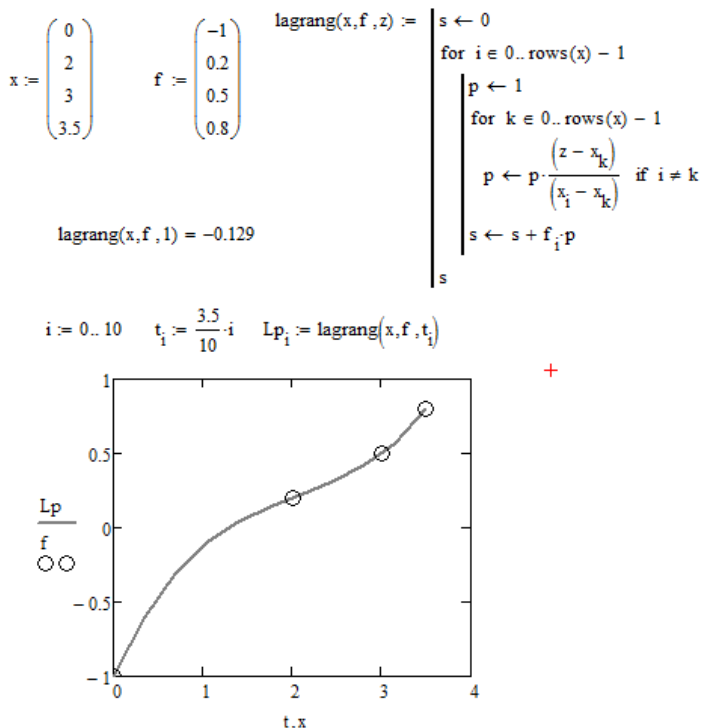


Рис. 3.3. Нахождение полинома Лагранжа в пакете MathCAD

Разделенные разности являются аналогами первой, второй, ..., N -й производной от заданной таблично функции. Их вычисление удобно проводить в таблице разделенных разностей:

x_0	f_0	—	—	—	—
x_1	f_1	$F(x_0, x_1)$	—	—	—
x_2	f_2	$F(x_1, x_2)$	$F(x_0, x_1, x_2)$	—	—
x_3	f_3	$F(x_2, x_3)$	$F(x_1, x_2, x_3)$	$F(x_1, x_2, x_3, x_4)$	—
...	...		...	...	—
x_N	f_N	$F(x_{N-1}, x_N)$	$F(x_{N-2}, x_{N-1}, x_N)$	$F(x_{N-3}, x_{N-2}, x_{N-1}, x_N)$	$F(x_N, x_{N-1}, \dots, x_1, x_0)$

Порядок узлов при вычислении разделенных разностей не важен. Если число узлов изменится, то в таблице разделенных разностей появится еще одна строчка, а в формуле Ньютона — еще один член, все остальные вычисления повторять не надо.

Поскольку результатом интерполяционных формул Ньютона и Лагранжа является один и тот же полином N -го порядка, то их погрешность ведет себя одинаково.

Пример 3.4. Для исходных данных, использованных в примере 3.1, вычислим значение полинома Ньютона. Сначала заполним таблицу разделенных разностей:

x	F	$F(x_i, x_j)$	$F(x_i, x_j, x_k)$	$F(x_0, x_1, x_2, x_3)$	$z - x_i$
0.00	-1	—	—	—	1.00
2.00	0.2	0.6	—	—	-1.00
3.00	0.5	0.3	-0.1	—	-2.00
3.50	0.8	0.6	0.2	0.085714	-2.50

Используя формулу Ньютона, получим:

$$P_3(1) = -1 + 0.6 \cdot 1 + (-0.1) \cdot 1 \cdot (-1) + 0.0857 \cdot 1 \cdot (-1) \cdot (-2) = -0.129.$$

3.6 Ряды Фурье

Ряд Фурье позволяет изучать как периодические, так и непериодические функции, разлагая их на компоненты. Переменные токи и напряжения, смещения, скорость и ускорение кривошипно-шатунных механизмов, акустические волны - это типичные практические примеры применения периодических функций в инженерных расчетах. В терминах обработки сигналов преобразование Фурье берет представление функции сигнала в виде временных рядов и отображает его в частотный спектр. То есть оно превращает функцию времени в функцию частоты; это разложение функции на гармонические составляющие на различных частотах. Преобразование Фурье может представить сигнал, изменяющийся во времени, в виде зависимости частоты и амплитуды, также оно даёт информацию о фазе (рис.3.4).

Разложение в ряд Фурье основывается на предположении, что все имеющие практическое значение функции в интервале $\pi \leq x \leq \pi$ можно выразить в виде сходящихся тригонометрических рядов (ряд считается сходящимся, если сходится последовательность частичных сумм, составленных из его членов).

Согласно гипотезе Фурье не существует функции, которую нельзя было бы разложить в тригонометрический ряд. Разложим функцию $f(t)$ в ряд на отрезке $[-\pi, \pi]$

$$f(t) = a_0/2 + a_1\cos(t) + a_2\cos(2t) + a_3\cos(3t) + \dots + b_1\sin(t) + b_2\sin(2t) + b_3\sin(3t) + \dots, \quad (3.1)$$

где n -ые элементы ряда выражаются как

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos(nt) dt, \quad (3.2)$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin(nt) dt \quad (3.3)$$

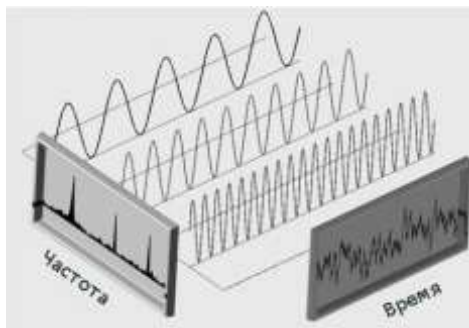


Рис. 3.4. Иллюстрация к разложению в ряд Фурье

Коэффициенты a_n и b_n называют **коэффициентами Фурье**, а представление функции $f(t)$ по формуле (3.1) – **разложением в ряд Фурье**. Иногда разложение в ряд Фурье, представленное в таком виде, называют действительным разложением в ряд Фурье, а коэффициенты – действительными коэффициентами Фурье (в отличие от комплексного разложения).

Проанализируем выражения (3.2) и (3.3). Коэффициент a_0 представляет собой среднее значение функции $f(t)$ на отрезке $[-\pi, \pi]$ или постоянную составляющую сигнала $f(t)$. Коэффициенты a_n и b_n (при $n > 0$) – это амплитуды косинусных и синусных составляющих функции (сигнала) $f(t)$ с угловой частотой равной n . Другими словами, данные коэффициенты задают величину частотных составляющих сигналов. Например, когда мы говорим о звуковом сигнале с низкими частотами (например, звуки бас-гитары), это означает, что коэффициенты a_n и b_n больше при меньших значениях n и наоборот – в высокочастотных звуковых

колебаниях (например, звук скрипки) больше при больших значениях n .

Колебание самого большого периода (или самой низкой частоты), представленное суммой $a_1 \cos(t)$ и $b_1 \sin(t)$ называют колебанием основной частоты или первой гармоникой. Колебание с периодом равным половине периода основной частоты – второй гармоникой, колебание с периодом равным $1/n$ основной частоты – n -гармоникой. Таким образом, с помощью разложения функции $f(t)$ в ряд Фурье, мы можем осуществить переход из временной области в частотную. Такой переход обычно необходим для выявления особенностей сигнала, которые «незаметны» во временной области.

Обратим внимание, что формулы (3.2) и (3.3) применимы для периодического сигнала с периодом равным 2π . В общем случае в ряд Фурье можно разложить периодический сигнал с периодом T , тогда при разложении используется отрезок $[-T/2, T/2]$. Период первой гармоники равен T и составляющие примут вид $\cos(2\pi t/T)$ и $\sin(2\pi t/T)$, составляющие n -гармоники – $\cos(2\pi n t/T)$ и $\sin(2\pi n t/T)$. Если обозначить угловую частоту первой гармоники $\omega_0 = 2\pi/T$, тогда составляющие n -гармоники принимают вид $\cos(\omega_0 n t)$, $\sin(\omega_0 n t)$ и

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(\omega_0 n t) + b_n \sin(\omega_0 n t)], \quad (3.4)$$

где коэффициенты Фурье вычисляются по формулам

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos(\omega_0 n t) dt, \quad b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin(\omega_0 n t) dt.$$

Разложение в **ряд Фурье** используется для гармонического или спектрального анализа периодических сигналов. Для спектрального анализа непериодических сигналов используется **преобразование Фурье**. Для этого ряд (3.4) представим, используя систему базисных функций в виде экспонент с мнимыми показателями:

$$f(t) = \sum_{n=-\infty}^{\infty} C_n \exp\left(\frac{2j\pi n t}{T}\right) \quad (3.5)$$

$$C_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) \exp\left(-\frac{2j\pi nt}{T}\right) dt \quad (3.6)$$

Опустив ряд выкладок, выражение (3.6) запишем в виде

$$C(\omega) = \int_{-\infty}^{+\infty} f(t) \exp(-j\omega t) dt.$$

Данная формула называется **прямым преобразованием Фурье** или преобразованием Фурье. Обычно преобразование Фурье обозначают той же (только прописной) буквой, что и аппроксимируемая функция (которая обычно обозначается строчной буквой)

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) \exp(-j\omega t) dt.$$

Функция $F(\omega)$ называется функцией **спектральной плотности** (или просто спектральной плотностью, преобразованием Фурье, Фурье-образом). Область значений функции $F(\omega)$ в общем случае является множество комплексных чисел.

Обратное преобразование Фурье, обеспечивающее восстановление исходной функции $f(t)$ по функции спектральной плотности вычисляется следующим образом

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) \exp(j\omega t) dt.$$

Дискретное преобразование Фурье (ДПФ, DFT — Discrete Fourier Transform) — это одно из преобразований Фурье, широко применяемых в алгоритмах цифровой обработки сигналов (его модификации применяются в сжатии звука в MP3, сжатии изображений в JPEG и др.), а также в других областях, связанных с анализом частот в дискретном (к примеру, оцифрованном аналоговом) сигнале. Дискретное преобразование Фурье требует в качестве входа дискретную функцию. Такие функции часто создаются путём дискретизации (выборки значений из непрерывных функций). Недостатком данного алгоритма является большой объем повторяющихся вычислений. Устранение этих избыточных операций приводит к так называемому алгоритму

быстрого преобразования Фурье, который обычно и используется.

Быстрое преобразование Фурье (БПФ, FFT) — алгоритм быстрого вычисления дискретного преобразования Фурье (ДПФ). То есть алгоритм вычисления за число действий, меньшее чем $O(N^2)$, требуемых для прямого (по формуле) вычисления ДПФ (N — количество значений сигнала, измеренных за период, а также количество компонент разложения). Иногда под БПФ понимается один из быстрых алгоритмов, называемый алгоритмом прореживания по частоте/времени или алгоритмом по основанию 2.

Для того чтобы реализовать преобразование Фурье в пакете **MathCAD**, необходимо на панели Symbolic выбрать оператор `fourier` для прямого преобразования и `invfourier` - для обратного. Этот оператор нужно поместить следом за функцией, которую нужно преобразовать, а в качестве единственного параметра нужно указать переменную, относительно которой эта функция будет преобразована. Примеры использования показаны рис. 3.5 для функции $f(t) = e^{-2 \cdot |t|}$ и на рис. 3.6, где к функции $f(t)$ применяется амплитудно-частотная модуляция, а далее результат раскладывается в ряд.

$$f(t) := e^{-2 \cdot |t|} \quad F(\omega) := f(t) \text{ fourier, } t \rightarrow \frac{4}{\omega^2 + 4}$$

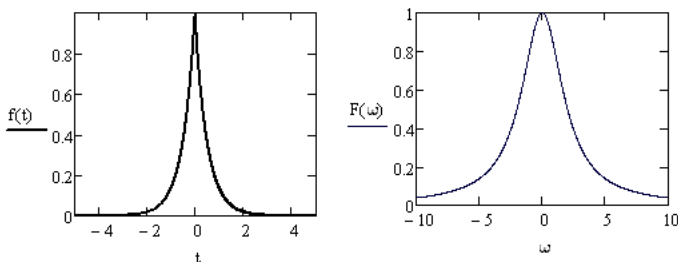


Рис. 3.5. Пример разложения в ряд Фурье с помощью символьной функции `fourier`

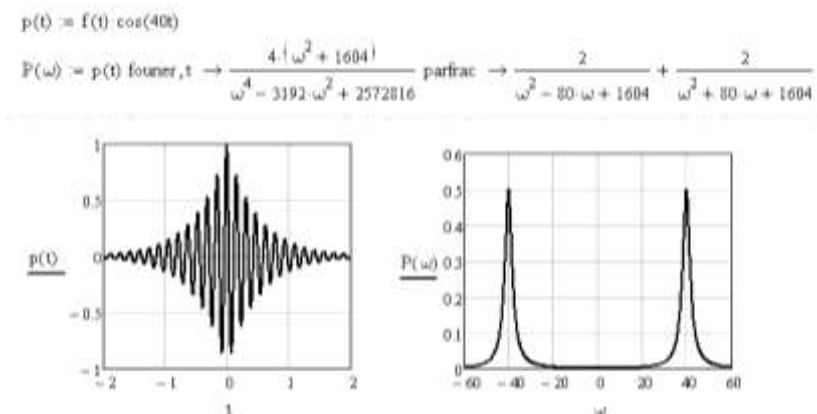


Рис. 3.6. Пример разложения в ряд Фурье с помощью символьной функции `fourier`

MathCAD содержит функции для быстрого дискретного преобразования Фурье (БПФ) и его обращения. Существует два типа функций для дискретного преобразования Фурье: `fft` и `ifft`, `cfft` и `icfft`. Эти функции дискретны: они берут в качестве аргументов и возвращают векторы и матрицы.

Функции `fft` и `ifft` используются, если выполнены следующие условия: (1) аргументы вещественные; (2) – вектор данных имеет 2^m элементов.

Во всех прочих случаях используются функции `cfft` и `icfft`.

Соблюдать первое условие необходимо, потому что функции `fft` и `ifft` используют тот факт, что для вещественных данных вторая половина преобразования Фурье является комплексно – сопряженной с первой. MathCAD отбрасывает вторую половину вектора результата, что сохраняет время и память при вычислениях. Пара функций `cfft` и `icfft` не используют симметрию в преобразовании и могут использоваться для вещественных и комплексных чисел.

Второе условие требуется, потому что пара функций `fft` и `ifft` используют высокоэффективный алгоритм быстрого преобразования Фурье. Для этого вектора аргумента, используемо-

го функцией `fft`, должен состоять из 2^m элементов. Алгоритм функций `cfft` и `icfft` допускает в качестве аргументов векторы и матрицы произвольного размера. Для двухмерного преобразования Фурье используются только эти функции. Функции `fft` и `ifft`, `cfft` и `icfft` взаимно обратные друг другу, то есть справедливо:

$$\text{ifft}(\text{fft}(y)) = y \quad \text{и} \quad \text{icfft}(\text{cfft}(v)) = v.$$

На рис. 3.7 проиллюстрировано использование функций `fft(v)` и `ifft(v)` для сигнала синусоидальной формы, на который наложены помехи с помощью функции `rnd(x)`, генерирующей случайные числа в диапазоне от 0 до x .

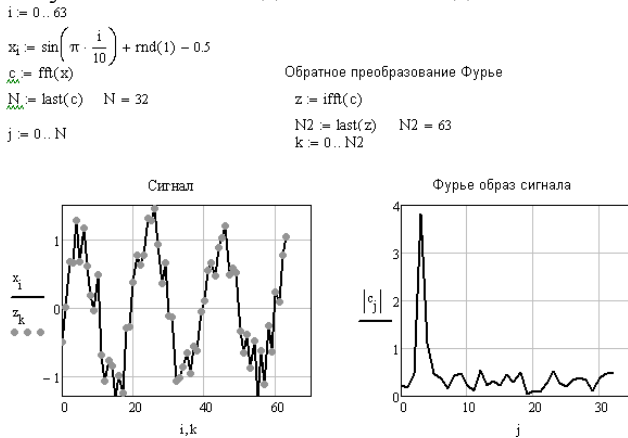


Рис. 3.7. Прямое и обратное преобразование Фурье с помощью функций `fft` и `ifft`

На данных графиках приведен Фурье образ сигнала `c` и сравнение исходного сигнала `x` с восстановленным из Фурье-образа. Более подробно о Фурье-анализе можно прочесть в [2] и [3].

3.7 Метод наименьших квадратов

Во всех вышеизложенных методах приближения функции условия интерполяции выполнялось точно. Однако в тех случаях, когда исходные данные $x_i, f_i, i=1, \dots, N$, заданы с некоторой погрешностью ε , можно требовать лишь приближенное выпол-

нение условий интерполяции: $|F(x_i) - f_i| < \varepsilon$. Это условие означает, что интерполирующая функция $F(x)$ проходит не точно через заданные точки, а в некоторой их окрестности, так, например, как это показано на рис. 3.8. Приближим исходные данные глобальным полиномом. Если решать задачу интерполяции точно, то полином должен иметь степень N . При рассмотрении полинома Лагранжа мы выяснили, что полином N -й степени хорошо приближает исходную функцию только при небольших значениях N .

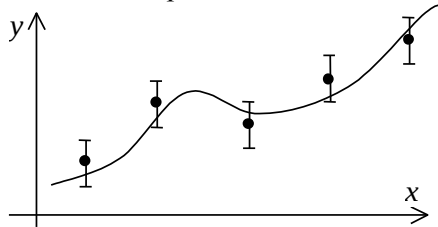


Рис. 3.8. Приближенное выполнение условий интерполяции

Будем искать полином низкой степени, например, $P_3(x) = a_1 + a_2x + a_3x^2 + a_4x^3$. Если $N > 4$, то точная задача решений не имеет: для четырех неизвестных коэффициентов (a_1, a_2, a_3, a_4) условия интерполяции дают $N > 4$ уравнений. Но теперь точного выполнения условий интерполяции не требуется, мы хотим, чтобы полином проходил рядом с заданными точками. Существует много таких полиномов, каждый из которых определяется своим набором коэффициентов. Среди всех возможных полиномов этого вида выберем тот, что имеет наименьшее среднеквадратичное отклонение в узлах интерполяции от заданных значений, т.е. многочлен должен быть самым близким к заданным точкам из всех возможных многочленов третьей степени в смысле *метода наименьших квадратов* (МНК). В i -й точке полином $P_3(x)$ отклоняется от значения f_i на величину $(P_3(x_i) - f_i)$. Суммируем квадраты отклонений полинома по всем точкам $i = 1, 2, \dots, N$, получим функционал квадратов отклонений:

$$G(a_1, a_2, a_3, a_4) = \sum_{i=1}^N (P_3(x_i) - f_i)^2 = \sum_{i=1}^N (a_1 + a_2x_i + a_3x_i^2 + a_4x_i^3 - f_i)^2$$

Найдем минимум этого функционала. Для этого приравняем нулю его частные производные по переменным a_1, a_2, a_3, a_4 . Используя стандартные правила дифференцирования, получим:

$$\begin{aligned}\frac{\partial \mathcal{G}}{\partial a_1} &= 2 \sum_{i=1}^N (a_1 + a_2 x_i + a_3 x_i^2 + a_4 x_i^3 - f_i) = 0 \\ \frac{\partial \mathcal{G}}{\partial a_2} &= 2 \sum_{i=1}^N x_i (a_1 + a_2 x_i + a_3 x_i^2 + a_4 x_i^3 - f_i) = 0 \\ \frac{\partial \mathcal{G}}{\partial a_3} &= 2 \sum_{i=1}^N x_i^2 (a_1 + a_2 x_i + a_3 x_i^2 + a_4 x_i^3 - f_i) = 0 \\ \frac{\partial \mathcal{G}}{\partial a_4} &= 2 \sum_{i=1}^N x_i^3 (a_1 + a_2 x_i + a_3 x_i^2 + a_4 x_i^3 - f_i) = 0\end{aligned}$$

Собирая коэффициенты при неизвестных a_i , получим СЛАУ относительно вектора неизвестных (a_1, a_2, a_3, a_4) :

$$\begin{aligned}N \cdot a_1 + \sum_{i=1}^N x_i \cdot a_2 + \sum_{i=1}^N x_i^2 \cdot a_3 + \sum_{i=1}^N x_i^3 \cdot a_4 &= \sum_{i=1}^N f_i \\ \sum_{i=1}^N x_i \cdot a_1 + \sum_{i=1}^N x_i^2 \cdot a_2 + \sum_{i=1}^N x_i^3 \cdot a_3 + \sum_{i=1}^N x_i^4 \cdot a_4 &= \sum_{i=1}^N f_i \cdot x_i \\ \sum_{i=1}^N x_i^2 \cdot a_1 + \sum_{i=1}^N x_i^3 \cdot a_2 + \sum_{i=1}^N x_i^4 \cdot a_3 + \sum_{i=1}^N x_i^5 \cdot a_4 &= \sum_{i=1}^N f_i \cdot x_i^2 \\ \sum_{i=1}^N x_i^3 \cdot a_1 + \sum_{i=1}^N x_i^4 \cdot a_2 + \sum_{i=1}^N x_i^5 \cdot a_3 + \sum_{i=1}^N x_i^6 \cdot a_4 &= \sum_{i=1}^N f_i \cdot x_i^3\end{aligned}$$

Полученная система называется *нормальной*. Для ее решения используют стандартные методы решения СЛАУ. Как правило, число неизвестных системы (т.е. число коэффициентов интерполирующей функции) невелико, поэтому можно использовать точные методы решения СЛАУ, например, метод Крамера или метод Гаусса. Метод наименьших квадратов позволяет «приблизить» исходные данные с помощью линейной комбинации любых элементарных функций. Часто используются приближения линейной $F(x)=a_1+a_2x$, тригонометрической $F(x)=a_1\sin(x)+a_2\cos(x)$, экспоненциальной $F(x)=a_1e^x+a_2e^{-x}$ и т.д. функциями.

Пример 3.5. Найдем линейную зависимость $F(x)=a_1+a_2x$, приближающую заданную таблично функцию:

x	-1	-0.5	0.7	1	1.2
f	-0.5	-0.2	0.2	0.5	1

В этом случае $N=5$, нормальная система имеет вид:

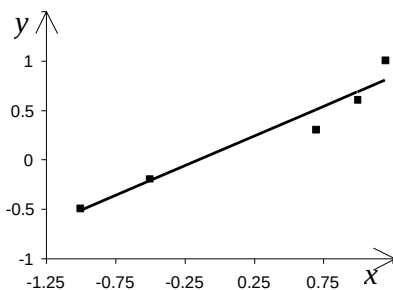


Рис. 3.9. Подбор линейной зависимости МНК

$$N \cdot a_1 + \sum_{i=1}^N x_i \cdot a_2 = \sum_{i=1}^N f_i$$

$$\sum_{i=1}^N x_i \cdot a_1 + \sum_{i=1}^N x_i^2 \cdot a_2 = \sum_{i=1}^N f_i \cdot x_i$$

Вычисляем

$$\sum_{i=1}^N x_i, \quad \sum_{i=1}^N x_i^2, \quad \sum_{i=1}^N f_i, \quad \sum_{i=1}^N f_i x_i,$$

подставляем в нормальную систему:

$$\begin{cases} 5 \cdot a_1 + 1.4 a_2 = 1 \\ 1.4 \cdot a_1 + 4.18 a_2 = 2.44 \end{cases}$$

Решаем методом Крамера:

$$\Delta = \begin{vmatrix} 5 & 1.4 \\ 1.4 & 4.18 \end{vmatrix} = 18.94,$$

$$\Delta_1 = \begin{vmatrix} 1 & 1.4 \\ 2.44 & 4.18 \end{vmatrix} = 0.764, \quad \Delta_2 = \begin{vmatrix} 5 & 1 \\ 1.4 & 2.44 \end{vmatrix} = 10.8,$$

$$a_1 = \frac{\Delta_1}{\Delta} = -0.04, \quad a_2 = \frac{\Delta_2}{\Delta} = 0.57.$$

Проверка. Вычисляем значение $G = \sum_{i=1}^5 (a_1 + a_2 x_i - f_i)^2 = 0.148$.

График функции $F(x) = -0.04 + 0.57x$ показан на рис. 3.9 сплошной линией. Точками показаны исходные данные. Можно видеть, что найденная линейная функция действительно приближает заданные точки.

В MathCAD метод наименьших квадратов тесно связан с линейной регрессией ($y(x) = b + ax$), поскольку коэффициенты a и b вычисляются из условия минимизации суммы квадратов ошибок $|b + ax_i - y_i|$. Для расчета в MathCAD имеются два дублирующих друг друга способа:

`line (x, y)` возвращает вектор из двух элементов коэффициентов линейной регрессии $b + ax$;

`intercept (x, y)` возвращает коэффициент b линейной регрессии;

`slope (x, y)` возвращает коэффициент a линейной регрессии;

x - вектор действительных данных аргумента;

y - вектор действительных данных значений того же размера.

Примеры 3.6 и 3.7 иллюстрируют применение этих функций: здесь x и y – это исходные данные, которые мы аппроксимируем линейной функцией $f(t)$.

Пример 3.6

$$x := (0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6)^T \quad y := (4.1 \ 2.4 \ 3 \ 4.3 \ 3.6 \ 5.2 \ 5.9)^T$$

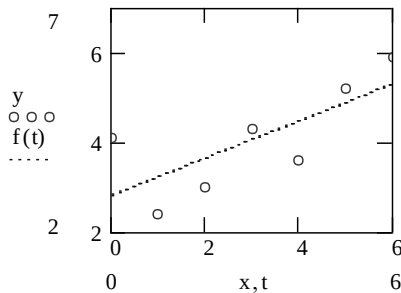
$$\text{line}(x, y) = \begin{pmatrix} 2.829 \\ 0.414 \end{pmatrix} \quad f(t) := \text{line}(x, y)_0 + \text{line}(x, y)_1 \cdot t$$

Пример 3.7 (другая форма записи).

$$x := (0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6)^T \quad y := (4.1 \ 2.4 \ 3 \ 4.3 \ 3.6 \ 5.2 \ 5.9)^T$$

$$\text{intercept}(x, y) = 2.829 \quad \text{slope}(x, y) = 0.414$$

$$f(t) := \text{intercept}(x, y) + \text{slope}(x, y) \cdot t$$



3.9. Заключение

Таким образом, мы познакомились с различными способами построения интерполирующей функции. Каждый из методов имеет свою точность и особенности использования. Можно сделать некоторые общие рекомендации по выбору метода интерполяции. Локальные методы, особенно кусочно-постоянный и

кусочно–линейный, имеют достаточно простую реализацию. Достоинством этих методов является то, что эти методы сходятся, т.е. с ростом числа узлов погрешность методов убывает. Наиболее точным из рассмотренных локальных методов является кубический интерполяционный сплайн. Кусочно–линейный, кусочно–параболический и кусочно–кубический интерполяционные методы можно использовать также и для вычисления производных.

Глобальный интерполяционный полином, записанный в форме Лагранжа или Ньютона, дает маленькую погрешность при небольшом числе узлов ($N < 20$). В этом случае погрешность гораздо меньше, чем погрешность локальных методов. Однако, начиная с некоторого N погрешность глобального полинома начинает расти, что может даже привести к аварийному останову выполнения программы ЭВМ. В случае большого числа узлов использование глобальных способов нецелесообразно.

С помощью разложения функции $f(t)$ в ряд Фурье, мы можем осуществить переход из временной области в частотную. Такой переход обычно необходим для выявления особенностей сигнала, которые «незаметны» во временной области.

С помощью МНК можно решить задачу интерполяции приближенно. Метод позволяет приблизить сложные или таблично заданные функции с помощью линейной комбинации некоторых элементарных функций.

Глава 4. Решение задач оптимизации

В этом разделе рассматриваются инструменты MathCAD, позволяющие решать задачи на поиск экстремума функции, т.е. максимального или минимального значения функции $F(x)$ в бесконечной или конечной области определения аргументов. Поскольку задача о нахождении максимального значения функции $F(x)$ равнозначна задаче поиска минимального значения функции $-F(x)$, будем называть задачу поиска экстремума задачей минимизации.

Если аргумент функции x является вещественным скаляром, то речь идет о поиске минимума функции одной переменной. Если аргументом функции является вектор с n компонентами, то мы имеем дело с поиском минимума функции n переменных. Если аргумент сам является функцией $x=x(t)$, то необходимо найти минимум **функционала**. Если множество значений аргумента ограничено системой дополнительных уравнений или неравенств, то говорят о поиске **условного** экстремума. Если минимизируемая функция и система ограничений линейны, то возникает задача **линейного программирования**. Такие задачи типичны для экономических расчетов, направленных на оптимизацию какого-либо экономического показателя (издержек производства, прибыли, времени и т.д.).

4.1. Поиск экстремума функции одной переменной аналитическим методом

Проблема поиска минимума включает в себя задачи нахождения глобального и локального минимума. На рис. 4.1. приведен график функции полинома четвертого порядка, имеющего два локальных минимума, один из них является глобальным, а второй – локальным.

Локальные минимумы функции одной переменной можно найти, решив нелинейное в общем случае алгебраическое уравнение

$$\frac{dF(x)}{dx} = 0 \quad (4.1)$$

Если функция $F(x)$ задана аналитически, то можно получить уравнение (4.1) с помощью символьного дифференцирования, а затем решить его с помощью методов, описанных в главе 1. Соответствующий пример такого решения задачи оптимизации в MathCAD приведен на рис. 4.2.

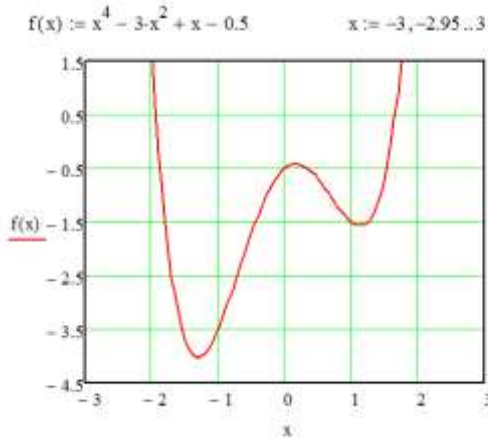


Рис. 4.1. График функции с двумя локальными минимумами

$$\begin{aligned}
 f(x) &:= x^4 - 3x^2 + x - 0.5 \\
 f'(x) &:= \frac{d}{dx}f(x) \rightarrow 4x^3 - 6x + 1 & y &:= \begin{pmatrix} 1 \\ -6 \\ 0 \\ 4 \end{pmatrix} & z &:= \text{polyroots}(y) & z &= \begin{pmatrix} -1.301 \\ 0.17 \\ 1.131 \end{pmatrix} \\
 f''(x) &:= \frac{d}{dx}f'(x) \rightarrow 12x^2 - 6 \\
 f(z_0) &= -4.014 & f(z_1) &= -0.416 & f(z_2) &= -1.57 \\
 f''(z_0) &= 14.306 & f''(z_1) &= -5.653 & f''(z_2) &= 9.347
 \end{aligned}$$

Рис. 4.2. Нахождение локальных экстремумов функции одной переменной с помощью символьного дифференцирования

Здесь $f_1(x)$, $f_2(x)$ – функции первой и второй производной, вектор y содержит коэффициенты полинома третьего порядка $f_1(x)$, компонентами вектора z являются корни полинома $f_1(x)$.

Для того чтобы определить, при каких значениях x достигается максимум, а при каких – минимум, в последних строках

листинга на рис. 4.2. проводится подстановка найденных корней в исходное уравнение. Кроме того, выполнена проверка значений второй производной в точках локальных экстремумов, которая показала, что в двух точках вторая производная положительна (локальные минимумы), а в одной – отрицательна (локальный максимум). Для нахождения глобального минимума необходимо найти все локальные минимумы, а затем выбрать из них наименьший. В приведенном примере глобальный минимум $f = -4.014$ достигается в точке $x = -1.301$.

Однако зачастую уравнение (4.1) имеют весьма сложный вид и для его решения необходимо применять итерационные методы (см. главу 1), которые к тому же плохо сходятся. Если ищется минимум функции n переменных, то уравнение (4.1) перейдет в систему нелинейных уравнений

$$\frac{\partial F(x)}{\partial x_i} = 0, i = 1, \dots, n. \quad (4.1')$$

При отыскании минимума функционала необходимо решать дифференциальное или интегро-дифференциальное уравнение. Поэтому в общем случае решение задач оптимизации проводится приближенными методами, не связанными с приведением задачи к уравнению вида (4.1). В MathCAD реализованы несколько так называемых **градиентных** методов, к которым принадлежит, например, описанный в главе 1 метод Ньютона. Суть методов состоит в последовательном продвижении по множеству значений аргумента x с учетом значений градиента минимизируемой функции $F(x)$. Эти же методы положены в основу встроенной функции Find, которая использовалась при решении СЛАУ (см. главу 2). MathCAD предлагает три различных варианта градиентных методов. Более подробно о принципах действия, выборе конкретного метода и его параметров см. [4].

4.2. Поиск экстремума функции с помощью встроенных функций MathCAD

Поиск экстремумов реализован в MathCAD в виде встроенных функций Minimize и Maximize. В качестве аргументов функций Minimize и Maximize задаются имя функции, ко-

торую необходимо оптимизировать, и список значений аргументов, по которым производится оптимизация.

Поскольку в основу работы этих функций положены приближенные алгоритмы, перед вызовом функций необходимо задать начальное значение аргумента, от выбора которого будет зависеть найденное значение минимума или максимума. Таким образом, чтобы найти глобальный экстремум, необходимо несколько раз решить задачу поиска локальных экстремумов с различными начальными приближениями, а затем «вручную» выбрать из них максимальное или минимальное значение.

На рис. 4.3. приведено решение задачи поиска локальных экстремумов для функции, исследованной в предыдущем параграфе с помощью аналитического метода. Из примера можно видеть, что в случае «неудачного» задания начального приближения для функции `Maximize` в последнем случае решение не найдено. Заданное значение $x=20$ лежит на бесконечно возрастающей ветке решения, и, несмотря на то, что в другой стороне находится локальный максимум, численный метод не смог его найти. Имя функции подсвечивается красным, что свидетельствует о невозможности вывести результат вычислений.

```
f(x) := x4 - 3·x2 + x - 0.5
x := 0      Minimize(f,x) = -1.301      x := 1      Minimize(f,x) = 1.131
x := 0      Maximize(f,x) = 0.17        x := 20      Maximize(f,x) = ■
```

Рис. 4.3. Листинг MathCAD с примерами нахождения экстремумом функции одной переменной

Нахождение экстремума функции нескольких переменных принципиально не отличается от случая функции одной переменной. На рис. 4.4 приведен фрагмент рабочего листа MathCAD, на котором построен трехмерный график исследуемой функции $Z(x, y) = x^3 + 2x \cdot y + y^2$, найдены первые и вторые частные производные. На рис. 4.5 аналитически найдены стационарные точки, в которых обращаются в ноль первые частные

производные функции $Z(x,y)$, и выполнена проверка достаточных условий локальных экстремумов:

$D = Z_{xx} \cdot Z_{yy} - Z_{xy}^2 > 0, Z_{xx} > 0$ - точка локального минимума,

$D > 0, Z_{xx} < 0$ - точка локального максимума.

Затем (рис. 4.6) выполнен поиск локальных экстремумов с помощью стандартных функций MathCAD.

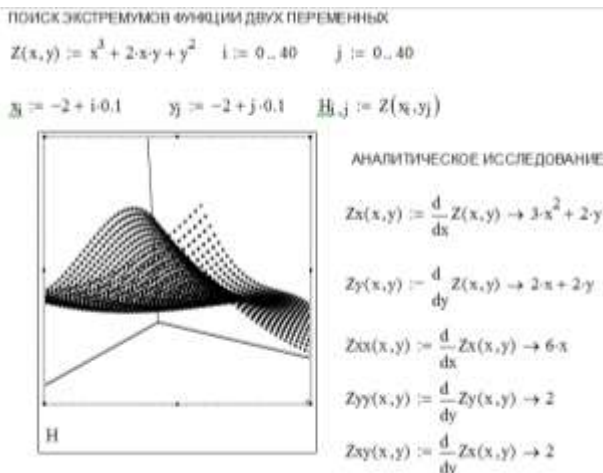


Рис. 4.4. Листинг MathCAD с примером графика функции двух переменных



Рис. 4.5. Листинг MathCAD с примером аналитического исследования функции двух переменных

```

x := 1      y := 1      Minimize(Z, x, y) =  $\begin{pmatrix} 0.667 \\ -0.667 \end{pmatrix}$ 

x := 0      y := 0      Maximize(Z, x, y) =  $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ 

x := 1      y := 1      Maximize(Z, x, y) =

```

Рис. 4.6. Листинг MathCAD с примером нахождения локального минимума функции двух переменных

Необходимо обратить внимание на то, что при задании в качестве начального приближения значений $x=0$, $y=0$ (стационарная точка, не являющаяся локальным экстремумом) обе стандартные функции дают «ложное» решение. При задании в качестве начального приближения значений x , y , не являющихся координатами стационарной точки, стандартные функции работают «правильно». Найденный локальный минимум совпадает с тем, что найден аналитическим способом, а функция `Maximize` не дает результата, что свидетельствует о неограниченности функции Z сверху.

Поставим теперь задачу нахождения условного экстремума.

```

x := 1      y := 1
Given      1 < x < 2      0 < y < 1

Minimize(Z, x, y) =  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 

x := 1      y := 1
Given      1 < x < 2      0 < y < 1

Maximize(Z, x, y) =  $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ 

```

Рис. 5.7. Пример решения условной задачи оптимизации функции двух переменных

Дополнительные условия, записанные в виде логических выражений, должны быть добавлены на рабочий лист после служебного слова `Given` (Дано). После этого стандартным образом вызывается функция `Maximize` или `Minimize`. Листинг, приведенный на рис. 4.7 показывает решение задачи о функции двух переменных, для которой выше решена задача безусловной оптимизации.

В заключение данного раздела заметим, что мы рассмотрели самые простые задачи оптимизации, связанные с нахождением экстремумов функций. За пределами нашего внимания остались многочисленные оптимизационные задачи, которые могут быть успешно решены с использованием MathCAD. Некоторые дополнительные сведения можно найти в [3, 4, 10].

Глава 5. Численное дифференцирование и интегрирование

5.1. Численное дифференцирование

Численное дифференцирование, т.е. нахождение значений производных заданной функции $y = f(x)$ в заданных точках x , в отличие от рассматриваемого в следующем параграфе численного интегрирования, можно считать не столь актуальной проблемой в связи с отсутствием принципиальных трудностей с аналитическим нахождением производных. Однако имеется ряд важных задач, для которых численное дифференцирование является единственным способом нахождения производной. Это, например, поиск производной таблично заданной функции или дифференцирование функции в процессе численного решения, когда значения этой функции известны только в узлах сетки. Кроме того, возможно сильное усложнение задачи при аналитическом дифференцировании функции, и использование численного подхода упрощает задачу.

Существует несколько способов получения формул численного дифференцирования, которые в конечном итоге могут привести к одним и тем же формулам. Во-первых, можно аппроксимировать таблично заданную функцию каким-либо способом (линейная интерполяция, многочлен Лагранжа, сплайн-функции и т.д.) и дифференцировать полученную непрерывную функцию, приближающую исходную. Эта задача рассматривалась выше. Во-вторых, для вывода формул численного дифференцирования можно воспользоваться понятием *конечных разностей*. Пусть узлы таблицы x_i расположены на равных расстояниях: $x_i = x_0 + ih$, f_i – соответствующие значения функции; величину h называют шагом таблицы. Разности $f_{i+1} - f_i$ называют разностями первого порядка. Эти величины обозначают как разность вперед ($\Delta_+ f_i = f_{i+1} - f_i$), разность назад ($\Delta_- f_{i+1} = f_{i+1} - f_i = \Delta_+ f_i$) и центральную разность $\Delta_{\pm} f_i = f_{i+1} - f_{i-1}$. Разности высших порядков образуют при помощи рекуррентных соотношений $\Delta^m f_i = \Delta(\Delta^{m-1} f_i) = \Delta^{m-1} f_{i+1} - \Delta^{m-1} f_i$. Используя эти формулы, первую производную можно определить разными способами:

$$f'_+(x_i) = \frac{f_{i+1} - f_i}{h}, \quad (5.1)$$

$$f'_-(x_i) = \frac{f_i - f_{i-1}}{h}, \quad (5.2)$$

$$f'_\pm(x_i) = \frac{f_{i+1} - f_{i-1}}{2h}. \quad (5.3)$$

Геометрически вычисление производной по трем этим формулам эквивалентно замене касательной в точке B прямыми BC , AB и AC соответственно и поиску тангенса угла наклона этих прямых вместо тангенса угла наклона касательной (рис. 5.1).

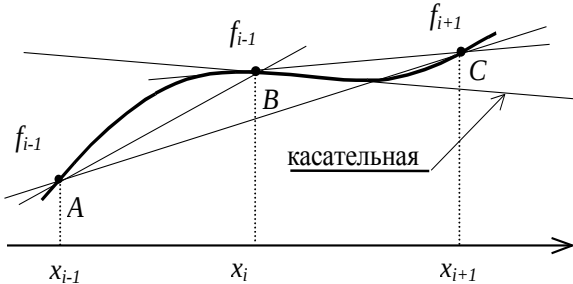


Рис. 5.1. Геометрическая интерпретация разностного дифференцирования

Изучим вопрос о порядке точности (аппроксимации) этих формул. Разложим $f(x)$ в ряд Тейлора в окрестности точки x_i .

$$f_{i+1} = f_i + hf'_i + \frac{h^2}{2} f''_i + \frac{h^3}{6} f'''_i + \dots, \quad f_{i-1} = f_i - hf'_i + \frac{h^2}{2} f''_i - \frac{h^3}{6} f'''_i + \dots$$

Подставив эти разложения в (5.1), получаем

$$f'_+(x_i) = \frac{f_i + hf'_i + \frac{h^2}{2} f''_i + \frac{h^3}{6} f'''_i + \dots - f_i}{h} = f'_i + \frac{h}{2} f''_i + \frac{h^2}{6} f'''_i + \dots$$

Здесь f'_i – первая производная, которую необходимо найти, а $\frac{h}{2} f''_i + \frac{h^2}{6} f'''_i + \dots$ – погрешность, с которой вычисляется производная. Видим, что первый член погрешности имеет порядок h ,

значит, при измельчении шага сетки погрешность будет уменьшаться пропорционально h^1 . Поэтому говорят, что формула имеет первый порядок точности. Нетрудно показать, что формула (5.2) также имеет первый порядок аппроксимации.

Покажем, что формула центральной разности имеет второй порядок точности. Подставим в (5.3) разложения для f_{i+1} и f_{i-1}

$$\begin{aligned} f'_{\pm}(x_i) &= \frac{f_i + hf'_i + \frac{h^2}{2} f''_i + \frac{h^3}{6} f'''_i - f_i + hf'_i - \frac{h^2}{2} f''_i + \frac{h^3}{6} f'''_i + \dots}{2h} = \\ &= f'_i + \frac{h^2}{6} f'''_i + \dots \end{aligned}$$

Погрешность вычисления производной пропорциональна h^2 , значит формула (5.3) имеет второй порядок аппроксимации.

Основываясь на понятии конечных разностей, можно получить формулы для аппроксимации производных высших порядков. Покажем это на примере формулы центральной разности для аппроксимации второй производной.

$$f''_{\pm}(x_i) = \frac{\Delta_+ f_i - \Delta_- f_i}{h^2} = \frac{f_{i+1} - f_i - f_i + f_{i-1}}{h^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}.$$

Можно доказать, что эта формула имеет второй порядок точности (доказать самостоятельно).

Наконец, третьим способом получения формул численного дифференцирования является *метод неопределенных коэффициентов*. Представим приближенно в точке $x = x_0$ k -ю производную таблично заданной функции в виде линейной комбинации ее значений в узлах

$$f^{(k)}(x_0) \approx \sum_{i=1}^N C_i f(x_i). \quad (5.4)$$

где C_i – числовые коэффициенты, которые выберем из условия, чтобы эта формула была точна для многочлена максимально высокой степени. Иными словами потребуем, чтобы для функции

$P(x) = \sum_{j=0}^m a_j x^j$ приближенное выражение (5.4) было точным.

Для этого необходимо и достаточно, чтобы для любой сте-

пени коэффициенты при a_j были равны. Поскольку $(x^j)^{(k)} = j(j-1)\dots(j-k+1)x^{j-k}$, получаем линейную систему уравнений относительно C_i :

$$\sum_{i=1}^N C_i x_i^j = j(j-1)\dots(j-k+1)x_0^{j-k}, \quad j=0, 1, \dots, m.$$

Если $m = N-1$, то число уравнений равно числу неизвестных. Можно показать, что определитель такой системы не обращается в нуль. Очевидно, что $m \geq k$, $N \geq k+1$. При этом точность вычисления производной имеет порядок $O(h^{m+1-k})$, хотя при определенном положении узлов (обычно это симметричное положение относительно точки x_i) порядок многочлена, а следовательно, и точность можно повысить на единицу.

Пример 5.1. Задана функция $f(x) = \sin(x)$, $x \in [0, 1]$. Вычислить значение производной на правой границе (в точке $x = 1$) с первым и со вторым порядком точности ($h = 0,1$).

Решение. Для вычисления производной составим таблицу значений функции

i	N	$N-1$	$N-2$
x	1	0.9	0.8
f	0.841	0.783	0.717

Точное значение производной $f'(x_N) = \cos(x_N) = 0.540$.

Находим производную по формуле «разность назад»:

$$f'(x_N) = \frac{f_N - f_{N-1}}{h} = \frac{0.841 - 0.783}{0.1} = 0.581.$$

Как было показано выше, формула имеет первый порядок точности. Для вычисления производной со вторым порядком воспользуемся методом неопределенных коэффициентов. Обозначим x на правой границе через x_N . Требуется построить метод второго порядка, поэтому необходимо использовать три точки x_N , x_{N-1} , x_{N-2} , ($x_{N-2} = x_N - 2h$, $x_{N-1} = x_N - h$), тогда

$$f'(x_N) = C_1 f(x_{N-2}) + C_2 f(x_{N-1}) + C_3 f(x_N).$$

Степень полинома будет $m = N - 1$

$$P(x) = a_0 + a_1 x + a_2 x^2, \quad P'(x_N) = a_1 + 2a_2 x_N$$

Подставляем $P(x)$ в выражение для $f'(x_N)$

$$f'(x_N) = C_1(a_0 + a_1x_{N-2} + a_2x_{N-2}^2) + \\ + C_2(a_0 + a_1x_{N-1} + a_2x_{N-1}^2) + C_3(a_0 + a_1x_N + a_2x_N^2).$$

Собираем коэффициенты при a_0, a_1, a_2 и, учитывая, что эти коэффициенты должны быть равны в выражении для $P'(x_N)$ и $f'(x_N)$, получаем СЛАУ.

$$C_1 + C_2 + C_3 = 0, (-2h)C_1 + (-h)C_2 = 1, (4h^2)C_1 + (h^2)C_2 = 0.$$

Отсюда $C_1 = \frac{1}{2h}$, $C_2 = -4\frac{1}{2h}$, $C_3 = 3\frac{1}{2h}$, тогда

$$f'(x_N) \approx \frac{3f_N - 4f_{N-1} + f_{N-2}}{2h} = \frac{3 \cdot 0.841 - 4 \cdot 0.783 + 0.717}{2 \cdot 0.1} = 0.542.$$

Видно, что формула второго порядка гораздо точнее формулы первого порядка.

С помощью MathCAD можно вычислять производные скалярных функций любого количества аргументов, от 0-го до 5-го порядка включительно. И функции, и аргументы могут быть как действительными, так и комплексными. Невозможно дифференцирование функций только вблизи точек их сингулярности.

Вычислительный процессор MathCAD обеспечивает превосходную точность численного дифференцирования. При этом символьный процессор позволяет вычислять производные громоздких функций, поскольку, в отличие от всех других операций, символьное дифференцирование выполняется успешно для подавляющего большинства аналитически заданных функций.

Первая производная

Для того чтобы продифференцировать функцию $f(x)$ в некоторой точке:

1. Определяем точку x , в которой будет вычислена производная, например, $x:=1$.
2. Вводим оператор дифференцирования нажатием кнопки Derivative (Производная) на панели Calculus (Вычисления) или вводим с клавиатуры вопросительный знак $<?>$.
3. В появившихся местозаполнителях вводим функцию, зависящую от аргумента x , т.е. $f(x)$, и имя самого аргумента x .

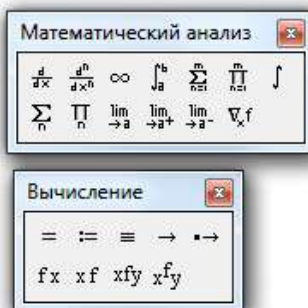
4. Вводим оператор численного $\langle \Rightarrow \rangle$ или $\langle \rightarrow \rangle$ символьного вывода для получения ответа.

Пример 5.2. На листинге ниже приведен пример численного и символьного дифференцирования:

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

$$df(x) := \frac{d}{dx} f(x) \rightarrow -\sin(x)$$

$$df\left(\frac{\pi}{2}\right) = -1$$



Для численного дифференцирования MathCAD применяет довольно сложный алгоритм, вычисляющий производную с колоссальной точностью до 7-8 знака после запятой. Этот алгоритм (метод Рундлера) описан во встроенной справочной системе MathCAD, доступной через меню **Help** (Справка). Погрешность дифференцирования не зависит от констант TOL или CTOL, в противоположность большинству остальных численных методов, а определяется непосредственно алгоритмом.

Производные высших порядков

MathCAD позволяет численно определять производные высших порядков от 0-го до 5-го включительно. Чтобы вычислить производную функции $f(x)$ N -го порядка в точке x , нужно проделать те же самые действия, что и при взятии первой производной, за тем исключением, что вместо оператора производной необходимо применить оператор N -й производной (Nth Derivative). Этот оператор вводится с той же панели Calculus (Вычисления), либо с клавиатуры $\langle \text{Ctrl} \rangle + \langle ? \rangle$, и содержит еще два местозаполнителя, в которые следует поместить число N . «Производная» при $N = 0$ по определению равна самой функции, при $N = 1$ получается обычная первая производная. Важно перед оператором дифференцирования не забывать при-

сваивать аргументу функции значение, для которого будет вычисляться производная.

Чтобы вычислить производную порядка выше 5-го, следует последовательно применить несколько раз оператор N -й производной подобно тому, как вводятся операторы кратного интегрирования. Однако для символьных вычислений этого не требуется — символьный процессор умеет считать производные порядка выше 5-го.

Пример 5.3. Рассмотрим образец численного и символьного вычисления второй производной:

$$z(x) := \sin(x) \cdot x^4$$

$$dz2(x) := \frac{d^2}{dx^2} z(x) \rightarrow 2 \cdot \sin(x) - x^2 \cdot \sin(x) + 4 \cdot x \cdot \cos(x)$$

$$dz2\left(1 + \frac{\pi}{3}\right) = -5.702$$

Расчет производных высших порядков производится тем же вычислительным методом Риддера, что и расчет первых производных. Причем для первой производной этот метод обеспечивает точность до 7-8 значащих разрядов числа, а при повышении порядка производной на каждую единицу точность падает примерно на один разряд.

Частные производные

С помощью символьного и численного процессоров MathCAD можно вычислять производные функций любого количества и по разным аргументам, т.е. *частные* производные. Чтобы вычислить частную производную, необходимо ввести оператор производной с панели Calculus (Вычисления) и в соответствующем местозаполнителе напечатать имя переменной, по которой должно быть осуществлено дифференцирование.

Пример 5.4. Символьное и численное вычисление частных производных в пакете MathCAD выглядит следующим образом:

$$f(x, y) := x^{2y} + \cos(x) \cdot y$$

$$\frac{\partial}{\partial x} f(x, y) \rightarrow 2 \cdot x^{2y} \cdot \frac{y}{x} - \sin(x) \cdot y$$

$$\frac{\partial}{\partial y} f(x, y) \rightarrow 2 \cdot x^{2y} \cdot \ln(x) + \cos(x)$$

$$x := 1 \quad y := 0.1$$

$$\frac{\partial}{\partial y} f(x, y) = 0.54 \quad \frac{\partial}{\partial y} f(x, y) \rightarrow \cos(1)$$

Как показано в примере, в первой строке определена функция двух переменных, а в двух следующих строках символьным образом вычислены ее частные производные по обеим переменным x и y соответственно. Чтобы определить частную производную численным методом, необходимо предварительно задать значения всех аргументов. При этом частные производные высших порядков рассчитываются точно так же, как и обычные производные высших порядков.

Также имеется возможность выбора формы записи частной производной, причем вид записи не влияет на вычисления, а служит лишь более привычной формой представления расчетов. Для того чтобы изменить вид оператора дифференцирования на представление частной производной, следует:

1. Вызвать контекстное меню из области оператора дифференцирования нажатием правой кнопки мыши.
2. Выбрать в контекстном меню верхний пункт View Derivative As (Показывать производную как).
3. В появившемся подменю выбрать пункт Partial Derivative (Частная производная).

Чтобы вернуть вид производной, принятый по умолчанию, выберите в подменю пункт Default (По умолчанию) либо, для представления в обычном виде — Derivative (Производная).

5.2. Численное интегрирование

Интегрирование функций является составной частью многих научных и технических задач. Поскольку аналитическое интег-

рирование не всегда возможно, используют различные методы численного интегрирования.

Математическая постановка задачи: необходимо найти значение определенного интеграла

$$I = \int_a^b f(x)dx,$$

где a, b – конечны, $f(x)$ – непрерывна на $[a, b]$. Из математического анализа хорошо известна формула Ньютона–Лейбница для нахождения определенного интеграла с помощью первообразных. Однако первообразную можно найти не для всех функций, для некоторых элементарных функций первообразной вообще не существует. Поэтому строятся формулы приближенного интегрирования, которые называются *квадратурными формулами*.

При решении практических задач часто бывает, что интеграл неудобно или невозможно взять аналитически: он может не выражаться в элементарных функциях, подынтегральная функция может быть задана в виде таблицы и пр. В таких случаях применяют методы численного интегрирования.

Известно, что определенный интеграл численно равен значению площади фигуры, ограниченной подынтегральной функцией, осью x , прямыми $x=a$ и $x=b$ (рис. 5.2). Общий подход к вычислению интеграла численными методами сводится к нахождению этой площади. Чаще всего интервал $[a, b]$ разбивают на множество меньших интервалов. Находят приблизительно площади каждой части и потом сумму всех площадей.

Хорошо известно, что для функций, допускающих на промежутке $[a, b]$ конечное число точек разрыва первого рода, такое значение существует единственно и может быть формально получено по определению:

$$I = \lim_{n \rightarrow \infty} \sum_{i=1}^n f(\xi_i)(x_i - x_{i-1}), \quad (5.5)$$

где x_i — произвольная упорядоченная система точек отрезка $[a, b]$, а ξ_i — произвольная точка элементарного промежутка $[x_{i-1}, x_i]$.

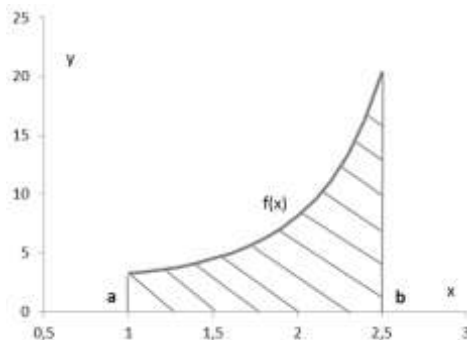


Рис. 5.2. К объяснению геометрического смысла определенного интеграла

Простейшие квадратурные формулы выводятся непосредственно из определения интеграла, т.е. из представления (5.5). Зафиксировав некоторое $n \geq 1$, будем иметь

$$I \approx \sum_{i=1}^n f(\xi_i)(x_i - x_{i-1}) \quad (5.6)$$

Это приближенное равенство назовем *общей формулой прямоугольников*. Геометрически площадь криволинейной трапеции приближенно заменяется площадью ступенчатой фигуры, составленной из прямоугольников, основаниями которым служит $[x_{i-1}, x_i]$, высотами - ординаты $f(\xi_i)$.

Рассмотрим ряд наиболее употребительных квадратурных формул. Условимся в дальнейшем пользоваться равномерным разбиением отрезка $[a, b]$ на n частей точками x_i с шагом $h = \frac{b-a}{n}$, полагая $x_0 = a$, $x_i = x_{i-1} + h$, $x_n = b$. При таком разбиении формула (5.6) приобретает вид

$$I \approx h \sum_{i=1}^n f(\xi_i), \quad \xi_i \in [x_{i-1}, x_i]. \quad (5.7)$$

Теперь дело за фиксированием точек $f(\xi_i)$ на элементарных отрезках $[x_{i-1}, x_i]$. Рассмотрим три случая.

1) Положим $\xi_i = x_{i-1}$. Тогда из (5.7) получаем

$$I \approx h \sum_{i=1}^n f(x_{i-1}) = h \sum_{i=1}^{n-1} f(x_i).$$

Эта формула называется формулой левых прямоугольников. Ее геометрическая интерпретация приведена на рис. 5.3.

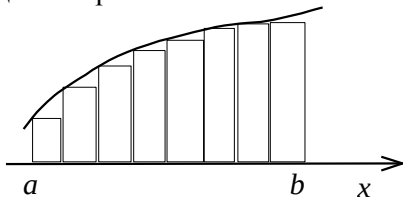


Рис. 5.3. Геометрическая интерпретация формулы левых прямоугольников

2) Пусть $\xi_i = x_i$. Тогда имеем

$$I \approx h \sum_{i=1}^n f(x_i).$$

Это формула *правых* прямоугольников.

3) Фиксируем $\xi_i = \frac{1}{2}(x_i + x_{i-1})$. В результате получим квадратурную формулу *средних* прямоугольников:

$$I \approx h \sum_{i=1}^{n-1} f\left(x_i + \frac{h}{2}\right) = h \sum_{i=1}^n f\left(x_i - \frac{h}{2}\right).$$

Покажем, что эта формула имеет второй порядок точности. Рассмотрим сначала вычисление интеграла на отрезке $[-h/2, h/2]$,

$$\int_{-h/2}^{h/2} f(x) dx \approx hf_0,$$

где $f_0 = f(0)$. Пусть

$$F(x) = \int_0^x f(x) dx, \quad F_{\pm 1/2} = F(\pm h/2);$$

$$\int_{-h/2}^{h/2} f(x) dx = F(h/2) - F(-h/2).$$

Разлагая в ряд Тейлора с остаточным членом в форме Лагранжа, имеем

$$F_{\pm 1/2} = \pm \frac{h}{2} f_0 + \frac{h^2}{8} f_0' \pm \frac{h^3}{48} f''(\xi_{\pm}),$$

где $\xi_{\pm}$ – некоторые точки такие, что $-h/2 < \xi_- < \xi_+ < h/2$. Тогда

$$\int_{-h/2}^{h/2} f(x) dx = hf_0 + \frac{h^3}{24} f''(\xi), \quad |\xi| < \frac{h}{2}.$$

Для всего интервала $[a, b]$ формула имеет вид:

$$\int_a^b f(x) dx = h \sum_{i=0}^n f_{i+1/2} + \frac{(b-a)}{24} h^2 f''(\xi), \quad a < \xi < b.$$

Таким образом, ошибка численного интегрирования по формуле средних прямоугольников убывает пропорционально квадрату шага h , т.е. формула имеет второй порядок точности. Нетрудно убедиться, что погрешность численного интегрирования по формулам левых и правых прямоугольников убывает лишь по линейному закону.

Подстановка в интеграл $\int_a^b f(x) dx$ вместо функции $f(x)$ ее ин-

терполяционного многочлена Лагранжа той или иной степени приводит к семейству квадратурных формул, называемых *формулами Ньютона–Котеса*. Однако использование в этих формулах многочленов высоких порядков может быть оправдано только для достаточно гладких подынтегральных функций. Чаще используются квадратурные правила, получающиеся путем дробления промежутка интегрирования на большое число мелких частей. Интегрирование на каждой из частей производится с помощью однотипных простейших формул невысокого порядка. Приведем два таких правила – трапеций и Симпсона.

Простейшая *формула трапеций* получается, если на каждом отрезке $[x_{i-1}, x_i]$ участок кривой (f_{i-1}, f_i) интерполировать линейной зависимостью. Эта формула имеет второй порядок точности и может быть записана в виде

$$\int_a^b f(x)dx \approx h \left(\frac{f_0 + f_n}{2} + \sum_{i=1}^{n-1} f_i \right).$$

Если на каждом отрезке $[x_{i-1}, x_i, x_{i+1}]$ участок кривой (f_{i-1}, f_i, f_{i+1}) интерполировать параболой, то придем к *формуле Симпсона*, имеющей четвертый порядок точности:

$$\int_a^b f(x)dx \approx (f_0 + 4f_1 + 2f_2 + 4f_3 + \dots + 2f_{n-2} + 4f_{n-1} + f_n).$$

Пример 5.4. Вычислить интеграл с помощью метода трапеций

$$n = 4, I = \int_{2.0}^{3.6} \frac{x+2}{\sqrt{1+x^2}} dx.$$

Решение при помощи пакета MathCAD в приведенном ниже листинге включает задание числа отрезков разбиения n , вычисление шага интегрирования h , задание дискретной переменной i и массива координат узлов разбиения x_i , описание подынтегральной функции и расчет значения интеграла по формуле трапеций:

$n := 4$

$$h := \frac{3.6 - 2}{4} = 0.4$$

$i := 0..n$

$$x_i := 2 + i \cdot h$$

$$f(x) := \frac{x+2}{\sqrt{1+x^2}}$$

$$I := h \cdot \left(\frac{f(x_0) + f(x_n)}{2} + \sum_{i=1}^{n-1} f(x_i) \right) = 2.601$$

Для проверки можно вычислить интеграл при помощи встроенных возможностей MathCAD и убедиться в соответствии значений:

$$\int_2^{3.6} f(x) dx = 2.599 \cdot$$

Оператор интегрирования

Интегрирование устроено в MathCAD по принципу «как пишется, так и вводится». Чтобы вычислить определенный интеграл, следует напечатать его обычную математическую форму в документе. Делается это с помощью панели Calculus (Вычисления) нажатием кнопки со значком интеграла или вводом с клавиатуры сочетания клавиш <Shift>+<7> (или символа «&», что то же самое). Появится символ интеграла с несколькими местозаполнителями, в которые нужно ввести нижний и верхний интервалы интегрирования, подынтегральную функцию и пере-

менную интегрирования:

Численное и символьное вычисление определенного интеграла

$$\int_0^{\pi} \sin(x) dx = 2 \qquad \int_0^{\pi} \sin(x) dx \rightarrow 2$$

Интегрирование функции двух переменных по разным переменным

$$\alpha := 2 \quad \int_0^{\pi} \alpha \sin(x) dx = 4$$

$$x := 1 \quad \int_0^{\pi} \alpha \sin(x) d\alpha = 42.074$$

Использование оператора интегрирования в функции пользователя

$$g(\alpha) := \int_0^{\pi} \alpha \cdot \sin(x) dx$$

$$i := 1..5$$

$$g(i) =$$

2
4
6
8
10

Чтобы получить результат интегрирования, следует ввести равенства или символьного равенства. В первом случае интегрирование будет проведено численным методом, во втором – в случае успеха, будет найдено точное значение интеграла с помощью символьного процессора MathCAD. Ниже показаны оба способа. Конечно, символьное интегрирование возможно только для небольшого круга подынтегральных функций.

Пример 5.5. Продемонстрируем нахождение интеграла в пакете MathCAD численными методами, описанными в разделе 5.2.

В качестве подынтегральной функции выберем $f(x)=\sin(x)$, интервал интегрирования $[0,\pi/2]$.

Точное решение в этом случае будет равно:

$$St := \int_0^{\frac{\pi}{2}} \sin(x) dx = 1$$

Сначала зададим пространственную сетку x_i и вычислим значения функции в узловых точках y_i :

$$f(x) := \sin(x) \quad x_{min} := 0 \quad x_{max} := \frac{\pi}{2}$$

$$N := 100 \quad i := 0..N \quad \Delta x := \frac{x_{max} - x_{min}}{N}$$

$$x_i := x_{min} + \Delta x \cdot i \quad y_i := f(x_i)$$

Вычислим интегралы по формулам правых прямоугольников (**S_pr**), левых прямоугольников (**S_l**) и трапеций (**S_tr**):

$$S_{pr} := \left(\sum_{i=1}^N y_i \right) \cdot \Delta x = 1.00783 \quad S_{pr} - St = 7.833 \times 10^{-3}$$

$$S_l := \left(\sum_{i=0}^{N-1} y_i \right) \cdot \Delta x = 0.99213 \quad S_l - St = -7.875 \times 10^{-3}$$

$$S_{tr} := \left(\frac{1}{2} \cdot f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2} \cdot f(x_N) \right) \cdot \Delta x = 0.99998 \quad S_{tr} - St = -2.056 \times 10^{-5}$$

Сравнение с точным решением результатом помогает оценить погрешность каждого из методов.

Результат численного интегрирования – это не точное, а приближенное значение интеграла, определенное с погрешностью, которая зависит от встроенной константы TOL. Чем она меньше, тем с лучшей точностью будет найден интеграл, но и больше времени будет затрачено на расчеты. По умолчанию TOL = 0.001. Для того чтобы ускорить вычисления, можно установить меньшее значение TOL. Кроме того, пользователь имеет возможность выбирать сам алгоритм численного интегрирования. Для этого:

1. Щелкните правой кнопкой мыши в любом месте на левой части вычисляемого интеграла.
2. В появившемся контекстном меню выберите один из четырех численных алгоритмов.

При этом возможны четыре численных метода интегрирования:

Romberg (Ромберга) – для большинства функций, не содержащих особенностей;

Adaptive (Адаптивный) – для функций, быстро меняющихся на интервале интегрирования;

Infinite Limit (Бесконечный предел) – для интервалов с бесконечными пределами;

Singular Endpoint – для интегралов с сингулярностью на конце. Модифицированный алгоритм Ромберга для функций, не определенных на одном или обоих концах интервала интегрирования.

Итерационный алгоритм Ромберга (применяется, если подынтегральная функция не меняется на интервале интегрирования слишком быстро и не обращается на нем в бесконечность). Его основные идеи:

1. Сначала строится несколько интерполирующих полиномов, которые заменяются на интервале интегрирования подынтегральную функцию $f(x)$. В качестве первой итерации полиномы вычисляются по 1, 2 и 4 интервалам. Например, первый полином, построенный по 1 интервалу, – это прямая линия, проведенная через две граничные точки интервала интегрирования, второй – квадратичная парабола и т.д.
2. Интеграл от каждого полинома с известными коэффициентами легко вычисляются аналитически. Таким образом, определяется последовательность интегралов от интерполирующих полиномов: $I_1, I_2, I_4, \dots$. Например, по правилу трапеций $I_1 = (b - a) \cdot (f(a) + f(b)) / 2$ и т.д.
3. Из-за интерполяции по разному числу точек вычисленные интегралы $I_1, I_2, \dots$ несколько отличаются друг от друга. Причем чем больше точек используется для интерполяции, тем интеграл от интерполяционного полинома ближе к истинному интегралу, стремясь к нему в пределе бесконечного числа точек. Поэтому определенным образом осуществляется экстраполяция последовательности $I_1, I_2, I_4, \dots$ до нулевой ширины элементарного интервала. Результат этой экстраполяции J принимается за приближение к вычисляемому интегралу.
4. Осуществляется переход к новой итерации с помощью еще более частого разбиения интервала интегрирования, добавления нового члена последовательности интерполирующих полиномов и вычисления нового (N -го) приближения **Ромберга** J^N .
5. Чем больше количество точек интерполяции, тем ближе очередное приближение **Ромберга** к вычисляемому интегралу и, соответственно, тем меньше оно отличается от приближения предыдущей итерации. Как только разница между двумя последними итерациями $|J^N - J^{N-1}|$ становится мень-

ше погрешности TOL или меньше $TOL \cdot |J^N|$, итерации прерываются и J^N появляется на экране в качестве результата интегрирования.

О расходящихся интегралах

Если интеграл расходится (равен бесконечности), то вычислительный процессор MathCAD может выдать сообщение об ошибке, выделив при этом оператор интегрирования красным цветом. Чаще всего ошибка будет иметь тип «**Found a number with a magnitude greater than 10^307**» (Найдено число, превышающее значение 10^{307}) или «**Can't converge to a solution**» (Не сходится к решению), как, например, при попытке вычислить

интеграл $\int_0^{\infty} \frac{1}{\sqrt{x}} dx$. Тем не менее, символьный процессор справляется с этим интегралом, совершенно правильно находя его бесконечное значение.

Кратные интегралы

Для вычисления кратного интеграла:

1. Вводится, как обычно, оператор интегрирования.
2. В соответствующих местозаполнителях вводится имя первой переменной интегрирования и пределы интегрирования по этой переменной.
3. На месте ввода подынтегральной функции вводится еще один оператор интегрирования.
4. Точно также вводится вторая переменная, пределы интегрирования и подынтегральная функция (если интеграл двукратный) или следующий оператор интегрирования (если более чем двукратный) и т.д., пока выражение с многократным интегралом не будет введено окончательно.

Ниже приведен пример символьного и численного расчета двукратного интеграла в бесконечных пределах. При этом символьный процессор вычисляет точное значение интеграла π , а вычислительный определяет его приближенно и выдает в виде числа 3,142. Пример символьного и численного вычисления кратного интеграла:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)} dx dy \rightarrow \pi$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)} dx dy = 3.142 \blacksquare$$

Пример символьного вычисления кратного интеграла:

$$\int_a^b \int_{-1}^1 x + y^3 dx dy \rightarrow \frac{1}{2} \cdot b^4 - \frac{1}{2} \cdot a^4 \quad \int_a^b \int_{-1}^1 x + y^3 dy dx \rightarrow b^2 - a^2 .$$

Глава 6. Решение обыкновенных дифференциальных уравнений

Зачастую при рассмотрении прикладных задач механики, физики, химии и техники приходится решать обыкновенные дифференциальные уравнения (ОДУ), представляющие собой математическое упрощение исходной постановки. ОДУ описывают движение системы взаимодействующих материальных точек, химической кинетики, электрических цепей, моделируют статический прогиб упругого стержня (сопротивление материалов) и многие другие процессы. Ряд важных задач для уравнений в частных производных также сводится к задачам для ОДУ. Так бывает, если многомерная задача допускает разделение переменных, например, задачи на нахождение собственных колебаний упругих балок и мембран простейшей формы.

6.1. Задача Коши

6.1.1. Постановка задачи

Требуется найти решение ОДУ первого порядка

$$\frac{dy}{dx} = f(x, y). \quad (6.1)$$

Известно, что общее решение (6.1) содержит произвольную константу C , т.е. является однопараметрическим семейством интегральных кривых $y(x) = \int f(x, y) dx + C$.

Для выбора конкретной интегральной кривой следует определить значение константы C , для чего достаточно задать при каком-либо значении $x = x_0$ значение

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

Поэтому задача Коши, или задача с начальными данными, позволяющая получить единственное решение уравнения (6.1), формулируется так: найти $y(x)$ - решение уравнения (6.1) с начальным условием (6.2).

В случае точного решения ОДУ мы получаем аналитическое представление искомой функции. Несмотря на внешнюю простоту уравнения (6.1) решить его аналитически, т.е. найти

общее решение $y = y(x, C)$ с тем, чтобы затем выделить из него интегральную кривую $y = y(x)$, проходящую через точку (x_0, y_0) , удастся лишь для некоторых специальных типов уравнений. Поэтому большое значение приобретают приближенные способы решения начальных задач ОДУ. При численном же решении мы будем искать приближенное решение в узлах расчетной сетки $x_i = x_0 + ih$, $i = 0, 1, \dots, n$ с шагом $h = \frac{x_n - x_0}{n}$. Т.е.

вместо непрерывной зависимости $y(x)$ мы найдем приближенные значения в узлах сетки $y_i = y(x_i)$.

Для построения численных методов решения ОДУ проинтегрируем уравнение на отрезке $[x_i, x_{i+1}]$. Получим

$$y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(x, y) dx. \quad (6.3)$$

Для того чтобы найти все значения y_i , нужно каким-то образом вычислить интеграл, стоящий в правой части (6.3). Применяя различные квадратурные формулы, будем получать методы решения задачи (6.1), (6.2) разного порядка точности.

6.1.2. Метод Эйлера

Если для вычисления интеграла в (6.3) воспользоваться простейшей формулой левых прямоугольников первого порядка

$$\int_{x_i}^{x_{i+1}} f(x, y) dx = hf(x_i, y_i),$$

то получим **явную формулу Эйлера**:

$$y_{i+1} = y_i + hf(x_i, y_i), \quad i = 0, 1, \dots, n-1, \quad (6.4)$$

имеющую первый порядок аппроксимации.

Реализация метода. Поскольку $x_0, y_0, f(x_0, y_0)$ известны, последовательно применяя (6.4), определим все y_i : $y_1 = y_0 + hf(x_0, y_0)$, $y_2 = y_1 + hf(x_1, y_1)$, ...

Геометрическая интерпретация метода Эйлера (рис. 6.1.):

Пользуясь тем, что в точке x_0 известно решение $y(x_0) = y_0$ и значение его производной $y'(x_0) = \left. \frac{dy}{dx} \right|_{x=x_0} = f(x_0, y_0)$, можно за-

писать уравнение касательной к графику искомой функции $y = y(x)$ в точке $f(x_0, y_0)$: $y = y_0 + f(x_0, y_0)(x - x_0)$. При достаточно малом шаге h ордината $y_1 = y_0 + hf(x_0, y_0)$ этой касательной, полученная подстановкой в правую часть значения $x_1 = x_0 + h$, должна мало отличаться от ординаты $y(x_1)$ решения $y(x)$ задачи Коши. Следовательно, точка (x_1, y_1) пересечения касательной с прямой $x = x_1$ может быть приближенно принята за новую начальную точку. Через эту точку снова проведем прямую $y = y_1 + f(x_1, y_1)(x - x_1)$, которая приближенно отражает поведе-

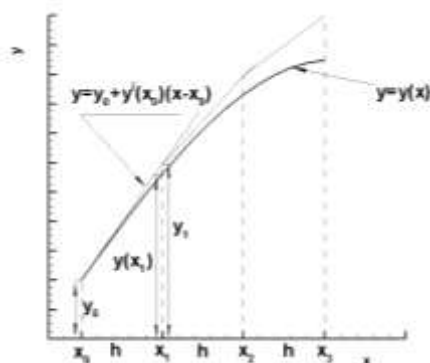


Рис. 6.1. Геометрическая интерпретация метода Эйлера

лучим неявный метод Эйлера

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1}), \quad i = 0, 1, \dots, n-1. \quad (6.5)$$

Этот метод называют неявным, поскольку для вычисления неизвестного значения $y_{i+1} \approx y(x_{i+1})$ по известному значению $y_i \approx y(x_i)$ требуется решать уравнение, в общем случае нели-

ние касательной к $y(x)$ в точке (x_1, y_1) . Подставляя сюда $x_2 = x_1 + h$ (т.е. пересечение с прямой $x = x_2$), получим приближенное значение $y(x)$ в точке x_2 : $y_2 = y_1 + hf(x_1, y_1)$ и т.д. В итоге для i -ой точки получим формулу Эйлера.

Если в (6.3) использовать формулу правых прямоугольников, то по-

нейное. Неявный метод Эйлера также имеет первый порядок аппроксимации.

6.1.3. Модифицированный метод Эйлера

В данном методе вычисление y_{i+1} состоит из двух этапов:

$$\begin{aligned}\tilde{y}_{i+1} &= y_i + hf(x_i, y_i), \\ y_{i+1} &= y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, \tilde{y}_{i+1})].\end{aligned}\quad (6.6)$$

Данная схема называется также методом предиктор-корректор. Это английское название, означающее «предсказать-исправить». Действительно, на первом этапе, совпадающим с формулой Эйлера, приближенное значение предсказывается с первым порядком точности, а на втором этапе это значение корректируется, так что в результате схема имеет второй порядок точности.

6.1.4. Методы Рунге-Кутты

Идея построения явных методов Рунге-Кутты p -го порядка заключается в получении приближений к значениям $y(x_{i+1})$ по формуле вида $y_{i+1} = y_i + h\varphi(x_i, y_i, h)$, где

$$\begin{aligned}\varphi(x_i, y_i, h) &= \sum_{n=1}^q c_n k_n^i(h) \\ k_1^i(h) &= f(x_i, y_i), \\ k_2^i(h) &= f(x_i + \alpha_2 h, y_i + \beta_{21} k_1^i(h)), \\ k_3^i(h) &= f(x_i + \alpha_3 h, y_i + \beta_{31} k_1^i(h) + \beta_{32} k_2^i(h)), \\ &\dots \\ k_q^i(h) &= f(x_i + \alpha_q h, y_i + \beta_{q1} k_1^i(h) + \dots + \beta_{q, q-1} k_{q-1}^i(h)).\end{aligned}$$

Здесь α_n, β_{nj} , $0 < j < n \leq q$ – некоторые фиксированные числа (параметры), которые подбирают таким образом, чтобы получить нужный порядок аппроксимации p . Как правило, для каждого p существует не одна схема Рунге-Кутты порядка p , а целое

параметрическое семейство. Так, схемы Рунге-Кутта **второго порядка точности** образуют однопараметрическое семейство

$$\begin{aligned} k_1^i &= f(x_i, y_i), \quad k_2^i = f\left(x_i + \frac{h}{2a}, y_i + \frac{h}{2a} k_1^i\right) \\ y_{i+1} &= y_i + h[(1-a)k_1^i + ak_2^i]. \end{aligned} \quad (6.7)$$

Выделим из семейства методов (6.7) два наиболее простых и часто используемых частных случая. При $a = 1/2$ получаем формулы

$$\begin{aligned} k_1^i &= f(x_i, y_i), \quad k_2^i = f(x_i + h, y_i + hk_1^i) \\ y_{i+1} &= y_i + \frac{h}{2}[k_1^i + k_2^i], \quad i = 0, 1, 2, \dots \end{aligned} \quad (6.8)$$

которые совпадают с формулами модифицированного метода Эйлера (6.6). При $a=1$ выводим новый простой метод

$$\begin{aligned} k_1^i &= f(x_i, y_i), \quad k_2^i = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2} k_1^i\right), \\ y_{i+1} &= y_i + hk_2^i, \quad i = 0, 1, 2, \dots \end{aligned}$$

который называется методом средней точки.

Схема Рунге-Кутта четвертого порядка точности. При $p=4$ можно получить один из вариантов метода:

$$\begin{aligned} k_1 &= f(x_i, y_i), \quad k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right), \\ k_3 &= f\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}\right), \quad k_4 = f(x_i + h, y_i + hk_3), \\ y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad i = 0, 1, 2, \dots \end{aligned} \quad (6.9)$$

6.1.5. Функции MathCAD для решения ОДУ

Стандартные функции

В пакете MathCAD существуют инструменты для решения ОДУ, основанные на применении различных численных методов. Для решения задачи Коши используется встроенная проце-

дура с названием `Odesolve`. При этом стоит учитывать некоторые ограничения, налагаемые на вид ОДУ: член с высшей производной должен быть линейным и количество начальных условий должно равняться порядку ОДУ.

При использовании данной процедуры необходимо создать блок решения, который заканчивается вызовом процедура-функция `Odesolve`. Блок начинается с задания самого ОДУ и начальных условий, чему предшествует ключевое слово `Given`.

Пример 6.1. Дано ОДУ первого порядка, для него поставлена задача Коши:

$$\frac{du}{dt} = \frac{3t^2}{u}, u(t_0) = 1, t \in (1..20).$$

`Given`

$$\frac{d}{dt}u(t) = \frac{3 \cdot t^2}{u(t)}$$

$$u(1) = 1$$

$$u := \text{Odesolve}(t, b, 10)$$

Рис. 6.2. Пример использования ПФ *Odesolve*

Необходимо найти решение на заданном интервале. Блок решения будет выглядеть, как показано на рис. 6.2.

Для данной задачи существует аналитическое решение, которое просто находится:
 $u = \sqrt{2t^3 - 1}.$

При создании блока решения необходимо использовать логическую операцию равенства (из панели «Булева алгебра») для определения самого ОДУ и его начальных условий (жирное равенство). В данном примере в качестве входных параметров `Odesolve` используются: t — аргумент функции, b — правая граница интервала по t , и 10 — число шагов. Последний параметр указывать не обязательно, в этом случае пакет сам выберет оптимальное число разбиений отрезка.

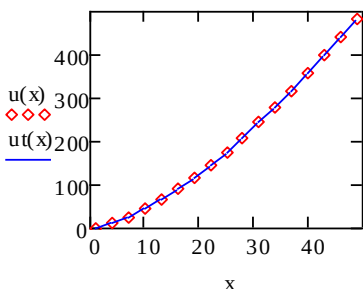


Рис. 6.3. Сравнение точного (сплошная линия) и приближенного (маркеры) решения для примера 6.1

По умолчанию для решения ОДУ используется метод Адамса [7] или его модификация BDF для случая жестких систем, о которых будет сказано ниже. Также для решения можно использовать метод Рунге-Кутты четвертого порядка точности с постоянным или адаптивным шагом. Имеется опция выбора метода Радау для решения жестких систем.

Чтобы сменить используемый метод решения либо узнать, какой метод используется в текущий момент, необходимо нажать правой кнопкой мыши на операторе `Odesolve`, и выпадающее меню отобразит доступные опции решателя.

В результате решения мы получаем дискретную функцию, которую можно визуализировать графически и сравнить с точным решением, выписанном выше, как показано на рис. 6.3. Здесь решение получено с помощью метода Рунге-Кутты с постоянным шагом.

Пользовательские функции

Для реализации методов решения задачи Коши, не включенных в стандартные функции MathCAD, например, метода Эйлера, удобно воспользоваться возможностями программирования в среде MathCAD. При этом создаваемая процедура-функция (ПФ) будет работать для различных входных условий (границ отрезка, начальных данных, видов функций в правой части ОДУ), и может быть отлажена (протестирована) на примере известного аналитического решения с тем, чтобы использоваться в дальнейшем для более сложных уравнений. Пример такой ПФ, реализующей метод Эйлера, приведен на рис. 6.4.

$$\text{Euler}(a, b, n, y_0) := \left| \begin{array}{l} h \leftarrow \frac{b - a}{n - 1} \\ u_1 \leftarrow y_0 \\ \text{for } i \in 1..n - 1 \\ \quad \left| \begin{array}{l} x_i \leftarrow a + (i - 1) \cdot h \\ u_{i+1} \leftarrow u_i + h \cdot f(x_i, u_i) \end{array} \right. \\ x_n \leftarrow b \\ u \end{array} \right.$$

Рис. 6.4. Процедура-функция, реализующая метод Эйлера

Входными условиями для этой ПФ с названием Euler служат границы a и b интервала изменения аргумента (для уравнения (6.1) это переменная x), количество разбиений отрезка и начальное условие y_0 (6.2). В первой строке процедуры-функции задается шаг изменения аргумента h , во второй строке — начальное условие, т.е.

значение искомой функции в первой точке. Отметим, что оператор ORIGIN:=1 в этом случае определен выше в документе, для совпадения индексов массива и вычисляемой дискретной функции. Если ORIGIN равен 0 (что происходит по умолчанию), необходимо изменить цикл программы соответствующим образом. В первом операторе тела цикла *for*, где переменная цикла соответствует номеру элемента массива искомой функции, вычисляется текущее значение аргумента x_i . Понятно, что аргумент в цикле будет меняться от a до $b-h$ с шагом h . Затем используется формула для расчета искомой функции в точке x_i по методу Эйлера (6.4). При этом функция $f(x, u)$, стоящая в правой части ОДУ (6.1), должна быть предварительно описана. В результате работы процедуры-функции Euler формируется вектор-столбец u , который и является выходным параметром ПФ.

Используя аналогичный алгоритм, можно запрограммировать любой из вышеприведенных методов решения задачи Коши.

Пример 6.2.

Как и в примере 6.1, задано ОДУ первого порядка, для него поставлена задача Коши:

$$\frac{du}{dt} = \frac{3t^2}{u}, \quad u(t_0) = 1, \quad t \in (1..2).$$

Необходимо найти решение при помощи метода Эйлера.

Листинг программы приведен ниже на рис. 6.5.

```

ORIGIN:= 1
f(t,u) :=  $\frac{3t^2}{u}$ 

Euler(a,b,n,y0) :=  $\left\{ \begin{array}{l} h \leftarrow \frac{(b-a)}{n-1} \\ u_1 \leftarrow y0 \\ \text{for } i \in 1..n-1 \\ \quad \left\{ \begin{array}{l} x_i \leftarrow a + (i-1) \cdot h \\ u_{i+1} \leftarrow u_i + h \cdot f(x_i, u_i) \end{array} \right. \\ x_n \leftarrow b \\ \text{augment}(x, u) \end{array} \right.$ 

analyt(a,b,n,y0) :=  $\left\{ \begin{array}{l} h \leftarrow \frac{(b-a)}{n-1} \\ u_1 \leftarrow y0 \\ \text{for } i \in 1..n \\ \quad \left\{ \begin{array}{l} x_i \leftarrow a + (i-1) \cdot h \\ u_i \leftarrow \sqrt{2 \cdot (x_i)^3 - 1} \end{array} \right. \\ \text{augment}(x, u) \end{array} \right.$ 

a := 1      b := 20      n := 11  y0 := 1
a1 := Euler(a,b,n,y0)    b1 := analyt(a,b,n,y0)

```

Рис. 6.5. Листинг документа MathCAD для решения примера 6.2. методом Эйлера

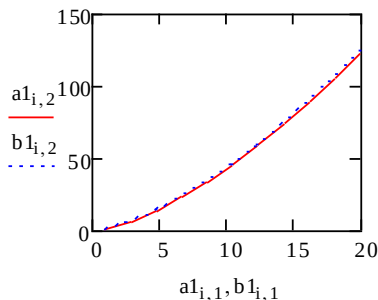


Рис. 6.6. Сравнение точного (штриховая линия) и приближенного (сплошная линия) решений примера 6.2

ные $a1$ и $b1$, соответственно. В последующем можно просмотреть их числовые значения, или, что более наглядно, построить график, как показано на рис. 6.6. Пунктирной линией приведено точное решение, сплошной линией — численное решение по методу Эйлера. Можно видеть, что ошибка растет по мере увеличения аргумента. Причины этого были описаны выше.

Процедура-функция `analyt` вычисляет аналитическое решение данного ОДУ. Здесь же приведена ПФ `Euler`, рассмотренная выше, с небольшими видоизменениями, сделанными для удобства вывода результатов. Результат расчета по методу Эйлера и точное решение записываются в переменные

6.1.6. Задача Коши для систем ОДУ и ОДУ высших порядков

Рассмотренные выше методы решения задачи Коши для одного уравнения могут быть использованы также для решения систем дифференциальных уравнений первого порядка и дифференциальных уравнений более высокого порядка.

Пусть задана задача Коши для системы двух уравнений первого порядка:

$$\begin{cases} \frac{dy}{dx} = \varphi(x, y, z), \\ \frac{dz}{dx} = \psi(x, y, z), \end{cases} \quad (6.10)$$

для которой известны начальные условия $y(x_0) = y_0$, $z(x_0) = z_0$. Обобщим формулы явного метода Эйлера (6.4) для этой системы, записав схему для каждого уравнения (6.10):

$$\begin{aligned} y_{i+1} &= y_i + h\varphi(x_i, y_i, z_i), \\ z_{i+1} &= z_i + h\psi(x_i, y_i, z_i). \end{aligned}$$

Модифицированный метод Эйлера (6.6) примет вид:

$$\begin{aligned} \tilde{y}_{i+1} &= y_i + h\varphi(x_i, y_i, z_i), \\ \tilde{z}_{i+1} &= z_i + h\psi(x_i, y_i, z_i), \\ y_{i+1} &= y_i + \frac{h}{2} [\varphi(x_i, y_i, z_i) + \varphi(x_i, \tilde{y}_{i+1}, \tilde{z}_{i+1})], \\ z_{i+1} &= z_i + \frac{h}{2} [\psi(x_i, y_i, z_i) + \psi(x_i, \tilde{y}_{i+1}, \tilde{z}_{i+1})], \end{aligned}$$

а схема Рунге-Кутты четвертого порядка точности (6.9):

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4), \\ z_{i+1} &= z_i + \frac{h}{6} (l_1 + 2l_2 + 2l_3 + l_4), \end{aligned}$$

$$k_1 = \varphi(x_i, y_i, z_i), l_1 = \psi(x_i, y_i, z_i),$$

$$k_2 = \varphi\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}, z_i + \frac{hl_1}{2}\right), l_2 = \psi\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}, z_i + \frac{hl_1}{2}\right),$$

$$k_3 = \varphi\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}, z_i + \frac{hl_2}{2}\right), l_3 = \psi\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}, z_i + \frac{hl_2}{2}\right),$$

$$k_4 = \varphi(x_i + h, y_i + hk_3, z_i + hl_3), l_4 = \psi(x_i + h, y_i + hk_3, z_i + hl_3).$$

Вычисления приближенного решения проводятся путем последовательного применения этих формул для каждого узла расчетной сетки.

Также аналогичным способом можно решить ОДУ высокого порядка. Например, рассмотрим задачу Коши для уравнения второго порядка

$$\frac{d^2 y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right), y(x_0) = y_0, \frac{dy}{dx}(x_0) = z_0.$$

Введем обозначение $z(x) = \frac{dy}{dx}$. Тогда исходная задача Коши для

уравнения второго порядка сводится к следующей задаче для системы двух ОДУ первого порядка:

$$\begin{cases} \frac{dy}{dx} = z, \\ \frac{dz}{dx} = f(x, y, z), \end{cases}$$

$$y(x_0) = y_0, z(x_0) = z_0.$$

Можно заметить, что эта запись эквивалентна системе (6.10) при $\varphi(x, y, z) = z$ и $\psi(x, y, z) = f(x, y, z)$. Таким образом, полученная система решается вышеописанным способом при помощи одного из методов решения задачи Коши.

Пример 6.3. Найти решение задачи Коши:

$$\frac{d^2 y}{dx^2} + 2 \frac{dy}{dx} + y(x) = x, y(0) = 1, \frac{dy}{dx}(0) = 0 \text{ на отрезке } [0, 1].$$

Для данного ОДУ известно точное решение:

$$y(x) = 3e^{-x} + 2xe^{-x} + x - 2.$$

Проверим, что точное решение удовлетворяет уравнению:

$$\frac{dy}{dx} = -e^{-x} - 2xe^{-x} + 1, \quad \frac{d^2y}{dx^2} = -e^{-x} + 2xe^{-x},$$

$$\begin{aligned} \frac{d^2y}{dx^2} + 2\frac{dy}{dx} + y(x) &= \\ &= -e^{-x} + 2xe^{-x} + 2(-e^{-x} - 2xe^{-x} + 1) + 3e^{-x} + 2xe^{-x} + x - 2 = x \\ y(0) &= 3e^0 + 0 + 0 - 2 = 1, \quad \frac{dy}{dx}(0) = -e^0 + 1 = 0 \end{aligned}$$

Далее будут рассмотрены примеры решения этой задачи встроенными средствами MathCAD и при помощи пользовательских функций.

6.1.7. Решение ОДУ 2-го порядка и систем ОДУ в MathCAD

Решение ОДУ второго порядка

Для решения ОДУ высокого порядка также применяется функция `Odesolve`, с созданием аналогичного блока решения. Для примера 6.3 блок расчета и результат в виде графика представлены на рис. 6.7.

При необходимости реализации метода, не являющегося стандартным, необходимо следовать процедуре, описанной выше, для решения систем ОДУ первого порядка. Для этого введем функцию $z(x) = \frac{dy}{dx}$ и получим следующую задачу Коши

для системы двух ОДУ первого порядка:

$$\frac{dy}{dx} = z, \quad \frac{dz}{dx} = -2z - y + x, \quad y(0) = 1, \quad z(0) = 0. \quad (6.11)$$

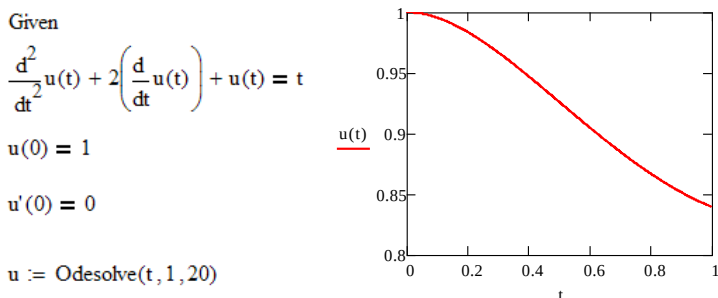


Рис. 6.7. Решение ОДУ второго порядка из примера 6.3 средствами MathCAD

Стандартные функции для решения систем ОДУ

Систему (6.11) можно решить при помощи метода Рунге-Кутты (rkfixed или Rkadapt) или Булирша-Штера (Bulstoer). Принципиальным отличием от решения одного ОДУ является использование векторов в качестве входных и выходных условий в ПФ.

С целью перехода к записи в обозначениях MathCAD, перепишем (6.11), обозначив функции y и z через u_1 и u_2 , элементы вектора решения. Тогда система примет вид

$$\frac{du_1}{dx} = u_2, \quad \frac{du_2}{dx} = -2u_2 - u_1 + x \quad \text{и} \quad \begin{aligned} u_1(0) &= 1, \\ u_2(0) &= 0. \end{aligned}$$

Из правых частей системы составим вектор D , а из начальных значений — вектор u_0 . Поскольку наша система состоит из двух уравнений, то и векторы будут иметь по два элемента. Пример решения системы (6.11) при помощи метода Рунге-Кутты с постоянным шагом представлен на рис. 6.8. Здесь описаны вектор правых частей D и вектор начальных данных u_0 , а затем получен вектор решения u .

$$D(x, u) := \begin{pmatrix} u_2 \\ -2 \cdot u_2 - u_1 + x \end{pmatrix}$$

$$u0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$u := \text{rkfixed}(u0, 0, 1, 10, D)$$

Рис. 6.8. Листинг решения системы (6.11) с помощью стандартной ПФ

Входными параметрами для ПФ `rkfixed` в данном примере служит вектор начальных данных `u0`, начало отрезка 0 и конец отрезка 1, количество разбиений отрезка 10 и вектор правых частей системы. Результатом является матрица `u`, состоящая из трех столбцов — аргумент, первая неизвестная функция,

вторая неизвестная функция — и 11 строк, поскольку заданное число разбиений 10. Понятно, что решение для первой искомой функции должно совпадать с решением на рис. 6.7.

Пользовательские функции для решения систем ОДУ

Приведем формулы для решения системы (6.11) с помощью вышеописанных методов. Конечно-разностное представление будет иметь вид:

$$y_{i+1} = y_i + h z_i,$$

$$z_{i+1} = z_i + h(-2z_i - y_i + x_i),$$

$$y_0 = 1, \quad z_0 = 0,$$

для модифицированного метода Эйлера:

$$\tilde{y}_{i+1} = y_i + h z_i$$

$$\tilde{z}_{i+1} = z_i + h(-2z_i - y_i + x_i)$$

$$y_{i+1} = y_i + \frac{h}{2}[z_i + \tilde{z}_{i+1}]$$

$$z_{i+1} = z_i + \frac{h}{2}[(-2z_i - y_i + x_i) + (-2\tilde{z}_{i+1} - \tilde{y}_{i+1} + x_{i+1})],$$

$$x_{i+1} = x_i + h,$$

и четырехэтапного метода Рунге – Кутты:

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$z_{i+1} = z_i + \frac{h}{6}(l_1 + 2l_2 + 2l_3 + l_4)$$

$$k_1 = z_i, l_1 = -2z_i - y_i + x_i,$$

$$k_2 = z_i + \frac{hl_1}{2}, l_2 = -2\left(z_i + \frac{hl_1}{2}\right) - \left(y_i + \frac{hk_1}{2}\right) + x_i + \frac{h}{2},$$

$$k_3 = z_i + \frac{hl_2}{2}, l_3 = -2\left(z_i + \frac{hl_2}{2}\right) - \left(y_i + \frac{hk_2}{2}\right) + x_i + \frac{h}{2},$$

$$k_4 = z_i + hl_3, l_4 = -2(z_i + hl_3) - (y_i + hk_3) + x_i + h.$$

Программа для решения примера 6.3 методом Эйлера приведена на рис. 6.9.

```

g(x,y,z) := -2·z - y + x
Euler_sys(a,b,N,u0) :=
    |
    |  ORIGIN ← 1
    |  h ← (b - a) / (N - 1)
    |  u1,1 ← u01
    |  u1,2 ← u02
    |  for i ∈ 1..N - 1
    |  |
    |  |  xi ← a + (i - 1)·h
    |  |  u(i+1),1 ← ui,1 + h·ui,2
    |  |  u(i+1),2 ← ui,2 + h·g(xi,ui,1,ui,2)
    |  |
    |  |  xN ← b
    |  |  augmen(x,u)
    |
    |
u0 := (1)
      (0)
A1 := Euler_sys(0,1,N,u0)      i := 1..N

```

Рис. 6.9. Листинг решения системы (6.11) с помощью пользовательской ПФ, реализующей метод Эйлера

Здесь используется ПФ Euler_syst, входными данными в которую служат границы отрезка, число разбиений отрезка и вектор начальных данных. Выходным результатом является объединенная матрица, состоящая из вектора значений аргумента и матрицы решений. Процедура расчета решения аналогична ПФ Euler, только для системы число расчетов увеличивается пропорционально числу ОДУ в системе.

6.1.8. Жесткие ОДУ

До сих пор мы имели дело с ОДУ, которые надежно решались явными численными методами Рунге-Кутты. Однако имеется класс так называемых жестких (*stiff*) систем ОДУ, для которых явные методы практически не применимы, поскольку их решение требует исключительно малого значения шага численного метода. Рассмотрим пример такой жесткой задачи.

Пример 6.4. Решить задачу Коши

$$y' = -100y + 100, y(0) = 2$$

Точным решением задачи является функция $y = 1 + e^{-100x}$, имеющая очень большой градиент вблизи точки $x=0$. Действительно, $y=2$ при $x=0$ (в силу начальных данных), однако уже при малых положительных значениях x решение близко к своему асимптотическому значению $y = 1$. Получим численное решение этой задачи явным методом Эйлера (6.4) с шагом $h = 0,02$.

$$y_{i+1} = (1 - 100h)y_i + 100h = 2 - y_i, i = 0, 1, \dots, n-1, y_0 = 2.$$

Решение будет представлять собой последовательность

$$y(0) = 2, y(0.02) = 0, y(0.04) = 2, y(0.06) = 0, \dots$$

что не соответствует точному. При $h = 0.01$ первая же вычисленная точка $y_1 = 1$ попадает на асимптоту решения, и последующие вычисления не изменяют значения приближенного решения. Существенно более мелкий шаг, например $h = 0.001$, позволит получить вполне удовлетворительное соответствие между приближенным и точным решением. Однако вычисления с таким мелким шагом потребуют больших вычислительных затрат.

$$y(0) = y_0 = 2, y(0.001) \approx y_1 = 1.9, y(0.002) \approx y_2 = 1.81, \dots$$

Воспользуемся неявным методом Эйлера:

$$y_{i+1} = \frac{100h + y_i}{1 + 100h}, i = 0, 1, \dots, n-1, y_0 = 2$$

с шагом $h = 0.1$. Получим последовательность

$$y(0) = 2, y(0.1) = 1.091, y(0.2) = 1.008, y(0.3) = 1.0007, \dots$$

Даже при очень крупном шаге $h = 0.99$ приближенное решение, полученное неявным методом Эйлера, получается качественно правильным. $y_0 = 2, y_1 = 1.01, y_2 = 1.0001, \dots$

Данный пример показывает, что получить приближенное решение данной задачи гораздо рациональнее с помощью неявного метода Эйлера.

В приведенном выше примере коэффициенты уравнения различаются на порядки, причем коэффициент при старшей производной меньше остальных. Рассмотрим уравнение

$$\frac{dy}{dx} = \alpha y, \alpha < 0, |\alpha| \gg 1, x \geq 0, y(0) = y_0, \quad (6.12)$$

с точным решением $y = y_0 e^{\alpha x}$. Поскольку при $\alpha < 0$ точное решение является убывающим, для численного решения должна выполняться цепочка неравенств

$$\|y_{i+1}\| \leq \|y_i\| \leq \|y_{i-1}\| \leq \dots \leq \|y_0\|,$$

известных из теории разностных схем как принцип максимума. Методы, решения которых удовлетворяют этим условиям, называются A -устойчивыми методами.

Запишем для уравнения (6.12) явный метод Эйлера и двухэтапный метод Рунге-Кутты.

$$y_{i+1} = y_i + h\alpha y_i = (1 + \alpha h)y_i = \lambda_1 y_i$$

$$y_{i+1} = y_i + h\alpha(y_i + \alpha y_i h/2) = (1 + \alpha h + \alpha^2 h^2/2)y_i = \lambda_2 y_i.$$

Используя эти формулы, можно выразить последовательно каждое y_i через предыдущее, тогда

$$y_{i+1} = \lambda_k^{i+1} y_0, i = 0, 1, 2, \dots, k = 1, 2$$

Для выполнения принципа максимума $\|y_{i+1}\| \leq \lambda_k^{i+1} \|y_0\|$ необходимо и достаточно, чтобы выполнялось условие $0 \leq \lambda_k \leq 1$. Отсюда

сразу следуют ограничения на шаги интегрирования для явных методов. Например, для явного метода Эйлера $h \leq 1/|\alpha|$, для двухэтапного метода Рунге-Кутты $h \leq 2/|\alpha|$.

Рассмотрим теперь простейший неявный метод Эйлера для решения уравнения (6.12):

$$y_{i+1} = y_i + h\alpha y_{i+1} = y_i / (1 - \alpha h) = \lambda y_i.$$

Можно видеть, что условие $0 \leq \lambda \leq 1$ выполняется для любых α , следовательно, имеет место принцип максимума, т.е. неявный метод Эйлера не имеет ограничения по α на шаг интегрирования и является А-устойчивым. Неявный метод Эйлера может быть обобщен на систему жестких ОДУ.

6.1.9. Решение жестких ОДУ в пакете MathCAD

Решение жестких систем ОДУ в пакете MathCAD можно осуществить с помощью встроенной функции `Stiffb` (y_0, t_0, t_1, M, F, J)

где y_0 — вектор начальных значений в точке t_0 ; t_0, t_1 — начальная и конечная точка расчета; M — число шагов численного метода; F — векторная функция $F(t, y)$ размера $1 \times N$ (N — размерность системы), задающая правую часть системы ОДУ; J — матричная функция $J(t, y)$ размера $N \times (N+1)$, составленная из вектора производных функции $F(t, y)$ по t (правый столбец) и ее якобиана (N левых столбцов). Решение выдается в виде матрицы.

Далее показано решение жесткой системы ОДУ в MathCAD. Пусть задана система ОДУ первого порядка с начальными условиями

$$\begin{aligned} \frac{dy_0}{dx} &= -0.1y_0 + 10^2 y_1 y_2 \\ \frac{dy_1}{dx} &= 0.1y_0 - 10^2 y_1 y_2 - 10^3 y_1, \quad \bar{y}^0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{ при } x_0=0 \\ \frac{dy_2}{dx} &= 10^3 y_1 \end{aligned} \quad (6.13)$$

Видно, что коэффициенты при слагаемых в правой части данной системы имеют сильно различающиеся порядки. Жесткость системы характеризуется матрицей Якоби (якобианом) данной системы, т.е. матрицей, составленной из частных производных правой части по y_0 , y_1 и y_2 . Чем ближе определитель матрицы Якоби к нулю, тем жестче система. Используя функции MathCAD, найдем якобиан системы ОДУ с начальными данными (6.13) при помощи средств пакета:

$$F(x, y) := \begin{pmatrix} -0.1y_0 + 100y_1 \cdot y_2 \\ 0.1y_0 - 100y_1 \cdot y_2 - 1000y_1 \\ 1000y_1 \end{pmatrix} \quad y_0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

$$J(x, y) := \begin{bmatrix} \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} t, \\ y_1 \\ y_2 \end{matrix} \right] \right]_0 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ y_1 \\ y_2 \end{matrix} \right] \right]_0 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ t \\ y_2 \end{matrix} \right] \right]_0 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ y_1 \\ t \end{matrix} \right] \right]_0 \\ \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} t, \\ y_1 \\ y_2 \end{matrix} \right] \right]_1 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ y_1 \\ y_2 \end{matrix} \right] \right]_1 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ t \\ y_2 \end{matrix} \right] \right]_1 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ y_1 \\ t \end{matrix} \right] \right]_1 \\ \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} t, \\ y_1 \\ y_2 \end{matrix} \right] \right]_2 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ y_1 \\ y_2 \end{matrix} \right] \right]_2 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ t \\ y_2 \end{matrix} \right] \right]_2 & \frac{\partial}{\partial \alpha} \left[F \left[\begin{matrix} x, \\ y_1 \\ t \end{matrix} \right] \right]_2 \end{bmatrix}.$$

Получим Якобиан вида

$$J(x, y) \rightarrow \begin{pmatrix} 0 & -0.1 & 100 \cdot y_2 & 100 \cdot y_1 \\ 0 & 0.1 & -100 \cdot y_2 - 1000 & -100 \cdot y_1 \\ 0 & 0 & 1000 & 0 \end{pmatrix}.$$

Первый столбец матрицы J – это вектор производных функции $F(x, \bar{y})$ по x , а три правых столбца – якобиан системы (6.13).

Теперь воспользуемся функцией `Stiffb` для получения решения:

`D:=Stiffb(y0, 0, 50, 200, F, J) .`

Графики функций $y_0(x)$, $y_1(x)$ и $y_2(x)$, составляющих вектор полученного решения, представлен на рис. 6.11.

Для решения этого же примера с помощью методов Рунге-Кутты потребовался бы шаг на два порядка меньше, чем при использовании функции `Stiffb`.

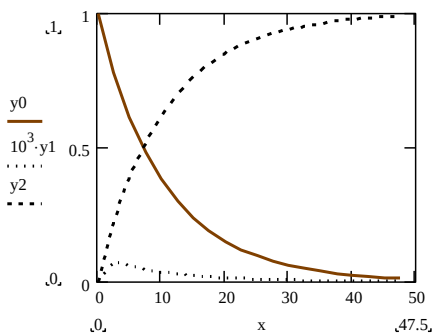


Рис 6.11 Приближенное решение жесткой системы ОДУ

6.2. Краевая задача для ОДУ второго порядка

6.2.1. Постановка краевой задачи

Дано дифференциального уравнения второго порядка

$$u'' + p(t)u' + g(t)u = f(t), \quad t \in [a, b]. \quad (6.14)$$

Здесь $p(t)$, $g(t)$, $f(t)$ – заданные функции коэффициентов.

Уравнение можно свести к системе двух ОДУ первого порядка:

$$\begin{cases} u' = f(t, u, v) \equiv v \\ v' = g(t, u, v) \equiv -p(t)v - g(t)u - f(t) \end{cases} \quad (6.15)$$

Для определения единственного решения необходимо задать два дополнительных условия на искомую функцию $u(t)$. Если оба условия заданы в одной точке $t=t_0$, то мы имеем задачу Коши, которая может быть решена методами, описанными в предыдущем разделе. Допустим теперь, что два дополнительных условия поставлены в разных точках: $x=a$ и $x=b$:

$$\begin{cases} k_1 u(a) + k_2 u'(a) = A \\ m_1 u(b) + m_2 u'(b) = B \end{cases} \quad (6.16)$$

где A, B, k_1, k_2, l_1, l_2 – заданные константы. Задача (6.14), (6.16) называется **краевой**, для приближенного решения которой используют методы:

- конечных разностей;
- сведение краевой задачи к задаче Коши (стрельбы, дифференциальной прогонки, редукции);
- балансов (конечных объемов);
- коллокации;
- проекционные (Галеркина);
- вариационные (наименьших квадратов, Ритца);
- проекционно-сеточные (конечных элементов).

Ниже рассмотрим некоторые из перечисленных методов.

6.2.2. Конечно-разностный метод

Введем на отрезке $[a, b]$ разностную сетку $(t_0, t_1, t_2, \dots, t_M)$, $t_i = a + \tau i$, $i = 0, 1, \dots, M$, $\tau = (b - a)/M$, M – число точек разностной сетки (параметр задачи). Вместо точного решения $u(t)$ будем отыскивать приближенное решение в узлах разностной сетки: $y_i = y(t_i)$. Используя формулы приближенного дифференцирования: $u''(t_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{\tau^2}$, $u'(t_i) \approx \frac{y_{i+1} - y_{i-1}}{2\tau}$, заменим исход-

ное уравнение и краевые условия разностной схемой:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{\tau^2} + p(t_i) \frac{y_{i+1} - y_{i-1}}{2\tau} + g(t_i) y_i = f(t_i), \quad i = 1, \dots, M,$$

$$k_1 y_0 + k_2 \frac{y_1 - y_0}{\tau} = A, \quad B$$

$$m_1 y_M + m_2 \frac{y_M - y_{M-1}}{\tau} = B.$$

итоге получили следующую систему $M+1$ линейных алгебраических уравнений на вектор неизвестных $(y_0, y_1, y_2, \dots, y_M)$:

$$-C_0 y_0 + B_0 y_1 = F_0$$

$$A_i y_{i-1} - C_i y_i + B_i y_{i+1} = F_i, i=1, 2, \dots, M-1$$

$$A_M y_{M-1} - C_M y_M = F_M$$

где $C_i = 2 - g(t_i) \tau^2$, $A_i = 1 - p(t_i) \tau/2$, $B_i = 1 + p(t_i) \tau/2$, $F_i = \tau^2 f(t_i)$,
 $C_0 = k_2 - \tau k_1$, $B_0 = k_2$, $F_0 = A \tau$, $C_M = m_2 + \tau m_1$, $A_M = m_2$, $F_M = -B \tau$.

Выпишем систему в матричном виде:

$$\begin{pmatrix} -C_0 & B_0 & 0 & 0 & \dots & 0 \\ A_1 & -C_1 & B_1 & 0 & \dots & 0 \\ 0 & A_2 & -C_2 & B_2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & A_{M-1} & -C_{M-1} & B_{M-1} \\ 0 & 0 & \dots & 0 & A_M & -C_M \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \dots \\ y_{M-1} \\ y_M \end{pmatrix} = \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ \dots \\ F_{M-1} \\ F_M \end{pmatrix}.$$

Поскольку матрица СЛАУ имеет трехдиагональный вид, то система решается методом прогонки, описанным в главе 2.

Метод прогонки является точным методом решения СЛАУ, следовательно, погрешность в приближенное решение вносится только на этапе замены исходных уравнений и краевых условий конечно-разностными соотношениями. Оценку погрешности метода можно провести, если вспомнить, что используемые формулы приближенного дифференцирования (центральная разность для первой производной и симметричная аппроксимация для второй производной) имеют второй порядок точности. В то же время при замене краевых условий (6.15) на разностные соотношения в приближенное решение вносится погрешность порядка τ (именно такую погрешность имеют формулы разность «вперед» и разность «назад»). Следовательно, суммарная погрешность аппроксимации уравнения и краевых условий будет пропорциональна τ . Однако в тех случаях, когда в краевые условия не входит производная, т.е. $k_2=0$, $l_2=0$, то краевые условия для приближенного решения выполняются точно, и тогда метод имеет погрешность порядка τ^2 .

6.2.3. Метод стрельбы

Этот метод основан на сведении краевой задачи к задаче Коши для системы (6.14). Пусть краевые условия имеют вид:

$$u(a)=u_0, u(b)=u_1.$$

Для того чтобы свести исходную краевую задачу к задаче Коши, необходимо в точке $t=a$ задать дополнительное краевое

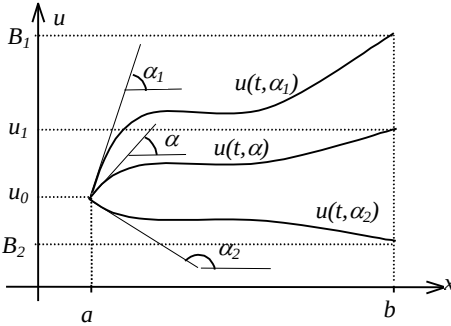


Рис. 6.12. Иллюстрация метода стрельбы

условие $u(a)=v_0$. Величина v_0 имеет геометрический смысл тангенса α — угла наклона касательной к решению в начальной точке. Графическая иллюстрация метода стрельбы приведена на рис. 6.11.

Задачу Коши для системы (6.14) можно решить, например, методом Рунге-Кутты.

Поскольку решение задачи Коши зависит от выбора α , можно записать: $u=u(t, \alpha)$. Требуется подобрать значение α , обеспечивающее «попадание», т.е. выполнение условия $u(t, \alpha)=u_1$. Понятно, что при произвольном выборе α полученное решение может не удовлетворять краевому условию на правом конце отрезка $t=b$. Может быть получено, что $u(t, \alpha_1)|_{t=b}=B_1 > u_1$ («перелет») или $u(t, \alpha_1)|_{t=b}=B_2 < u_1$ («недолет») (рис. 6.11). Задачу подбора нужного угла α можно рассматривать как решение нелинейного алгебраического уравнения $u(t, \alpha)|_{t=b}=u_1$, особенностью которого является то, что $F(x)$, определяющая нелинейное уравнение, не задана явно, а определен только способ нахождения $u(t, \alpha)|_{t=b}$.

6.2.4. Решение краевой задачи конечно-разностным методом в пакете MathCAD

Рассмотрим реализацию метода конечных разностей в среде MathCAD на примере ОДУ 2-го порядка вида (6.14):

$$u'' + tg(t)u' + u \cos^2 t = 0, \quad x \in [0, 1]$$

с краевыми условиями $u(0) = 1, u(1) = 10$.

Процедура-функция, реализующая вышеописанный метод, представлена на рис. 6.13. Входными параметрами ПФ kr служат число разбиений отрезка M , начало и конец отрезка a и b , и вектор начальных данных $u0$.

Алгоритм ПФ включает вычисление шага сетки (τ), в первом цикле рассчитываются аргумент и коэффициенты трехдиагональной матрицы A , B , C и F , описанные в параграфе 6.2.1. Затем вычисляются прогоночные коэффициенты α и β и далее реализуется обратный ход метода прогонки (см. главу 2), где вычисляются значения искомой функции в заданных точках сетки. Выходными данными является массив y решения ОДУ.

Для данной задачи имеется точное решение вида

$$ut(t) := \cos(\sin(t)) + C1 \cdot \sin(\sin(t)), \quad C1 := \frac{10 - \cos(\sin(1))}{\sin(\sin(1))}.$$

```
kr(M, a, b) :=
  tau ← (b - a) / M
  for i ∈ 0..M
    t_i ← a + i · tau
    aa_i ← 1 - p(t_i) · (tau / 2)
    bb_i ← 1 + p(t_i) · (tau / 2)
    cc_i ← 2 - g(t_i) · tau^2
    ff_i ← f(t_i) · tau^2
  al_1 ← 0
  bet_1 ← 1

  for i ∈ 1..M - 1
    al_{i+1} ← (bb_i) / (cc_i - al_i · aa_i)
    bet_{i+1} ← (aa_i · bet_i - ff_i) / (cc_i - al_i · aa_i)
  y_M ← 10
  i ← M - 1
  while i > -1
    y_i ← al_{i+1} · y_{i+1} + bet_{i+1}
    i ← i - 1
  y
```

Рис. 6.13 Листинг процедуры-функции, реализующей конечно-разностный метод решения краевой задачи

Используя точное решение, можно вычислить погрешность приближенного решения

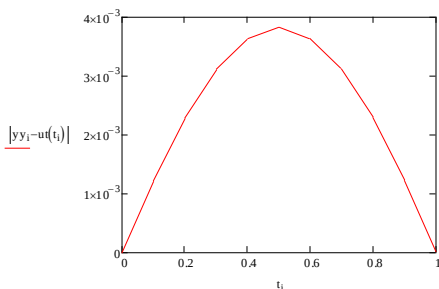


Рис. 6.14. Погрешность приближенного решения краевой задачи методом конечных разностей

в каждой точке заданного интервала и отобразить ее графически. Для $M=10$ график показан на рис. 6.14. Видно, что на концах отрезка, где известны краевые условия, погрешность равна нулю, и достигает максимума к середине интервала, но значение его невелико.

6.2.5. Метод коллокаций

Запишем краевую задачу для ОДУ второго порядка в операторном виде:

$$Lu = f(t), \quad l_0 u(a) = A, \quad l_1 u(b) = B, \quad (6.17)$$

где $L = \frac{d^2}{dt^2} + p(t)\frac{d}{dt} + q(t)$, $l_0 = k_1 + k_2 \frac{d}{dt}$, $l_1 = m_1 + m_2 \frac{d}{dt}$.

Зададим на $[a, b]$ некоторую систему базисных функций $\varphi_0(t)$, $\varphi_1(t)$, ..., $\varphi_n(t)$, таких, что $\varphi_0(t)$ удовлетворяет краевым условиям $l_0 \varphi_0(a) = A$, $l_1 \varphi_0(b) = B$, а остальные $\varphi_k(t)$ удовлетворяют однородным краевым условиям $l_0 \varphi_k(a) = 0$, $l_1 \varphi_k(b) = 0$, $k = 1, \dots, n$. Представим приближенное решение задачи (6.17) в виде линейной комбинации базисных функций:

$$y_n(t) = \varphi_0(t) + a_1 \varphi_1(t) + a_2 \varphi_2(t) + \dots + a_n \varphi_n(t) \quad (6.18)$$

с неизвестными пока коэффициентами a_1 , a_2 , ..., a_n . При этом $y_n(t)$ при любых значениях a_1 , a_2 , ..., a_n удовлетворяет краевым условиям. Подействуем на (6.18) оператором L . Функция

$$\psi(t, a_1, a_2, \dots, a_n) = Ly_n(t) - f(t) = L\varphi_0(t) - f(t) + \sum_{k=1}^n a_k L\varphi_k$$

Доказано, что если функция u является решением уравнения (6.19), то на ней функционал (6.20) достигает экстремума, и наоборот, функции, поставляющие экстремум функционала (6.20) являются решениями (6.19). Существуют также другие функционалы, связанные с уравнением (6.19), например,

$$I(u) = (Lu - f, Lu - f),$$

представляющий собой квадрат нормы невязки уравнения.

Таким образом, вместо того, чтобы искать решение уравнения, можно отыскивать функции, на которых тот или другой функционал достигает экстремума.

Пример 6.5.

Рассмотрим задачу о шарнирно опертой по концам балке, находящейся под действием равномерно распределенной нагрузки $Q(x)$. Дифференциальное уравнение, описывающее этот процесс, имеет вид

$$\frac{d^2 u(x)}{dx^2} = Q(x). \quad (6.21)$$

Будем искать приближенное решение v в виде комбинации n линейно-независимых базисных функций $\varphi_i(x)$:

$$v(x) = \sum_{i=1}^n c_i \varphi_i(x). \quad (6.22)$$

где c_i - неизвестные коэффициенты, которые должны быть определены из некоторых условий. Введем функционал квадрата нормы невязки и потребуем, чтобы он был минимален

$$G(v) = \int_a^b \left(\frac{d^2 v}{dx^2} - Q \right)^2 dx \rightarrow \min. \quad (6.23)$$

Подставим (6.22) в (6.23):

$$G(v) = \int_a^b \left(\frac{d^2 \sum_{i=1}^n c_i \varphi_i(x)}{dx^2} - Q \right)^2 dx.$$

Поскольку искомые коэффициенты не зависят от x , можно вынести их из-под знака дифференцирования, и переписать выражение в виде

$$G(v) = \int_a^b \left(\sum_{i=1}^n c_i \frac{d^2 \varphi_i(x)}{dx^2} - Q \right) \left(\sum_{i=1}^n c_i \frac{d^2 \varphi_i(x)}{dx^2} - Q \right) dx =$$

$$= \int_a^b \sum_{i=1}^n \sum_{j=1}^n c_i c_j \frac{d^2 \varphi_i(x)}{dx^2} \frac{d^2 \varphi_j(x)}{dx^2} dx - 2 \int_a^b \sum_{i=1}^n c_i \frac{d^2 \varphi_i(x)}{dx^2} Q dx + \int_a^b Q^2 dx.$$

Введем обозначения:

$$A_{ij} = \int_a^b \frac{d^2 \varphi_i(x)}{dx^2} \frac{d^2 \varphi_j(x)}{dx^2} dx, \quad \alpha_i = \int_a^b \frac{d^2 \varphi_i(x)}{dx^2} Q dx, \quad \beta = \int_a^b Q^2 dx.$$

Тогда

$$G(v) = \sum_{i=1}^n \sum_{j=1}^n c_i c_j A_{ij} - 2 \sum_{i=1}^n c_i \alpha_i + \beta.$$

По условию минимума функционала G его производные по коэффициентам c_i должны равняться нулю:

$$\frac{\partial G}{\partial c_i} = \sum_{j=1}^n c_j A_{ij} - \sum_{i=1}^n \alpha_i = 0.$$

Таким образом, мы получили СЛАУ из n уравнений

$$\mathbf{A} \vec{c} = \vec{\alpha}. \quad (6.24)$$

Поскольку матрица $\mathbf{A}$ симметрична и положительно определена, как следует из способа ее задания, система (6.24) имеет решение. Решим СЛАУ каким-либо из описанных выше методов, и, подставляя коэффициенты c_i в (6.22), получим приближенное решение уравнения (6.21).

Допустим, что $Q(x)=1$, длина балки 1. В этом случае одним из возможных вариантов решения может стать предложенное на рис. 6.15. Число разбиений M в данном случае для простоты равно 2. Вид базисных функций определен в $gg(x,i)$. Процедура-функция `mas` определяет массив $\mathbf{A}$, процедура-функция `vec` вычисляет вектор $\vec{\alpha}$ из формулы (6.24). Полученная СЛАУ в данном случае состоит всего из двух уравнений, что позволяет найти ее решение точным методом обратной матрицы.

```

ORIGIN:= 1
*****
M:= 2    LL:= 1    q(x) := 1

gg(x,i) := xi(LL - x)
mas(M,LL) :=
|
|   for k ∈ 1..M
|       for i ∈ 1..M
|           KKk,i ← ∫0LL gg(x,k) ·  $\frac{d^2}{dx^2}$  gg(x,i) dx
|
|   KK
|
KK := mas(M,LL)
KK =  $\begin{pmatrix} -0.333 & -0.167 \\ -0.167 & -0.133 \end{pmatrix}$ 
vec(M,LL) :=
|
|   for i ∈ 1..M
|       Fi ← - ∫0LL gg(x,i) · q(x) dx
|
|   F
|
FF := vec(M,LL)
FF =  $\begin{pmatrix} -0.167 \\ -0.083 \end{pmatrix}$ 
CC := KK-1 · FF
fi(x) := ∑i=1M (CCi · gg(x,i)) → - $\frac{x(x-1)}{2}$ 
CC =  $\begin{pmatrix} 0.5 \\ 1.155 \times 10^{-14} \end{pmatrix}$ 
fi( $\frac{LL}{2}$ ) = 0.125

```

Рис. 6.15. Листинг программы расчета уравнения (6.21) вариационным методом

В итоге мы получаем функцию $f_i(x)$, которая и есть решение исходного уравнения по вариационному методу. Ее аналитический вид представлен в предпоследней строке листинга. Значение прогиба балки в центральной точке вычислено в последней строке программы. Точное решение, вид которого для уравнения (6.21) при заданном Q и условии, что изгибающий момент на концах балки равен нулю, предлагается найти самостоятельно, дает значение 0,125.

6.2.7. Проекционные методы

Пусть задано уравнение вида $Lu = f$, где L — линейный оператор. Скалярное произведение функций $u(t)$, $v(t)$, заданных на отрезке $[0, T]$, можно определить как $(u, v) = \int_0^T u(t)v(t)dt = 0$.

Решение уравнения можно представить в виде $u = \sum_{k=1}^{\infty} c_k \varphi_k$.

Подставив приближенное решение в исходное уравнение, получим невязку $\psi = f - Lv$. Запишем тождество:

$$Lv + \psi = f.$$

По теореме о разложении следует, что невязка должна быть ортогональна всем базисным функциям φ_j , $j = 1, 2, \dots, n$, т.е. их скалярное произведение должно быть равно нулю:

$$(\psi, \varphi_j) = (Lv - f, \varphi_j) = (L \sum_{k=1}^n c_k \varphi_k - f, \varphi_j) = 0, j = 1, 2, \dots, n. \quad (6.25)$$

В силу линейности оператора L из (6.25) получим систему n линейных алгебраических уравнений для определения коэффициентов c_k : $\sum_{k=1}^n c_k \cdot (L\varphi_k, \varphi_j) = (f, \varphi_j)$, $j = 1, 2, \dots, n$.

Так же, как и в ранее описанном вариационном методе Ритца, решение сводится к нахождению решения СЛАУ вида (6.24), где коэффициенты матрицы $\mathbf{A}$ и вектора правых частей определяются как

$$a_{ij} = (L\varphi_j, \varphi_i), \quad \alpha_i = (f, \varphi_i), \quad (6.26)$$

что также совпадает с приведенными выше формулами вариационного метода Ритца.

В качестве базисных функций можно выбрать, например, степенной $\varphi_i = x^{i-1}$, или тригонометрический базис

$$\varphi_i = \sin\left(\frac{x-a}{b-a}\left(i\pi - \frac{\pi}{2}\right)\right), \quad i=1, \dots, n.$$

Недостатком этих базисов является то, что матрица системы (6.24) является заполненной, и расчетные затраты растут пропорционально n^3 , что ограничивает применение методов для больших n .

6.2.8. Метод конечных элементов

Как отмечено выше, проекционные и вариационные методы решения краевой задачи (6.17) приводят к одинаковой системе линейных алгебраических уравнений с коэффициентами и правыми частями, заданными (6.26). Заполненность матрицы зависит от вида базисных функций. В методе конечных элементов в качестве базисных выбираются финитные функции, отличные от нуля на некотором ограниченном интервале отрезка $[a, b]$.

Основное отличие метода конечных элементов от вышеприведенных проекционных и вариационных состоит в том, что вместо нахождения глобальной функции решения на всей области последняя разбивается на отдельные элементы, в каждом из которых выбирается своя базисная функция (или функция формы). Вид этих функции может быть довольно простым (включая постоянные значения и линейные функции). Затем на каждом элементе используется один из алгоритмов минимизации функционала (метод Галеркина, Ритца и т.п.) и в итоге решение собирается (ассамблируется) из отдельных решений. Т.е. метод конечных элементов является в некотором роде симбиозом вариационных методов и методов конечных разностей, где также производится разбиение области решения.

Метод конечных элементов широко применяется в современной инженерной практике, в частности, при решении задач строительства и используются во многих современных расчетных пакетах. Отметим, что метод конечных элементов впервые нашел свое приложение в задачах строительства, расчета стержневых и балочных конструкций, поэтому имеется физическая интерпретация данного метода, основанного на методе перемещений. Отсюда же идет терминология, используемая в методе, например, матрица жесткости системы. Здесь мы приводим математическую формулировку метода, полностью согласующуюся с физическим подходом.

Приведем пример решения уравнения (6.17) с нулевыми краевыми условиями $u(a)=u(b)=0$ методом конечных элементов. Для этого разобьем отрезок $[a, b]$ на $n-1$ частей и запишем базисные финитные функции в виде

$$\varphi_i = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in [x_{i-1}, x_i], \\ -\frac{x - x_{i+1}}{x_{i+1} - x_i}, & x \in [x_i, x_{i+1}], \\ 0, & x \notin [x_{i-1}, x_{i+1}]. \end{cases}$$

Предположим, что размер элементов постоянен и равен h :

$$\varphi_i = \begin{cases} \frac{x - x_{i-1}}{h}, & x \in [x_{i-1}, x_i], \\ -\frac{x - x_{i+1}}{h}, & x \in [x_i, x_{i+1}], \\ 0, & x \notin [x_{i-1}, x_{i+1}]. \end{cases} \quad (6.27)$$

В этом случае производные базисных функций равны

$$\varphi'_i = \begin{cases} \frac{1}{h}, & x \in [x_{i-1}, x_i], \\ -\frac{1}{h}, & x \in [x_i, x_{i+1}], \\ 0, & x \notin [x_{i-1}, x_{i+1}]. \end{cases} \quad (6.28)$$

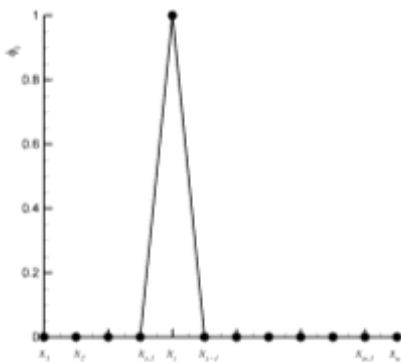


Рис. 6.16. Базисная функция в МКЭ

График одной базисной функции показан на рис. 6.16. Т.е. функции подбираются таким образом, чтобы в узле рассматриваемого элемента ее значение равнялась единице, а на соседних элементах — нулю. Будем искать приближенное решение в виде (6.22). Тогда значения неизвестных коэффициентов принимают смысл значений функции в узлах элементов. Найдем правые части системы согласно (6.26)

$$\alpha_i = \int_a^b f(x) \varphi_i(x) dx = \sum_{k=1}^{n-1} \int_{x_k}^{x_{k+1}} f(x) \varphi_i(x) dx.$$

Подставив сюда выражение (6.26), получим

$$\begin{aligned} \alpha_i &= \int_{x_{i-1}}^{x_i} f(x) \frac{(x - x_{i-1})}{h} dx - \int_{x_i}^{x_{i+1}} f(x) \frac{(x - x_{i+1})}{h} dx = \\ &= \frac{1}{h} \left\{ \int_{x_{i-1}}^{x_i} f(x) (x - x_{i-1}) dx - \int_{x_i}^{x_{i+1}} f(x) (x - x_{i+1}) dx \right\}. \end{aligned} \quad (6.29)$$

Зная функцию $f(x)$, выражение (6.29) можно проинтегрировать и получить конкретный вид правых частей СЛАУ. В тех случаях, когда аналитическое решение невозможно, необходимо воспользоваться методами численного интегрирования.

Теперь получим коэффициенты матрицы СЛАУ

$$a_{ij} = (L \varphi_j, \varphi_i) = \left(\frac{d^2 \varphi_j}{dx^2}, \varphi_i \right) = \int_a^b \frac{d^2 \varphi_j}{dx^2} \varphi_i dx,$$

и, применяя интегрирование по частям, имеем

$$\begin{aligned}
 a_{ij} &= \varphi_i(x) \varphi_j'(x) \Big|_a^b - \int_a^b \varphi_j'(x) \varphi_i'(x) dx = \\
 &= \varphi_i(b) \varphi_j'(b) - \varphi_i(a) \varphi_j'(a) - \int_a^b \varphi_j'(x) \varphi_i'(x) dx.
 \end{aligned}$$

Как видно из определения базисных функций, на концах отрезка $[a, b]$ все они обращаются в ноль. Поэтому два первых члена в этом выражении равны нулю и

$$a_{ij} = - \int_a^b \varphi_j'(x) \varphi_i'(x) dx = \sum_{k=1}^n \int_{x_k}^{x_{k+1}} -\varphi_j'(x) \varphi_i'(x) dx.$$

Согласно (6.28), производные базисных функций равны нулю везде, кроме соседних с исследуемой точкой отрезках (элементах). Т.е. $\varphi_j'(x) = 0$ при $i-1 \leq j \leq i+1$. Отсюда следует вывод, что полученная нами матрица **A** является трехдиагональной. Элементы главной диагонали ($i=j$) определяются как

$$\begin{aligned}
 a_{ii} &= \sum_{k=1}^n \int_{x_k}^{x_{k+1}} -(\varphi_i')^2 dx = \int_{x_{i-1}}^{x_i} -\frac{1}{h^2} dx + \int_{x_i}^{x_{i+1}} -\left(-\frac{1}{h}\right)^2 dx = \\
 &= -\frac{1}{h^2} \left(\int_{x_{i-1}}^{x_i} dx + \int_{x_i}^{x_{i+1}} dx \right) = -\frac{2}{h}
 \end{aligned}$$

На соседних с главной диагональю

$$\begin{aligned}
 a_{i,i+1} &= \sum_{k=1}^n \int_{x_k}^{x_{k+1}} -\varphi_i' \varphi_{i+1}' dx = \int_{x_i}^{x_{i+1}} -\frac{1}{h} \left(-\frac{1}{h}\right) dx = \frac{1}{h^2} \left(\int_{x_i}^{x_{i+1}} dx \right) = \frac{1}{h}, \\
 a_{i,i-1} &= \sum_{k=1}^n \int_{x_k}^{x_{k+1}} -\varphi_i' \varphi_{i-1}' dx = - \int_{x_{i-1}}^{x_i} -\frac{1}{h} \frac{1}{h} dx = \frac{1}{h}.
 \end{aligned}$$

Таким образом, матрицу **A** можно записать как

$$\mathbf{A} = \begin{pmatrix} -2/h & 1/h & 0 & 0 & \dots & 0 \\ 1/h & -2/h & 1/h & 0 & \dots & 0 \\ 0 & 1/h & -2/h & \dots & 0 & 0 \\ 0 & 0 & \dots & 1/h & -2/h & 1/h \\ 0 & 0 & \dots & 0 & 1/h & -2/h \end{pmatrix}.$$

Матрица коэффициентов этой системы называется также матрицей жесткости. Систему $\mathbf{A}\vec{c} = \vec{a}$ можно решить точным экономичным методом прогонки, описанным в главе 2. В других случаях возможно применение итерационных методов решения СЛАУ.

В случае если краевые условия имеют более сложный вид, необходимо ввести замену переменных с тем, чтобы привести их к виду $u(a)=u(b)=0$. Более подробно о методе, его вариациях, обобщении на многомерные задачи можно узнать из литературы [4, 6].

Согласно примеру решения МКЭ, показанному выше, можно создать алгоритм решения краевой задачи методом конечных элементов. Например, такой:

1. Разбить область на подобласти (в общем случае, это могут быть неравномерные трехмерные элементы);
2. Выбрать вид базисной функции;
3. Для каждого элемента построить функционал и применить процедуру минимизации. Таким образом, определяются локальные коэффициенты СЛАУ;
4. Объединить локальные матрицы каждого элемента в глобальную матрицу жесткости (ансамблирование) и создать вектор правых частей;
5. Учесть граничные условия;
6. Решить трехдиагональную СЛАУ численно.

Пример 6.6. Рассмотрим задачу из примера 6.5, но примем, что наша нагрузка линейно изменяется в пространстве. Предположим, что Q есть линейная функция u . Для простоты положим $Q=u$. В качестве граничных условий возьмем $u(0)=0$ и $u(1)=1$.

Тогда решение методом конечных элементов можно получить, следуя алгоритму, приведенному выше:

1. Разбиваем отрезок $[0,1]$ на M равных частей (формируем массив x)
2. В качестве базисных функций возьмем линейное распределение, так что в узле каждого элемента соответствующая ему функция равнялась 1, а в остальных местах – нулю. Таким образом, на каждом элементе будут присутствовать две функции, соответствующие левому и правому его узлу ($N1$ и $N2$, соответственно)
3. В цикле процедуры `ansmb`, входным параметром которой служит число узлов, определяется функционал для каждого элемента и строятся локальные матрицы.
4. В этой же ПФ происходит ансамблирование коэффициентов и создание матрицы жесткости K , которая является выходным параметром работы `ansmb`. Далее создается вектор правых частей f .
5. В процедуру-функцию `solv` пересылаются количество узлов, вектор правых частей, граничные условия
6. Здесь же решается трехдиагональная СЛАУ методом прогонки (см. главу 2) и результат записывается в массив.

Невязка полученного решения для $M=3$ приведена на рис. 6.17. Можно убедиться, что порядок ошибки мал по сравнению с решением.

```

M := 3    i := 1..M + 1    a := 0    b := 1
h := (b - a) / M = 0.333    x1 := 0 + (i - 1)·h
fi1 := C    fiM := 1
N1(i,y) := (xi+1 - y - x1) / (xi+1 - x1)    N2(i,y) := y / (xi+1 - x1)

ansmb(M) :=
  for i ∈ 1..M
    for j ∈ 1..M
      Ki,j ← 0
    for i ∈ 1..M - 1
      hj ← xi+1 - x1
      ke1,2 ←  $\left[ \int_0^{hj} \left( \frac{d}{dy} N1(i,y) \cdot \frac{d}{dy} N2(i,y) + N1(i,y) \cdot N2(i,y) \right) dy \right]$ 
      ke2,1 ← ke1,2
      ke1,1 ←  $\left[ \int_0^{hj} \left[ \left( \frac{d}{dy} N1(i,y) \right)^2 + (N1(i,y))^2 \right] dy \right]$ 
      ke2,2 ← ke1,1
      for m ∈ 1..2
        for l ∈ 1..2
          Ki-1+m, i-1+l ← Ki-1+m, i-1+l + kem,l
      K
KK := ansmb(M + 1)

```

Рис. 6.17. Листинг решения примера 6.6. методом конечных элементов (начало)

$$f_1 := \frac{1}{x_2 - x_1}$$

$$f_{M+1} := \frac{1}{(x_{M+1} - x_M)}$$

$$i := 2..M$$

$$f_i := 0$$

$$\text{solv}(M, KK, FF, fl, fr) :=$$

```

L ← M
for i ∈ 2..L - 1
    CC1 ← -KKi,i
    BB1 ← KKi,i+1
    AAi ← KKi,i-1
    α3 ←  $\frac{-KK_{2,3}}{KK_{2,2}}$ 
    β3 ←  $\frac{FF_2}{KK_{2,2}}$ 
for i ∈ 3..L - 1
    αi+1 ←  $\frac{BB_1}{CC_1 - \alpha_1 \cdot AA_i}$ 
    βi+1 ←  $\frac{AA_i \cdot \beta_i - FF_i}{CC_1 - \alpha_1 \cdot AA_i}$ 
fiL ← fr
for i ∈ L - 1..1
    fii ← αi+1 · fii+1 + βi+1
fi

```

$$a1 := \text{solv}(M + 1, KK, f, fi1, fiM)$$

Рис. 6.17. Листинг решения примера 6.6 методом конечных элементов (окончание)

Глава 7. Решение уравнений в частных производных

7.1. Основные понятия. Постановка задачи

Для описания реального физического процесса (прогиба балки под действием нагрузки, движения газа в некотором объеме, распространение электромагнитных волн и пр.) строятся физико-математические модели, на основе которых можно анализировать процессы качественно и количественно. Задачи описания движения сплошных сред (газа, жидкости, твердых тел), а также задачи теплопроводности, теории упругости, электрических и магнитных полей, и многие другие приводят к дифференциальным уравнениям. Независимыми переменными в физических задачах являются время t и пространственные координаты x, y, z . В качестве зависимых переменных в разных задачах используются компоненты скорости частиц среды, плотность, давление, температура, упругие напряжения, деформации и др. характеристики.

Допустим, что требуется найти решение на временном промежутке $[t_0, t_1]$ в некоторой области изменения независимых переменных $G(x, y, z)$. Математическая постановка задачи состоит из дифференциального уравнения (уравнений), а также дополнительных условий, позволяющих выделить единственное решение среди семейства всех решений данного уравнения. Дополнительные условия, заданные при $t=t_0$ называются начальными данными, а условия, заданные на границе области $G(x, y, z)$ – граничными или краевыми условиями. В качестве начальных и краевых условий, как правило, задают значения искомых функций и их производных. Задачи, у которых имеются только начальные условия, называются **задачей Коши**. Задачи с начальными данными и граничными условиями называют **смешанной краевой задачей** или нестационарной краевой задачей.

При исследовании установившихся состояний или стационарных (не зависящих от времени) процессов используются уравнения, не зависящие от времени. В этом случае решение ищется в области $G(x, y, z)$, на границе которой задаются граничные условия. Такие задачи называются **краевыми**.

Особым вопросом в теории дифференциальных уравнений является **корректность** постановки начальных и смешанных задач. Корректной называется такая постановка дополнительных (начальных и граничных) условий, при которой решение задачи в целом существует, единственно и непрерывно зависит от этих данных и коэффициентов уравнения. Требование непрерывной зависимости необходимо, чтобы небольшие изменения коэффициентов уравнения, начальных данных и краевых условий не приводили к сильным изменениям решения задачи. В механике и физике существуют задачи, решение которых неустойчиво. Изучению таких некорректных задач занимается специальный раздел математики. Здесь мы будем рассматривать только корректные постановки, при решении которых не возникает неустойчивости, связанной с исходными уравнениями.

7.1.1. Характеристики. Типы уравнений.

Многие физические задачи приводят к решению уравнений второго порядка, которые достаточно хорошо изучены в теоретическом плане и для которых разработаны стандартные методы приближенного решения. В случае одной пространственной координаты уравнение в частных производных второго порядка можно записать в виде

$$Au_{tt} + Bu_{tx} + Cu_{xx} + Du_t + Eu_x = F.$$

Здесь $u(t, x)$ – искомая функция, t, x – независимые переменные, A, B, C, D, E и F – коэффициенты уравнения, которые, вообще говоря, могут зависеть от t, x и u .

Если все коэффициенты являются константами, то это линейное уравнение с постоянными коэффициентами. Если коэффициент F – линейная функция от неизвестной u , а остальные коэффициенты от u не зависят, то такое уравнение называется линейным с переменными коэффициентами. Если все коэффициенты зависят от u , то такие уравнения называются квазилинейными. В случае если коэффициенты зависят не только от искомым функций, но и от ее производных, уравнение будет нелинейным.

Если коэффициенты A, B, C – нулевые, а $D \neq 0$ и $E \neq 0$, то уравнение имеет первый порядок и называется уравнением переноса (адвекции). Если хотя бы один из коэффициентов A, B, C отличен от нуля, уравнение имеет второй порядок и может быть классифицировано по типам аналогично кривым второго порядка. Классификация уравнений связана с наличием характеристик – особых направлений, вдоль которых исходное уравнение может быть записано в виде полного дифференциала, и, следовательно, может быть проинтегрировано. Уравнение характеристик для уравнения в частных производных второго порядка имеет вид

$$\frac{dx}{dt} = \frac{B \pm \sqrt{B^2 - 4AC}}{2A}.$$

Количество характеристик зависит от знака дискриминанта $B^2 - 4AC$. Если он положителен, то уравнение (7.1) имеет две вещественных характеристики и называется гиперболическим. В случае нулевого дискриминанта уравнение имеет одну вещественную характеристику и является параболическим. Эллиптические уравнения не имеют вещественных характеристик (дискриминант отрицателен).

Физические процессы, описываемые уравнениями перечисленных типов, корректные постановки начально-краевых задач и свойства решений существенно отличаются друг от друга.

7.1.2. Аппроксимация и устойчивость

Как правило, уравнения в частных производных не могут быть решены аналитически (точно), и для нахождения решений используются приближенные методы, которые можно разделить на три основные группы: методы конечных разностей, методы конечных объемов и методы конечных элементов.

В **конечно-разностных методах** приближенное решение ищется в узлах специально построенной разностной сетки, покрывающей область решения исходной дифференциальной задачи. Дифференциальная задача с помощью формул приближенного дифференцирования заменяется системой алгебраиче-

ских уравнений (разностной схемой), которая далее решается с помощью точных или приближенных методов решения СЛАУ. При этом необходимо использовать такие разностные схемы, которые обеспечивают сходимость получаемого решения разностной задачи к решению исходной дифференциальной при уменьшении шага сетки.

Аппроксимация (от англ. *approximation* — приближение) характеризует, насколько хорошо разностная задача приближает исходную дифференциальную, и зависит от точности формул разностного дифференцирования и, конечно же, размеров разностной сетки. Для простых задач аппроксимация может быть исследована аналитически с помощью оценки главного члена погрешности разностной схемы при подстановке в нее точного решения. В более сложных случаях аппроксимацию исследуют экспериментально, с помощью оценки поведения погрешности при измельчении сетки.

Устойчивость характеризуют способность решения не накапливать ошибку. Другими словами, устойчивость - это непрерывная зависимость решения от входных данных: коэффициентов, правых частей, начальных данных и краевых условий. При этом следует различать устойчивость решения исходной дифференциальной задачи и устойчивость приближенного метода. Аппроксимируем записанную в общем виде дифференциальную задачу с начальными данными и краевыми условиями:

$$Lu = \varphi, \quad u^0 = v_0, \quad lu = \mu,$$

с помощью конечно-разностной схемы:

$$L_h u_h = \varphi_h, \quad u_h^0 = v_0, \quad l_h u_h = \mu_h.$$

Рассмотрим также конечно-разностную задачу с «возмущенными» правыми частями, начальными данными и краевыми условиями:

$$L_h \tilde{u}_h = \tilde{\varphi}_h, \quad \tilde{u}_h^0 = \tilde{v}_0, \quad l_h \tilde{u}_h = \tilde{\mu}_h.$$

Говорят, что решение разностной задачи непрерывно зависит от входных данных задачи, если существуют такие константы $M_1 > 0$, $M_2 > 0$, $M_3 > 0$, не зависящие от t и h , что

$$\|u_h(t) - \tilde{u}_h(t)\| \leq \|M_1|\varphi_h - \tilde{\varphi}_h| + M_2|\mu_h - \tilde{\mu}_h| + M_3|v_0 - \tilde{v}_0|\| \quad (7.0)$$

Для линейных разностных схем доказано, что устойчивость по начальным данным эквивалентна устойчивости по правой части, т.е. для линейных операторов L_h , l_h достаточно показать, что

$$\|u_h(t) - \tilde{u}_h(t)\| \leq M\|v_0 - \tilde{v}_0\|.$$

Если независимых переменных несколько (например, x и t), то вводится понятие условной и безусловной устойчивости. Устойчивость называется *безусловной*, если соотношение (7.0) имеет место при произвольном соотношении между шагами разностной сетки τ и h . Схема будет *условно устойчивой*, если для выполнения (7.0) шаги по независимым переменным должны подчиняться дополнительным соотношениям.

В теории разностных схем доказана теорема Лакса о том, что если разностная задача аппроксимирует дифференциальную и устойчива, то при измельчении сетки решение разностной задачи сходится к решению дифференциальной.

Подробнее о свойствах аппроксимации и устойчивости будет рассказано ниже при рассмотрении примеров уравнений в частных производных.

7.2. Параболические уравнения

7.2.1. Постановка задачи

К параболическим уравнениям приводят задачи теплопроводности, диффузии и некоторые другие. Рассмотрим уравнения этого типа на примере одномерного (т.е. с одной пространственной переменной) линейного уравнения теплопроводности с постоянными коэффициентами, описывающими распространение тепла в тонких однородных стержнях:

$$\frac{\partial u}{\partial t} = A \frac{\partial^2 u}{\partial x^2} + F(x, t). \quad (7.1)$$

Здесь $A > 0$ – константа (коэффициент теплопроводности), $u(x, t)$ – искомое решение, $F(x, t)$ – правая часть. Будем искать

решение в области $0 \leq x \leq L$, $0 \leq t \leq T$. Корректная постановка задачи кроме уравнения (7.1) включает в себя начальные данные:

$$u(x, 0) = u_0(x) \quad (7.2)$$

и краевые условия. Наиболее хорошо изучены линейные задачи, в которых краевые условия, так же как и само уравнение, линейны. Существует три типа краевых условий, которые называют условиями первого, второго и третьего рода. Условия первого рода означают, что на границах области задана зависимость температуры от времени:

$$u(0, t) = \mu_{11}(t), \quad u(L, t) = \mu_{12}(t). \quad (7.3')$$

Условия второго рода задают тепловые потоки (производные от температуры) через границы области:

$$u_x(0, t) = \mu_{21}(t), \quad u_x(L, t) = \mu_{22}(t). \quad (7.3'')$$

И, наконец, условия третьего рода задают на границе линейную комбинацию искомой функции и ее производной:

$$u(0, t) + \alpha_1 u_x(0, t) = \mu_{31}(t), \quad (7.3''')$$

$$u(L, t) + \alpha_2 u_x(L, t) = \mu_{32}(t) t^i$$

В курсе дифференциальных уравнений доказано, что уравнение (7.1) с начальными данными (7.2) и краевыми условиями (7.3'), или (7.3''), (7.3''') имеет единственное решение. Рассмотрим методы приближенного решения поставленной задачи.

7.2.2. Конечно-разностные схемы для одномерного уравнения

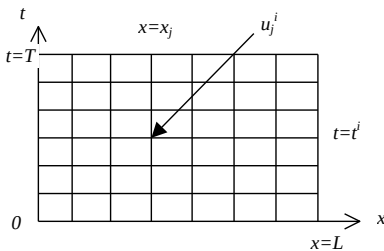


Рис. 7.1. Расчетная область и сетка

Введем в области решения прямоугольную равномерную разностную сетку. Для этого разобьем отрезок $[0, T]$ на M равных частей: $t^i = i \cdot \tau$, $\tau = T/M$; а отрезок $[0, L]$ – на N равных частей: $x_j = jh$, $h = L/N$. Вместо точного решения $u(x, t)$, будем искать при-

ближенное решение, заданное в узлах сетки $u_j^i = u(x_j, t^i)$. Область решения и построенная сетка представлены на рис. 7.1. На линиях $t=0$, $x=0$ и $x=L$ решение определено начальными данными и краевыми условиями, во всех остальных узлах сетки решение должно быть найдено из разностных аналогов уравнения (7.1). Приближим (аппроксимируем) исходную дифференциальную задачу конечно-разностной. Для этого заменим все входящие в уравнение (7.1) и краевые условия (7.3'), (7.3'') производные их конечно-разностными аналогами:

$$\begin{aligned}\frac{\partial u}{\partial t}(x_j, t^i) &\approx \frac{u(x_j, t^{i+1}) - u(x_j, t^i)}{\tau} = \frac{u_j^{i+1} - u_j^i}{\tau}, \\ \frac{\partial^2 u}{\partial x^2}(x_j, t^i) &\approx \frac{u(x_{j+1}, t^i) - 2u(x_j, t^i) + u(x_{j-1}, t^i)}{h^2} = \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2}, \\ \frac{\partial u}{\partial x}(x_0, t^i) &= \frac{u_1^i - u_0^i}{h}, \quad \frac{\partial u}{\partial x}(x_M, t^i) = \frac{u_M^i - u_{M-1}^i}{h}.\end{aligned}$$

Подставляя выражения для производных в уравнение, получим разностную схему:

$$\frac{u_j^{i+1} - u_j^i}{\tau} = A \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2} + F(x_j, t^i) \quad (7.4)$$

На первом временном слое решение известно: $u_j^0 = u(x_j, t^0) = u(x_j, 0) = u_0(x_j)$. Во всех внутренних точках расчетной области оно находится из явных формул, которые легко получаются из схемы (7.4):

$$\begin{aligned}u_j^{i+1} &= u_j^i + \tau A \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2} + F(x_j, t^i), \\ i &= 1, 2, \dots, N-1, \quad j = 0, 1, \dots, M-1.\end{aligned}$$

Для нахождения решения в крайних точках отрезка $[0, L]$ необходимо использовать краевые условия. Если заданы краевые условия первого рода, можно сразу определить значения искомых функций: $u_0^i = \mu_{11}(t^i)$, $u_M^i = \mu_{12}(t^i)$. Для условий второго рода получим: $u_0^{i+1} = u_1^{i+1} - h\mu_{21}(t^{i+1})$, $u_M^{i+1} = u_{M-1}^{i+1} + h\mu_{22}(t^{i+1})$.

Пусть $u(t, x)$ – точное решение. Исследуем, насколько численное решение, полученное по схеме (7.4) отличается от точного.

Для этого разложим $u(t^i, x_{j\pm 1})$, $u(t^{i+1}, x_j)$ в ряд Тэйлора в окрестности точки (x_j, t^i) :

$$u_{j\pm 1}^i = u(t^i, x_j \pm h) = u_j^i \pm h(u_x)_j^i + \frac{h^2}{2}(u_{xx})_j^i \pm \frac{h^3}{6}(u_{xxx})_j^i + \frac{h^4}{24}(u_{xxxx})_j^i \dots,$$

$$u_j^{i+1} = u(t^i + \tau, x_j) = u_j^i + \tau(u_t)_j^i + \frac{\tau^2}{2}(u_{tt})_j^i + \dots,$$

и подставим эти выражения в разностную схему (7.4):

$$\begin{aligned} & \frac{u_j^{i1} - u_j^i}{\tau} - A \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2} - F(x_j, t^i) = \\ & \frac{u_j^i + \tau(u_t)_j^i + \frac{\tau^2}{2}(u_{tt})_j^i + \dots - u_j^i}{\tau} - \\ & A \left(\frac{u_j^i + h(u_x)_j^i + \frac{h^2}{2}(u_{xx})_j^i + \frac{h^3}{6}(u_{xxx})_j^i + \frac{h^4}{24}(u_{xxxx})_j^i - 2u_j^i}{h^2} + \right. \\ & \left. \frac{u_j^i - h(u_x)_j^i + \frac{h^2}{2}(u_{xx})_j^i - \frac{h^3}{6}(u_{xxx})_j^i + \frac{h^4}{24}(u_{xxxx})_j^i \dots}{h^2} \right) - F(x_j, t^i) = \\ & (u_t)_j^i - A(u_{xx})_j^i - F(x_j, t^i) + \frac{\tau}{2}(u_{tt})_j^i - A \frac{h^2}{12}(u_{xxxx})_j^i. \end{aligned}$$

Первые три члена являются невязкой этого уравнения в точке (t^i, x_j) и равны 0, поскольку $u(x, t)$ – решение уравнения (7.1). Следовательно, погрешности этой схемы равна

$$\frac{\tau}{2}(u_{tt})_j^i - A \frac{h^2}{12}(u_{xxxx})_j^i,$$

т.е. схема является схемой первого порядка аппроксимации по времени и второго порядка – по пространству. Преимуществом явной схемы является то, что решение может быть найдено по явным алгебраическим формулам. Однако, как показали расчеты, приближенное решение, полученное с помощью явной схемы, может быть неустойчивым. Неустойчивость приводит к быстрому (экспоненциальному) росту погрешностей, вносимых в

численное решение за счет ошибок округления. Исследование устойчивости, выполненное на простейших решениях в виде единичной гармоники (Фурье-анализ) показывает, что эти решения будут устойчивы если

$$\gamma = A \frac{\tau}{h^2} \leq \frac{1}{2}. \quad (7.5)$$

Параметр γ называется числом Куранта. При нарушении условия (7.5) в численном решении возникают пилообразные осцилляции, амплитуда которых быстро растет, и за несколько временных шагов решение «разваливается».

Для иллюстрации приведем пример решения в пакете MathCAD уравнения (7.1), $A=1$, $F(x,t)=0$, с нулевыми краевыми условиями первого рода $u(0,t)=u(1,t)=0$ и с начальными данными в виде гауссоиды, центрированной относительно точки $x=1/2$:

$$\varphi(x,0) = e^{-20(x-0.5)^2} - e^{-20(x-1.5)^2} - e^{-20(x+0.5)^2} \quad (7.6)$$

Задача имеет точное решение

$$u(x,t) = \frac{1}{\sqrt{1+80t}} \left(e^{-\frac{20(x-0.5)^2}{1+80t}} - e^{-\frac{20(x-1.5)^2}{1+80t}} - e^{-\frac{20(x+0.5)^2}{1+80t}} \right),$$

график которого приведен на рис. 7.2.

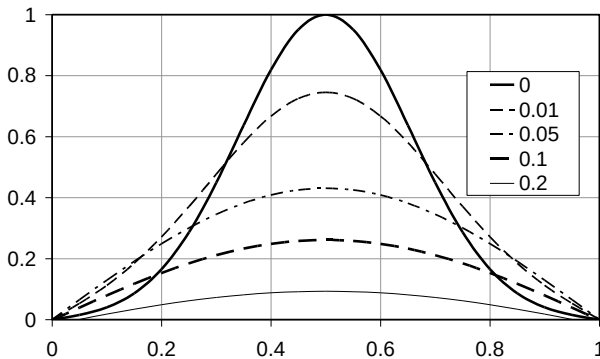


Рис. 7.2. Точное решение (7.1), (7.6) на различные моменты времени

Как показывает рис. 7.2, точное решение монотонно убывает со временем. Воспользуемся для решения явной схемой (7.4) на сетке $h=0,1$, $\tau=0,02$. Легко проверить, что в этом случае условие

(7.5) нарушается. Действительно, $A \frac{\tau}{h^2} = \frac{0.02}{0.01} = 2 > \frac{1}{2}$, и следует

ожидать, что решение будет неустойчиво. Приведенный на рис. 7.3 пример показывает, что уже через несколько временных шагов численное решение становится немонотонным, и в дальнейшем его график приобретает характерный «пилообразный» вид. Амплитуда «осцилляций» быстро растет, что приводит к переполнению арифметического устройства.

На практике условие (7.5) означает, что расчет приходится вести с очень маленьким шагом по временной переменной, что существенно ограничивает применение явных схем для решения уравнения теплопроводности. Действительно, пусть $h=10^{-2}$, $A=1$, тогда, согласно (7.5), для получения устойчивого решения необходимо соблюдать условие $\tau < 5 \cdot 10^{-5}$. Если решение надо получить на момент времени $T=1$, то для этого надо сделать $N=2 \cdot 10^4$ временных шагов. Если же решение надо получить на более подробной сетке по пространственной переменной, например $h=10^{-3}$, то число временных шагов возрастет до $N=2 \cdot 10^6$, и использование явной схемы делает решение задачи нереализуемым.

Применим для решения задачи (7.1)-(7.3) неявную схему:

$$\frac{u_j^{i+1} - u_j^i}{\tau} = A \frac{u_{j+1}^{i+1} - 2u_j^{i+1} + u_{j-1}^{i+1}}{h^2} + F(x_j, t^i). \quad (7.7)$$

Исследование аппроксимации показывает, что эта схема также имеет погрешность порядка $\tau^1 + h^2$. Схема устойчива при любом соотношении шагов τ , h . Это означает, что расчет можно вести со сколь угодно большим временным шагом. Такие схемы называют абсолютно устойчивыми.

Для получения решения необходимо на каждом временном шаге решить СЛАУ с трехдиагональной матрицей:

$$u_o^i = \mu_{11}(t^i),$$

$$\mu_{j-1}^{i+1} - (1 + 2\gamma)u_j^{i+1} + \mu_{j+1}^{i+1} = u_j^i + \tau F(x_j, t^i), i=1, 2, \dots, N-1. \quad (7.8)$$

$$u_M^i = \mu_{12}(t^i)$$

N:=10 M:=50 L:=1.0 T:=1 a:=1

fi0(x) := $\exp[-20(x-0.5)^2] - \exp[-20(x+0.5)^2] - \exp[-20(x-1.5)^2]$

psi1(t) := 0 левое краевое условие

psi2(t) := 0 правое краевое условие

$$h := \frac{L}{N}$$

$$h = 0.1$$

$$\tau := \frac{T}{M}$$

$$\tau = 0.02$$

шаги сетки

i:=0, 1..N

$x_i := i \cdot h$

$u_i := fi0(x_i)$

начальные данные

i =

0
1
2
3
4
5
6
7
8
9
10

$x_i =$

0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

$u_i =$

0
0.04
0.165
0.449
0.819
1
0.819
0.449
0.165
0.04
0

tn := 0

TN:=

for k ∈ 1..M

tn ← tn + tau

for j ∈ 1..N - 1

$$u1_j \leftarrow u_j + a \cdot \frac{\tau \cdot (u_{j-1} - 2 \cdot u_j + u_{j+1}))}{h^2}$$

u1_0 ← psi1(tn)

u1_N ← psi2(tn)

for p ∈ 0..N

u_p ← u1_p

u

k:=1..M

k =

1
2
3
4
5
6
7
8
9
10
11

	0
0	0
1	0.21
2	0.483
3	0.62
4	0.442
5	0.275
6	0.442
7	0.62
8	0.483
9	0.21
10	0

TN=

TN_i

u_i

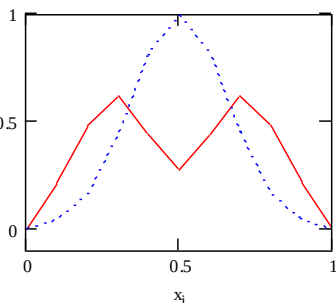


Рис. 7.3. Решение уравнения теплопроводности с помощью явной схемы

Решение системы (7.8) находится с помощью метода прогонки, описанного в главе 2.

На рис. 7.4. приведен листинг программы на MathCAD, где для решения задачи (7.1), (7.6) использована неявная схема (7.8).

Как показывает пример, неявная схема позволяет получить близкое к точному решение при достаточно большом значении шага по времени $\tau=0,067$.

Для случаев второй и третьей краевой задачи изменяются первое и последнее уравнения (7.8), из которых определяются значения первых прогоночных коэффициентов и решения в последнем узле сетки.

Схемы (7.4) и (7.7) являются представителями семейства двухслойных схем

$$\begin{aligned} \frac{u_j^{i+1} - u_j^i}{\tau} = \\ = A\sigma \frac{u_{j+1}^{i+1} - 2u_j^{i+1} + u_{j-1}^{i+1}}{h^2} + A(1-\sigma) \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2} + F(x_j, t^i), \end{aligned} \quad (7.9)$$

где $1 \geq \sigma \geq 0$ – параметр, который можно подбирать таким образом, чтобы добиться улучшения тех или других свойств схемы. При $\sigma=0$ схема (7.9) переходит в явную схему (7.4), а $\sigma=1$ – в чисто неявную схему (7.6).

При всех других значениях σ в каждом разностном уравнении будет завязано значения неизвестной функции в 6 разных точках, в отличие от схем (7.4), (7.8), в которых завязано по четыре различных точки. Графическое представление точек расчетной области, входящих в каждое разностное уравнение, называется шаблоном конечно-разностной схемы. Шаблоны схем (7.4), (7.8) и (7.9) при $\sigma \neq 0$ представлены на рис. 7.5 а-в, соответственно.

Как было указано выше, за счет выбора параметра σ , можно добиться, чтобы схема имела более высокий порядок аппроксимации. В частности, легко показать, что при $\sigma=0,5$ схема будет иметь порядок аппроксимации τ^2+h^2 . Кроме того, за счет специ-

ального выбора весового параметра $\sigma = \frac{1}{2} - \frac{h^2}{12A\tau}$ можно добиться-
ся, чтобы схема имела порядок аппроксимации $\tau^2 + h^4$.

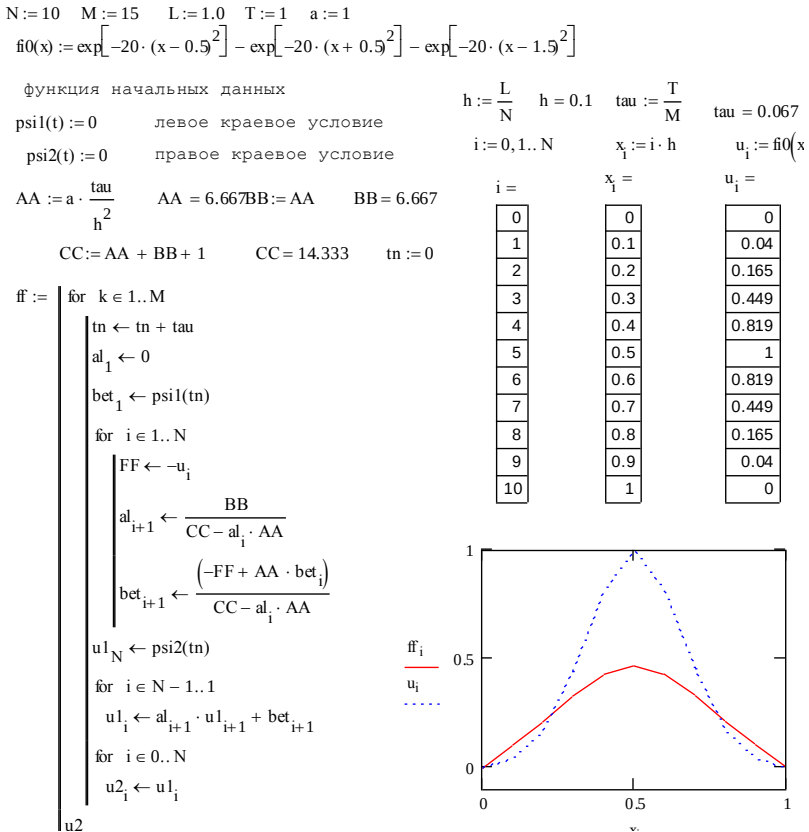


Рис. 7.4. Решение уравнения теплопроводности с помощью неявной схемы

Нахождение решения разностных схем типа (7.9) при $\sigma \neq 0$ аналогично случаю чисто неявной схемы. Система трехточечных уравнений, связывающих решение в точках верхнего $(i+1)$ -го слоя, имеет вид:

$$\sigma\gamma u_{j-1}^{i+1} - (1 + 2\sigma\gamma)u_j^{i+1} + \gamma\sigma u_{j+1}^{i+1} = u_j^i + \tau(1 - \sigma) \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2} +$$

$$+ \tau F(x_j, t^i), \quad i=0,1,\dots,M-1;$$

$$j=1,2,\dots,N-1.$$

Она отличается от уравнения (7.8) только правой частью, и, следовательно, также решается методом прогонки.

7.2.3. Конечно-разностные схемы для двумерной задачи

Пусть $G=[0,L_x]\times[0,L_y]$ – прямоугольная область на плоскости (x, y) , ∂G – граница области G , $u(x,y,t)$ – функция, определенная в области $G\times[0,T]$. Рассмотрим задачу нахождения $u(x,y,t)$, удовлетворяющего уравнению

$$\frac{\partial u}{\partial t} = A \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + F(x, y, t),$$

дополненного начальными данными

$$u(x,y,0) = u_0(x,y)$$

и краевыми условиями первого рода:

$$u(x,y,t) \big|_{\partial G} = \mu(t).$$

Введем в области $G\times[0,T]$ конечно-разностную сетку с шагами $h_x=L_x/N_x$, $h_y=L_y/N_y$ и $\tau=T/M$: $t^n=n\cdot\tau$, $x_i=i h_x$, $y_j=j h_y$. Построим семейство двухслойных конечно-разностных схем:

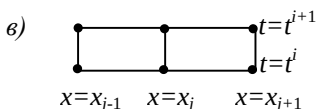
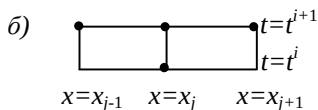
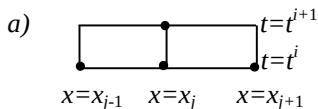


Рис. 7.5. Шаблоны схем семейства (7.9)

$$\frac{u_{ij}^{n+1} - u_{ij}^n}{\tau} = A\sigma \left(\frac{u_{i+1j}^{n+1} - 2u_{ij}^{n+1} + u_{i-1j}^{n+1}}{h_x^2} + \frac{u_{ij+1}^{n+1} - 2u_{ij}^{n+1} + u_{ij-1}^{n+1}}{h_y^2} \right) +$$

$$A(1-\sigma) \left(\frac{u_{i+1j}^n - 2u_{ij}^n + u_{i-1j}^n}{h_x^2} + \frac{u_{ij+1}^n - 2u_{ij}^n + u_{ij-1}^n}{h_y^2} \right) + F(x_i, y_j, t^n).$$

Можно видеть, что шаблон схемы, представленный на рис. 7.6, включает 9 точек на неизвестном, $n+1$ временном слое, и 9 точек на известном n -м слое. При $\sigma=0$ схема является явной, и ее решение можно найти по формулам:

$$u_{ij}^{n+1} = u_{ij}^n + \tau A \left(\frac{u_{i+1j}^n - 2u_{ij}^n + u_{i-1j}^n}{h_x^2} + \frac{u_{ij+1}^n - 2u_{ij}^n + u_{ij-1}^n}{h_y^2} \right) +$$

$$+ F(x_i, y_j, t^n), i=1,2,...,Nx; \quad j=1,2,...,Ny; \quad n=0,1,2,...,M-1. \quad (7.10)$$

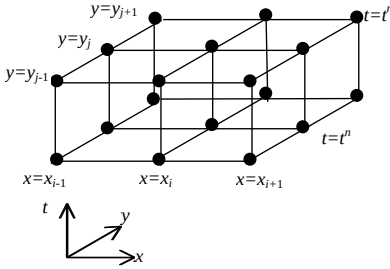


Рис. 7.6. Шаблон схемы для двумерного уравнения теплопроводности

Явная схема имеет порядок аппроксимации $\tau + h_x^2 + h_y^2$. Однако, как и в случае одной пространственной переменной, схема является условно-устойчивой. Для того, чтобы получить устойчивое приближенное решение, шаги разностной сетки должны удовлетворять условию Куранта:

$$A \left(\frac{\tau}{h_x^2} + \frac{\tau}{h_y^2} \right) \leq \frac{1}{2}.$$

Свойством безусловной устойчивости

схема будет обладать при $\sigma > \frac{1}{2} - \frac{h^2}{4\tau A}$, $h = \max(h_x, h_y)$.

При $\sigma \neq 0$ шаблон схемы (7.10) будет включать 9 точек на верхнем временном слое. Для нахождения решения необходимо для каждого t^n решать СЛАУ с заполненной матрицей, для кото-

рых экономичный метод прогонки не применим. В этом случае используются так называемые методы дробных шагов [7], в которых процесс нахождения решения на новом $n+1$ временном слое разбивается на несколько промежуточных (дробных) шагов таким образом, чтобы на каждом шаге по одному из пространственных направлений схема была явной, а по другому – неявной. Неявность схемы по выбранному направлению делает ее безусловно устойчивой. В то же время, для нахождения решения на новом временном слое не требуется решать СЛАУ с заполненной матрицей, а можно найти решение с помощью нескольких прогонок. Эта методика широко используется при решении многомерных уравнений теплопроводности. Существует много различных схем в дробных шагах. Приведем одну из возможных схем для двумерного уравнения теплопроводности.

$$\frac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^n}{\tau/2} = A \left(\frac{u_{i+1j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1j}^{n+\frac{1}{2}}}{h_x^2} + \frac{u_{ij+1}^n - 2u_{ij}^n + u_{ij-1}^n}{h_y^2} \right) + F(x_i, y_j, t^n)$$

$$\frac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\tau/2} = A \left(\frac{u_{i+1j}^{n+\frac{1}{2}} - 2u_{ij}^{n+\frac{1}{2}} + u_{i-1j}^{n+\frac{1}{2}}}{h_x^2} + \frac{u_{ij+1}^{n+1} - 2u_{ij}^{n+1} + u_{ij-1}^{n+1}}{h_y^2} \right) + F(x_i, y_j, t^{n+\frac{1}{2}}).$$

Сначала из первого разностного уравнения с помощью прогонки по направлению x надо найти решение на промежуточном $n+1/2$ временном слое. Затем из второго уравнения, также с помощью прогонки определяется решение на $n+1$ временном слое.

Пример 7.1. Предположим, что мы знаем распределение температуры плоской пластины в начальный момент времени. Заданы граничные условия на краях пластины. Нам необходимо решить нестационарную задачу, чтобы оценить поведение температуры в пластине на разные моменты времени. Такой процесс будет описываться параболическим уравнением в частных производных, неявная схема расчета которого методом дробных шагов приведена выше.

Итак, мы решаем уравнение вида

$$\frac{\partial u}{\partial t} = a \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right).$$

Здесь u — это температура, x и y — пространственные координаты, t — время. Коэффициент a полагается равным 1.

Листинг программы MathCAD для решения этой задачи приведен на рис. 7.7. Полученное распределение температуры на последний момент времени показано на рис. 7.8.

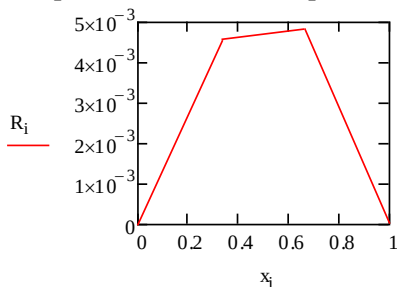


Рис. 7.7. Поведение ошибки (невязки) в примере 7.1.

```

Nx := 10  Ny := 10  M := 100  X := 1.0  Y := 1.0  T := 1  a := 1

f0(x,y) := sin(pi*x)-sin(pi*y)      функция начальных данных

u(t,x,y) := sin(pi*x)-sin(pi*y)-exp(-pi^2*t)

ly := Y/Ny      ly = 0.1      lx := X/Nx      lx = 0.1

i := 0..Nx-1      j := 0..Ny-1

xi := i*lx      yi := j*ly

tau := T/M      tau = 0.01      pa1(t,y) := 0  левое граничное условие
rx := a*tau/(2*lx^2)      rx = 0.5      pa2(t,x) := 0  нижнее граничное условие
ry := a*tau/(2*ly^2)      ry = 0.5      pa3(t,y) := 0  правое граничное условие
                                pa4(t,x) := 0  верхнее граничное условие

k := 0

w[i,j] := f0(xi,yi)

AAx := rx  BBx := AAx  CCx := 1+2*rx  AAy := ry  BBY := AAy  CCy := 1+2*ry
AAx = 0.5  BBx = 0.5  CCx = 2  AAy = 0.5  BBY = 0.5  CCy = 2

k := 0  B0x := 2*rx  C0x := 1+2*rx  B0y := 2*ry  C0y := 1+2*ry

```

Рис. 7.8. Программа расчета двумерного уравнения теплопроводности (пример 7.1) (начало)

```

unz(mn) > while t0 ≤ mn
    t0 ← t0 + tmu
    for j = 1..Ny - 1
        alj ← 0
        betj ← 0
        for i = 1..Nx - 1
            FF ← -ry·ui,j+1 - ui,j·(1 - 2ry) - ry·ui,j-1
            ali+1 ←  $\frac{BBx}{CCx - al_j \cdot AAx}$ 
            beti+1 ←  $\frac{-FF + AAx \cdot bet_j}{CCx - al_j \cdot AAx}$ 
        ulNx,j ← 0
        for i = Nx - 1..0
            uli,j ← ali+1·uli+1,j + beti+1
        for i = 0..Nx
            uli,0 ← 0
            uli,Ny ← 0
        for i = 1..Nx - 1
            ali ← 0
            beti ← 0
            for j = 1..Ny - 1
                FF ← -rx·uli+1,j - (1 - 2rx)·uli,j - rx·uli-1,j
                alj+1 ←  $\frac{BBy}{CCy - al_j \cdot AAy}$ 
                betj+1 ←  $\frac{-FF + AAy \cdot bet_i}{CCy - al_j \cdot AAy}$ 
            uli,Ny ← 0
            for j = Ny - 1..0
                uli,j ← alj+1·uli,j+1 + betj+1
        for j = 0..Ny
            ul0,j ← 0
            ulNx,j ← 0
        for i = 0..Nx
            for j = 0..Ny
                ui,j ← uli,j

```

Рис. 7.8. Программа расчета двумерного уравнения теплопроводности (пример 7.1) (окончание)

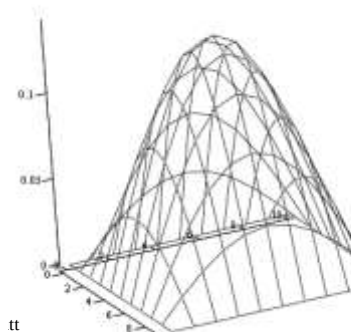


Рис. 7.9. Решение двумерного уравнения теплопроводности (пример 7.1)

7.3. Гиперболические уравнения

7.3.1. Постановка задачи

Типичным представителем уравнений гиперболического типа является так называемое волновое уравнение, описывающее распространение различных волн:

$$u_{tt} = g^2(x, t) u_{xx} + f(x, t). \quad (7.11)$$

Пусть требуется решить уравнение в области G : $x \in (0, 1)$, $t \in (0, T)$.

Дополним уравнение начальными данными

$$u(x, 0) = \sigma_1(x); \quad u_t(x, 0) = \sigma_2(x) \quad (7.12)$$

и краевыми условиями

$$\begin{aligned} p_0 u(0, t) + p_1 u_x(0, t) &= A(t) \\ s_0 u(1, t) + s_1 u_x(1, t) &= B(t). \end{aligned} \quad (7.13)$$

7.3.2. Явная конечно-разностная схема

Для приближенного решения будем использовать конечно-разностный метод. Для этого введем в области G разностную сетку, в качестве которой используем совокупность точек пересечения прямых $x = ih$, $t = j\tau$, $i = 0, 1, \dots, N$; $j = 0, 1, \dots, M$, где h и τ — шаги сетки по пространственной и временной координатам.

Если положить, что шаги h и τ связаны соотношением $\tau = rh$, $r = \text{const}$, то сетка будет зависеть только от одного параметра h .

Через u_i^j обозначим значение сеточной функции в точке (x_i, t^j) . Аппроксимируем входящие в (7.11)-(7.13) производные конечно-разностными соотношениями второго порядка точности:

$$\frac{\partial^2 u}{\partial t^2}(x_i, t^j) \approx \frac{u(x_i, t^{j+1}) - 2u(x_i, t^j) + u(x_i, t^{j-1}))}{\tau^2} = \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\tau^2},$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, t^j) \approx \frac{u(x_{i+1}, t^j) - 2u(x_i, t^j) + u(x_{i-1}, t^j))}{h^2} = \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2},$$

Подставив эти выражения в (7.11), получим явную разностную схему:

$$\frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\tau^2} = (g_i^j)^2 \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2} + f_i^j \quad (7.14)$$

Схеме (7.14) отвечает шаблон типа “крест”, изображенный на рис. 7.10. Он иллюстрирует тот факт, что для вычисления значения искомой функции на временном слое $j+1$ необходимо знать значения этой функции на двух предыдущих слоях j и $j-1$. Следовательно, для того, чтобы начать расчет, необходимо знать

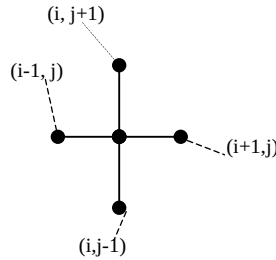


Рис. 7.10. Шаблон явной схемы для волнового уравнения

значения сеточной функции на первых двух временных слоях. Решение на временном слое $t = t_0$ определено начальными данными (7.12): $u_i^0 = \sigma_1(x_i)$. Для того чтобы вычислить решение при $t=t_1$, воспользуемся формулой Тэйлора, а также начальными данными (7.12) и уравнением (7.11):

$$u(t^1, x) = u(t^0, x) + \tau u_t(t^0, x) + \frac{\tau^2}{2} u_{tt}(t^0, x) + \frac{\tau^3}{6} u_{ttt}(t^0, x) + \dots =$$

$$\sigma_1(x) + \tau \sigma_2(x) + \frac{\tau^2}{2} \left(g^2(t^0, x) \frac{d^2 \sigma_1(x)}{dx^2} + f(t^0, x) \right) + C \tau^3.$$

Для нахождения значений сеточной функции u_i^j во внутренних точках $x_i = ih$, $i = 1, \dots, N-1$ на временных слоях t_j , $j = 2, 3, \dots, M$, используем разностную схему (7.14):

$$u_i^{j+1} = 2u_i^j - u_i^{j-1} + r^2 g^2(u_{i+1}^j - 2u_i^j + u_{i-1}^j) + \tau^2 f_i^j, \quad i = 1, 2, \dots, N-1. \quad (7.15)$$

Для определения искомой сеточной функции на линиях $x = x_0$, $x = x_N$, воспользуемся краевыми условиями. В случае первой краевой задачи ($p_1 = 0$, $s_1 = 0$) значения функции в граничных точках задаются точно: $u_0^j = A(t^j)$, $u_N^j = B(t^j)$. Если $p_1 \neq 0$, $s_1 \neq 0$ (вторая или третья краевая задачи), производные в соотношениях (7.13) необходимо заменить конечно-разностными соотношениями. Используем формулы первого порядка аппроксимации:

$$p_0 u_0^j + p_1 \frac{u_1^j - u_0^j}{h} = A^j, \quad s_0 u_N^j + s_1 \frac{u_N^j - u_{N-1}^j}{h} = B^j,$$

откуда

$$u_0^j = \frac{p_1 u_1^j - A^j h}{p_1 - p_0 h}, \quad u_N^j = \frac{B^j h + s_1 u_{N-1}^j}{h s_0 + s_1}, \quad j = 2, 3, \dots, M. \quad (7.16)$$

Можно показать, что схема (7.15) имеет второй порядок аппроксимации относительно h . Однако соотношения (7.16) имеют лишь первый порядок аппроксимации, что, несомненно, снижает общую точность полученного приближенного решения. Для того чтобы общий порядок аппроксимации задачи не понижался, для аппроксимации производных в граничных условиях (7.13) необходимо использовать соотношения второго порядка аппроксимации (см. раздел 4.1):

$$p_0 u_0^j + p_1 \frac{-3u_0^j + 4u_1^j - u_2^j}{2h} = A^j, \quad s_0 u_N^j + s_1 \frac{3u_N^j - 4u_{N-1}^j + u_{N-2}^j}{2h} = B^j,$$

откуда легко получить выражения для u_0^j , u_N^j , имеющие второй порядок аппроксимации.

7.3.3. Исследование устойчивости разностной схемы

Для того чтобы решение задачи (7.15) сходилось к решению исходной задачи, требуется, чтобы эта схема была устойчивой.

Опишем один из методов исследования устойчивости. Рассмотрим задачу Коши:

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} - g^2 \frac{\partial^2 u}{\partial x^2} &= 0, \quad -\infty < x < \infty, \quad 0 < t < T, \quad g = \text{const}, \\ u(x, 0) &= \psi_1(x), \quad \frac{\partial u(x, 0)}{\partial t} = \psi_2(x), \quad -\infty < x < \infty, \end{aligned} \quad (7.17)$$

которую аппроксимируем разностной схемой

$$\begin{aligned} \frac{u_m^{j+1} - 2u_m^j + u_m^{j-1}}{\tau^2} - g^2 \frac{u_{m+1}^j - 2u_m^j + u_{m-1}^j}{h^2} &= 0, \\ j &= 1, 2, \dots, M-1 \\ u_m^0 &= \psi_1(x_m), \quad \frac{u_m^1 - u_m^0}{\tau} = \psi_2(x_m), \quad m = 0, \pm 1, \dots \end{aligned} \quad (7.18)$$

Для устойчивости разностной схемы относительно возмущения начальных данных необходимо, чтобы решение задачи (7.18) удовлетворяло условию

$$\max_m |u_m^j| \leq C \cdot \max_m |u_m^0|, \quad j = 0, 1, \dots, M, \quad (7.19)$$

при произвольной ограниченной функции $\psi_1(x_m)$, в частности, для $\psi_1 = e^{i\alpha m}$, где α - вещественный параметр. Тогда решение задачи (7.17) можно искать в виде

$$u_m^j = \lambda^j e^{i\alpha m} \quad (7.20)$$

где $\lambda = \lambda(\alpha)$. Условие (7.20) выполняется, если числа $\lambda(\alpha)$ лежат внутри круга единичного радиуса, т.е.

$$|\lambda(\alpha)| \leq 1 \quad (7.21)$$

Неравенство (7.21) выражает необходимое условие устойчивости Неймана. Подставив (7.20) в (7.18), для определения $\lambda(\alpha)$ получим уравнение

$$\lambda^2 - 2(1 - 2r^2 g^2 \sin^2 \frac{\alpha}{2}) \lambda + 1 = 0 \quad (7.22)$$

По теореме Виета произведение корней этого уравнения равно 1, т.е. для выполнения условия (7.21) требуется, чтобы корни

$\lambda_{1,2}$ уравнения (7.22) были комплексно-сопряженными и лежали на единичной окружности. Для этого, в свою очередь, необходимо, чтобы дискриминант $D(\alpha)$ уравнения (7.22) был отрицателен:

$$D(\alpha) \equiv 4r^2 g^2 \sin^2 \frac{\alpha}{2} \left(r^2 g^2 \sin^2 \frac{\alpha}{2} - 1 \right) < 0.$$

Данное неравенство выполняется при всех α , если $gr \leq 1$. Следовательно, условием устойчивости схемы (7.18) будет

$$\tau \leq \frac{h}{g} \quad (7.23)$$

Пусть теперь $g=g(x,t) \neq const$. В этом случае применяется принцип “замороженных коэффициентов”, в соответствии с которым необходимое условие устойчивости Неймана можно записать в виде

$$\tau \leq \frac{h}{g_*}, \quad g_* = \max_{x,t} g(x,t) \quad (7.24)$$

В заключение отметим, что вопрос влияния граничных условий на устойчивость разностной схемы здесь не рассматривается.

7.3.4. Неявная разностная схема

При построении схемы (7.14) производная u_{xx} была заменена на конечную разность на временном слое $t_j = j\tau$. Если же использовать значения с временного слоя t_{j+1} , то получим схему

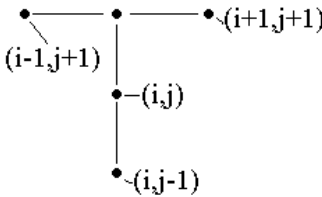
$$\frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{\tau^2} = g_i^{j+1/2} \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{h^2} + \tau^2 f_i^{j+1} \quad (7.25),$$

которой соответствует шаблон, изображенный на рис. 7.11.

Из уравнения (7.25) невозможно явно выразить u_i^{j+1} через значения функции u с предыдущих слоев по времени (j и $j-1$), поскольку в (7.22) наряду с u_i^{j+1} входят неизвестные u_{i+1}^{j+1} и u_{i-1}^{j+1} . Поэтому данная схема называется неявной. Анализ устойчивости показывает, что неявная схема безусловно устойчива,

т.е. обеспечивает сходимость разностной задачи к решению соответствующей дифференциальной при любом отношении τ/h . Решение на первых двух временных слоях определяется из начальных данных так же, как это сделано для явной схемы. Обозначив $\gamma = g_i^{j+1/2} r^2$, перепишем (7.25) в виде

$$u_{i+1}^{j+1} - (1 + 2\gamma)u_i^{j+1} + u_{i-1}^{j+1} = -2u_i^j + u_i^{j-1} + \tau^2 f_i^{j+1}, i = 1, 2, \dots, N-1 \quad (7.26)$$



Дополнив (7.26) формулами, аппроксимирующими краевые условия, получим СЛАУ с трехдиагональной матрицей, которая решается с помощью метода прогонки (см. главу 2).

Рис. 7.11. Шаблон неявной схемы для волнового уравнения

7.4. Приближенные методы решения уравнения Пуассона

В качестве классического представителя уравнений эллиптического типа рассмотрим двумерное уравнение Пуассона:

$$\Delta u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = F(x, y), \quad (7.27)$$

$F(x, y)$ – функция источников. Уравнение Пуассона описывает, например, распределение электростатического поля или стационарное распределение температуры. Частным случаем этого уравнения является уравнение Лапласа $\Delta u = 0$.

Пусть требуется определить решение в некоторой области G на плоскости (x, y) . Корректная постановка задачи требует задания граничных условий на границе этой области ∂G .

В одномерном случае уравнение Пуассона не что иное, как краевая задача первого рода для ОДУ второго порядка, решение которой было рассмотрено в главе 6.

Сравнивая уравнение Пуассона и рассмотренное выше двумерное уравнение теплопроводности, можно понять, что урав-

нение Пуассона является стационарным, т.е. независящим от времени вариантом уравнения теплопроводности. Поэтому для решения уравнения Пуассона часто используют так называемый метод установления. Для этого в правую часть уравнения (7.27) добавляют слагаемое $\partial u / \partial t$, и решают полученное уравнение теплопроводности с помощью описанных в предыдущем параграфе методов до тех пор, пока решение не выйдет на стационар, т.е. не перестанет изменяться в зависимости от времени. Время в этой задаче является фиктивным, и в разностных схемах надо использовать максимально возможный шаг. Решение нестационарной задачи стремится к решению стационарной независимо от выбора начальных данных. Процесс установления решения может занять продолжительное время, особенно если используются явные схемы, имеющие жесткое ограничение на временной шаг. В этом случае применение схем дробных шагов помогает существенно сократить время решения.

Кроме того, для решения уравнения Пуассона используются и другие методы, не связанные со сведением его к уравнению теплопроводности. Эти методы положены в основу стандартных функций пакета MathCAD.

Решение будем искать на плоской квадратной области, состоящей из $(M+1) \times (M+1)$ точек, при этом требуется, чтобы M было степенью числа 2, т.е. $M=2^n$, где n - некоторое целое число. Граничные условия должны быть определены пользователем на четырех сторонах квадрата. Простейший вариант – нулевые граничные условия. В этом случае можно использовать встроенную функцию:

`Multigrid(F,ncycle)`

Здесь F – матрица размера $(M+1) \times (M+1)$, содержащая правую часть уравнения (7.27) в узлах разностной сетки, `ncycle` – параметр численного алгоритма (количество циклов в пределах каждой итерации). Параметр `ncycle` в большинстве случаев можно выбрать равным 2. На рис. 7.10 приведен листинг программы с использованием функции `multigrid` для решения краевой задачи для уравнения Пуассона на сетке из 33×33 узлов. Функция правой части $F(x,y)$ представляет собой так называемую

мый точечный источник тепла, т.е. $F(x,y)=0$ всюду, кроме одной точки с номером (15, 20), в которой она принимает значение 10^4 .

Для решения краевой задачи с ненулевыми краевыми условиями можно использовать встроенную функцию `relax(a,b,c,d,e,F,v,r)`.

`M=32`

`FM,M = 0`

`F15,20 = 104`

`G=multigrid(-F, 2)`

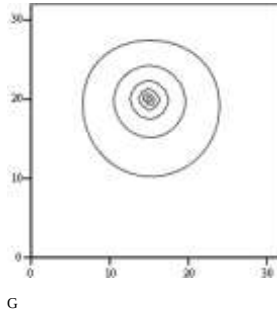


Рис. 7.12. Решение уравнения Пуассона с помощью функции `multigrid`

Здесь параметры a , b , c , d , e – квадратные матрицы коэффициентов разностной схемы, аппроксимирующей уравнение, F – квадратная матрица, задающая правую часть уравнения, v – квадратная матрица граничных условий и начального приближения к решению. Последний параметр r , характеризующий скорость сходимости метода, должен лежать на интервале $(0,1)$. На рис. 7.13 приведен пример программы на MathCAD, иллюстрирующий применение этой функции для решения уравнения (7.27), правая часть которого представляет собой три точечных источника, заданных в точках сетки с номерами (15, 20), (25, 30) и (10, 10).

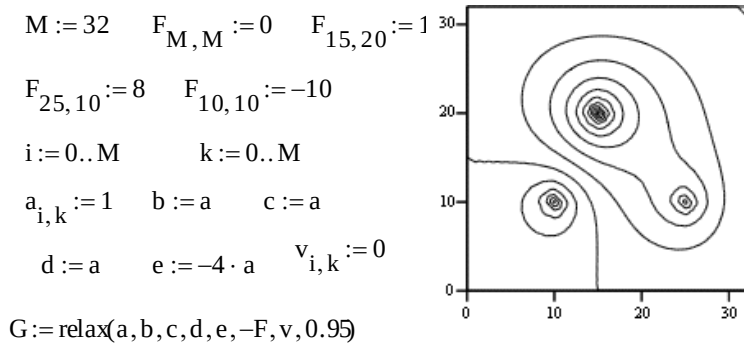


Рис. 7.13. Решение уравнения Пуассона с помощью функции `relax`

Заключение

В учебном пособии приведены методы приближенного решения нелинейных алгебраических уравнений, систем линейных алгебраических уравнений, методы интерполяции, оптимизации, численного дифференцирования и интегрирования, вычислительные подходы к решению задач Коши и краевых задач для обыкновенных дифференциальных уравнений, а также конечно-разностные методы решения уравнений в частных производных. Даны примеры реализаций данных численных подходов как на основе стандартных функции пакета MathCAD, так и с помощью средств программирования данной среды.

Тем не менее, читатель должен понимать, что приведенные в данном пособии методы являются лишь частью методик вычислительного моделирования, являясь верхушкой постоянно пополняемого набора численных подходов, разрабатываемых для широкого круга задач практики. Вводное знакомство с ними, предложенное в данном пособии, поможет начинающему исследователю или инженеру сориентироваться в данной тематике и приступить к самостоятельному изучению необходимых численных технологий, используя литературные источники, некоторые из которых приведены ниже в списке рекомендованной литературы.

Список рекомендованной литературы

1. Беклемишев Д.В. Курс аналитической геометрии и линейной алгебры / Д.В. Беклемишев. М.: ФИЗМАТЛИТ, 2005. – 304 с.
2. Портал DspSys.org: Системы цифровой обработки сигналов и данных [Электронный ресурс] / Режим доступа: <http://dspsys.org/fourier/56-fourier-transf> (дата обращения: 01.10.2013).
3. Поршнев С.В. Численные методы на базе Mathcad: учеб. Пособие / С.В. Поршнев, И.В. Беленкова. – СПб.: БХВ-Петербург, 2005. – 456 с.
4. Кирьянов Д.В. MathCAD 14. –СПб.: БХВ – Петербург, 2007. 704 с.
5. Метод конечных элементов и САПР / Сабоннадьер Ж.-К., Кулон Ж.-Л. – М.: Мир, 1989. – 190 с.
6. Зенкевич О. Конечные элементы и аппроксимация / О. Зенкевич, К. Морган. М.: Мир, 1986. — 318 с.
7. Вержбицкий В.М. Основы численных методов: учебник для ВУЗов / В.М. Вержбицкий. М.: Высшая школа, 2009. – 840 с.
8. Численные методы решения инженерных задач в пакете MathCAD / И.А. Бедарев, О.Н. Белоусова, Н.Н. Федорова; - Новосиб. гос. архитектур.-строит. ун-т. – Новосибирск, 2005. – 96 с.
9. Ильин В.П. Численные методы решения задач строительной механики: справ. Пособие / В.П. Ильин, В.В. Карпов, А.М. Масленников. – Минск: Вышэйшая школа, 1990. – 349 с.
10. Воскобойников Ю.Е. Решение инженерных задач в пакете MATHCAD: учеб. пособие / Ю.Е. Воскобойников, А.Ф. Задорожный, Л.А. Литвинов, Ю.Г. Черный. – Новосибирск: НГАСУ (Сибстрин), 2013. – 120 с.
11. Воскобойников Ю.Е. Основы вычислений и программирования в пакете MathCAD: Учеб. Пособие / Ю.Е. Воскобойников, А.Ф. Задорожный, Л.А. Литвинов, Ю.Г. Черный. – Новосибирск: НГАСУ (Сибстрин), 2012. – 212 с.
12. Бахвалов Н.С., Н.П., Жидков, Г.М Кобельков Численные методы / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. – М.: Бином. Лаборатория знаний, 2008. – 640 с.
13. Методы вычислений: учеб. пособие / И.А. Бедарев, Ю.В. Кратова, Н.Н. Федорова; - Новосиб. гос. архитектур.-строит. ун-т. – Новосибирск, 2009. – 112 с.
14. И.А. Бедарев, Н.Н. Федорова, И.А. Федорченко Компьютерное моделирование в задачах строительства. Учебное пособие. Новосибирск: НГАСУ (Сибстрин), 2012. - 152 с.