ковский государственный технический университет
имени Н.Э. Баумана

Учебное пособие

В.М. ГРАДОВ

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ПРАКТИКЕ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Часть 1

Издательство МГТУ имени Н.Э. Баумана

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. БАУМАНА

В.М. ГРАДОВ

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ПРАКТИКЕ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Часть 1

Рекомендовано редсоветом МГТУ им. Н.Э. Баумана в качестве учебного пособия

Москва
Издательство МГТУ им. Н.Э. Баумана
2005

УДК 51.001.57(075.8)
ББК 22в6
Г75

Рецензенты: *В.Н. Бакулин, Г.Н. Кувыркин*

Градов В.М.

Г75 Компьютерные технологии в практике математического моделирования: Учебное пособие. – Ч. 1. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. – 108 с: ил.

ISBN 5-7038-2628-4

Рассмотрены методы вычислений и средства их компьютерной реализации, используемые в различных областях инженерной и научной деятельности. Описание методов ориентировано на их реальное воплощение в работающее программное обеспечение с использованием языков высокого уровня. Даны рекомендации по эффективному применению алгоритмов. Значительное место в пособии отводится использованию в расчетной практике современных математических пакетов (Matlab и Mathcad), в каждом разделе приведены примеры использования данных пакетов для решения соответствующих задач.

Для студентов технических университетов.

Ил. 13. Табл. 2. Библиогр. 11 назв.

УДК 51.001.57(075.8)
ББК 22в6

ISBN 5-7038-2628-4

ПРЕДИСЛОВИЕ

В учебном пособии рассматриваются методы вычислений и средства их реализации, используемые в различных областях инженерной и научной деятельности. Возникающие здесь задачи отличаются разнообразием постановок, большим объемом вычислительной работы, конкретным научно-техническим выходом, широким спектром используемых методов решения, необходимостью завершения вычислительного процесса получением обширных числовых массивов и во многих случаях графическим представлением результатов моделирования. Проводится отбор численных методов для типичных задач, встречающихся в научно-технической практике и инженерных расчетах [1–8]. Описание методов ориентировано на компьютерную реализацию соответствующих алгоритмов и позволяет осуществить их реальное воплощение в работающее программное обеспечение с использованием различных языков высокого уровня (C, Pascal, Fortran) и инструментальных сред. Особенности алгоритмов иллюстрируются многочисленными примерами.

Одновременно значительное внимание в пособии уделено применению иного подхода к решению разнообразных проблем моделирования. Речь идет об использовании в расчетной практике современных математических пакетов: прежде всего таких пакетов, как Matlab и Mathcad, выделяющихся среди других вычислительных систем богатством реализованных в них численных методов. С помощью названных систем пользователь получает доступ к апробированным реализациям отдельных методов, дающий возможность формулировать весьма сложные вычислительные модели за счет комбинирования частных реализаций. Обе системы обладают встроенными языками программирования, что открывает широкие перспективы по разработке собственных алгоритмов, и имеют мощные средства визуализации результатов вычислений. Эти возможности пакетов использованы в примерах, рассмотренных в пособии.

Следует отметить, что в Matlab имеется среда GUIDE визуальной разработки приложений с графическим интерфейсом пользователя. Обе системы могут быть интегрированы друг с другом, а также с Excel. Пакеты функционируют в режиме интерпретации, однако в Matlab имеется модуль Matlab Compiler, который дает возможность компилировать программы, созданные на языке пакета. В Matlab можно использовать программы, написанные на языках C и Fortran, а из программ на C и Fortran можно получить доступ к огромному набору функций Matlab. Указанные и многие другие возможности пакетов описаны в работах [9–11].

1. ИНТЕРПОЛЯЦИЯ ФУНКЦИЙ ОДНОЙ И НЕСКОЛЬКИХ ПЕРЕМЕННЫХ

В задачах математического моделирования, которые для своей реализации требуют, как правило, применения ЭВМ, из различных способов задания функций наиболее часто используется табличный способ. При этом возникает необходимость определения значений функции для значений аргумента, отличных от тех, которые содержатся в таблице. В этих целях применяют процедуру интерполяции, а также экстраполяции. По сути, речь идет о восстановлении непрерывной функции по заданному набору ее значений для счетного множества значений аргумента. Задача восстановления функции может ставиться и иначе, когда ищется функция, близкая в определенном смысле к таблично заданной функции (например, наилучшее среднеквадратичное приближение). В обоих случаях исходную функциональную зависимость $y(x)$, по которой составлена таблица, заменяют приближенной функцией $\varphi(x, \overline{a})$, которую и используют в дальнейших вычислениях. При этом близость $y(x)$ и $\varphi(x, \overline{a})$ обеспечивается подбором свободных параметров $\overline{a} = \{a_0, a_1, ..., a_n\}$.

В простейшем варианте лагранжевой интерполяции, когда функция $y(x)$ задана в узлах некоторой сетки x_i, параметры $\overline{a}$ определяют из условия совпадения $\varphi(x, \overline{a})$ со значениями функции в фиксированном числе узлов.

Получаемая при этом система уравнений имеет вид

$$\varphi(x_i, a_0, a_1, ..., a_n) = y(x_i) \equiv y_i\ ,\ \ 0 \le i \le n\ . \qquad (1.1)$$

Из этой системы можно определить все a_i.

Интерполяция может быть линейной или нелинейной в соответствии с характером зависимости функции $\varphi(x, \overline{a})$ от параметров $\overline{a}$.

1.1. Линейная интерполяция

При линейной интерполяции функция $\varphi(x, \overline{a})$ имеет вид

$$\varphi(x, \overline{a}) = \sum_{j=0}^{n} a_j \varphi_j(x), \tag{1.2}$$

где все функции $\varphi_j(x)$ линейно независимы.

Подставляя (1.2) в (1.1), получим систему линейных уравнений для определения a_i:

$$\sum_{j=0}^{n} a_j \varphi_j(x_i) = y_i, \quad 0 \le i \le n. \tag{1.3}$$

Единственность решения обеспечивается требованием неравенства нулю определителя системы (1.3):

$$\Delta = \begin{vmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \varphi_n(x_0) \\ . & \dots & \\ \varphi_0(x_n) & \varphi_1(x_n) & \varphi_n(x_n) \end{vmatrix} \neq 0 \tag{1.4}$$

при $x_k \neq x_l$ (т. е. при любых несовпадающих узлах).

Систему функций, удовлетворяющую условию (1.4), называют чебышевской. Из различных систем функций наиболее распространены многочлены, хотя применяют также тригонометрические и экспоненциальные функции.

Если в качестве системы функций выбрать степенные функции аргумента, т. е. $\varphi_j(x) = x^j$, то определитель (1.4) окажется определителем Вандермонда, который не равен нулю при условии $x_k \neq x_l$. Следовательно, интерполяционный полином всегда существует и он единствен.

Для практических вычислений удобно использовать многочлен $P_n(x)$ в форме интерполяционного полинома Ньютона. Введем понятие разделенных разностей функции $y(x)$, заданной в узлах x_i:

$$y(x_i, x_j) = [y(x_i) - y(x_j)]/(x_i - x_j),$$

$$y(x_i, x_j, x_k) = [y(x_i, x_j) - y(x_j, x_k)]/(x_i - x_k), \qquad (1.5)$$

$$y(x_i, x_j, x_k, x_l) = [y(x_i, x_j, x_k) - y(x_j, x_k, x_l)]/(x_i - x_l).$$

Правило образования таких конструкций понятно из приведенной записи. Разделенные разности первого, второго и более высоких порядков связаны с производными соответствующих порядков.

Рассмотрим разделенные разности полинома $P_n(x)$. Многочлен $P_n(x) - P_n(x_0)$ обращается в нуль при $x = x_0$, поэтому он делится нацело на $(x - x_0)$, т. е. разделенная разность первого порядка многочлена n-й степени

$$P(x, x_0) = [P_n(x) - P_n(x_0)]/(x - x_0) \qquad (1.6)$$

есть многочлен степени $(n-1)$ относительно x, а в силу симметрии – и относительно x_0.

Разделенная разность второго порядка

$$P(x, x_0, x_1) = [P(x, x_0) - P(x_0, x_1)]/(x - x_1) \qquad (1.7)$$

по аналогии также есть многочлен, степень которого равна $(n-2)$, так как разность $P(x, x_0) - P(x_0, x_1)$ делится нацело на $(x - x_1)$.

Продолжая указанный процесс, придем к тому, что разность n-го порядка $P(x, x_0, x_1, ..., x_{n-1})$ окажется константой, т. е. многочленом нулевой степени, и все разделенные разности более высокого порядка – равными нулю, т. е.

$$P(x, x_0, x_1, ..., x_n) = [P(x, x_0, x_1 ... x_{n-1}) - P(x_0, x_1 ... x_n)]/(x - x_n) = 0.$$

Из выражений (1.6), (1.7) следует:

$$P_n(x) = P_n(x_0) + (x - x_0)P(x, x_0),$$

$$P(x, x_0) = P(x_0, x_1) + (x - x_1)P(x, x_0, x_1).$$

Выполняя далее указанные преобразования, получим

$$P_n(x) = P_n(x_0) + (x - x_0)P(x_0, x_1) + (x - x_0)(x - x_1)P(x_0, x_1, x_2) + \ldots +$$
$$+ (x - x_0)(x - x_1)\ldots(x - x_{n-1})P(x_0, x_1, x_2, \ldots, x_n). \qquad (1.8)$$

Таким образом, мы выразили многочлен n-й степени через его значения в $(n+1)$ узлах: $x_0, x_1, x_2, \ldots, x_n$. Ввиду того, что значения интерполяционного полинома в этих узлах совпадают со значениями интерполируемой функции, разделенные разности, входящие в (1.8), выражаются через узловые значения функции. В результате получается полином, называемый интерполяционным многочленом Ньютона:

$$y(x) \approx y(x_0) + \sum_{k=1}^{n} (x - x_0)(x - x_1)\ldots(x - x_{k-1})y(x_0, \ldots, x_k). \quad (1.9)$$

При вычислениях по этой формуле точность расчетов удобно оценивать, наблюдая за тем, насколько быстро убывают члены ряда. Если это происходит достаточно быстро, можно оставлять только те члены, которые больше заданной погрешности расчетов. В (1.9) безразличен порядок нумерации узлов, что очень удобно при подключении новых узлов для построения полинома более высокого порядка.

Погрешность многочлена Ньютона можно оценить по формуле

$$\left|y(x) - P_n(x)\right| \le \frac{M_{n+1}}{(n+1)!}\left|\overline{\omega}_n(x)\right|,$$

где $M_{n+1} = \max\left|y^{(n+1)}(\xi)\right|$ – максимальное значение производной интерполируемой функции на отрезке между наименьшим и наибольшим из значений $x_0, x_1, x_2, \ldots, x_n$, а полином $\overline{\omega}_n(x) = \prod_{i=0}^{n}(x - x_i)$.

На равномерной сетке узлов приведенная формула трансформируется к виду

$$\left|y(x) - P_n(x)\right| < \sqrt{\frac{2n}{\pi}} M_{n+1}\left(\frac{h}{2}\right)^{n+1}, \qquad (1.10)$$

где $h = x_{i+1} - x_i = x_i - x_{i-1} = \cdots$ Предполагается при этом, что шаг сетки постоянен. Выражение (1.10) свидетельствует, что с уменьшением расстояния между узлами (шага) h погрешность представления функций полиномом Ньютона убывает как $O(h^{n+1})$.

Трудность использования указанных теоретических оценок на практике состоит в том, что производные интерполируемой функции обычно неизвестны, поэтому для определения погрешности удобнее воспользоваться оценкой первого отброшенного члена.

Пример 1.1. Построить интерполяционный полином Ньютона четвертой степени для функции $y(x) = \cos(2x)$ в области значений аргумента $0 \le x \le \pi/4$.

Заполним таблицу разделенных разностей (табл. 1), вычисляемых по пяти узлам, представив для удобства вычислений $y(z) = \cos[(\pi/2)z]$, $0 \le z \le 1$.

Таблица 1

z_0	$y(z_0)$	$y(z_0, z_1)$	$y(z_0, z_1, z_2)$	$y(z_0, \ldots, z_3)$	$y(z_0, \ldots, z_4)$
0	1				
		−0,304			
0,25	0,924		−1,128		
		−0,868		0,363	
0,5	0,707		−0,856		0,149
		−1,296		0,512	
0,75	0,383		−0,472		
		−1,532			
1	0				

Полином Ньютона

$$y(z) \approx 1 - 0,304z - 1,128z(z-0,25) + 0,363z(z-0,25)(z-0,5) + \\ + 0,149z(z-0,25)(z-0,5)(z-0,75).$$

Вычислим

$$y(0,6) \approx 1 - 0,182 - 0,237 + 7,623 \cdot 10^{-3} - 4,694 \cdot 10^{-3} = 0,589.$$

Точное значение $y(0,6) = 0,588$, т. е. точность вычислений по приближенной формуле оказалась весьма высокой.

Помимо полинома Ньютона в практике вычислений находит применение еще один полином, называемый полиномом Эрмита. Его используют, если в узлах задана не только функция, но и ее производные различного порядка. В этом случае целесообразно осуществлять так называемую эрмитову интерполяцию, когда в узлах совпадают не только значения заданной функции, но и значения производных. Полином, обладающий указанным свойством, обозначают $H_n(x)$.

Вывод формулы для полинома $H_n(x)$ производят построением по $(n+1)$ узлам полинома Ньютона. В областях между узлами средние наклоны кривых $y(x)$ и полинома совпадают. Если сближать узлы x_i и x_{i+1}, то средний наклон будет все точнее передавать первую производную функции. В пределе после совпадения узлов получают искомый многочлен $P_n(x, x_0, x_1, ..., x_i, x_{i+1}, ..., x_n)$ с кратным узлом, являющийся полиномом Эрмита n-й степени. В общем случае полином Эрмита

$$H_n(x) = P_n(x, \underbrace{x_0, x_0, ..., x_0}_{n_0}, x_1, x_1, ..., x_1, x_2, x_2, ..., x_2, \underbrace{x_k, x_k, ..., x_k}_{n_k}),$$

$$\sum_{l=0}^{k} n_l = n+1, \qquad (1.11)$$

в произвольном узле x_l имеет производные, значения которых равны производным интерполируемой функции вплоть до порядка $n_l - 1$. Использовать формулу (1.11) в вычислениях непосредственно нельзя, так как в выражениях для разделенных разностей появляется неопределенность типа 0/0. При кратности узлов не выше двух формулы для разделенных разностей получают предельным переходом:

$$y(x_0, x_0) = \lim_{x_1 \to x_0} \frac{y(x_0) - y(x_1)}{x_0 - x_1} = y'(x_0),$$

$$y(x_0, x_0, x_1) = \frac{y(x_0, x_0) - y(x_0, x_1)}{x_0 - x_1} = \frac{y'(x_0) - y(x_0, x_1)}{x_0 - x_1},$$

$$y(x_0, x_0, x_1, x_1) = \frac{y(x_0, x_0, x_1) - y(x_0, x_1, x_1)}{x_0 - x_1} =$$

$$= \frac{y'(x_0) - 2y(x_0, x_1) + y'(x_1)}{(x_0 - x_1)^2}.$$

Выражения для разделенных разностей в случае узлов кратности выше двух удобнее находить дифференцированием полинома Ньютона (1.9). В итоге общее выражение для разделенных разностей при кратности узлов m имеет вид

$$\underbrace{y(x_0, x_0, \ldots, x_0)}_{m} = \frac{1}{(m-1)!} y^{(m-1)}(x_0).$$

Пример 1.2. Построить полином Эрмита, передающий в двух узлах значения функции и ее первой производной:

$$P_n(x, x_0, x_0, x_1, x_1) = y(x_0) + (x - x_0) y(x_0, x_0) +$$
$$+ (x - x_0)^2 y(x_0, x_0, x_1) + (x - x_0)^2 (x - x_1) y(x_0, x_0, x_1, x_1) =$$
$$= y(x_0) + (x - x_0) y'(x_0) + (x - x_0)^2 \frac{y'(x_0) - y(x_0, x_1)}{x_0 - x_1} +$$
$$+ [y'(x_0) - 2y(x_0, x_1) + y'(x_1)] \frac{(x - x_0)^2 (x - x_1)}{(x_0 - x_1)^2}.$$

Помимо полинома Ньютона существует еще одна важная форма интерполяционного многочлена – полином в форме Лагранжа

$$L_n(x) = \sum_{i=0}^{n} L_i^{(n)}(x) y(x_i),$$

где лагранжевы коэффициенты $L_i^{(n)}(x)$ определяются по формуле

$$L_i^{(n)}(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{(x - x_k)}{(x_i - x_k)}.$$

Полином Лагранжа в отличие от полинома Ньютона содержит в явном виде значения интерполируемой функции в узлах $y(x_i)$.

Отметим также, что в практике вычислений для интерполяции полиномы степени выше пятой обычно не используют, т. е. число узлов интерполяции не превышает шести. Если при таком числе узлов не обеспечивается заданная погрешность, следует уменьшать расстояние между узлами.

1.2. Нелинейная интерполяция

Для табулирования быстроменяющихся функций требуется весьма малый шаг, т. е. возникает необходимость создавать таблицы очень больших объемов, что в ряде случаев неприемлемо. Оказывается, что преобразованием переменных $\eta = \eta(y)$ и $\xi = \xi(x)$ можно добиться того, чтобы в новых переменных график $\eta(\xi)$ был близок к прямой хотя бы на отдельных участках. В этом случае интерполяцию проводят в переменных (η, ξ), а затем обратным интерполированием находят $y_i = y(\eta_i)$.

Преобразования $\eta(y)$ и $\xi(x)$ должны быть достаточно простыми с использованием логарифмической, экспоненциальной, тригонометрической и некоторых других функций. При этом надо заботиться о том, чтобы и обратное преобразование $y(\eta)$ оказалось несложным. Во многих задачах теплофизики, гидродинамики, оптики, переноса излучения и других областей науки и техники часто встречается степенная зависимость функции от своих аргументов. В этом случае удобны преобразования типа логарифмирования.

Вообще говоря, в каждом конкретном случае приходится специально подбирать вид функций $\eta = \eta(y)$ и $\xi = \xi(x)$. Например, функция $y = \dfrac{a}{b + ce^{dx}}$ может быть представлена следующим образом:

$$\ln\left(\frac{a}{y} - b\right) = dx + \ln c, \quad \text{т. е.} \quad \eta = \ln\left(\frac{a}{y} - b\right), \quad \xi \equiv x.$$

Для функциональной зависимости $y = \dfrac{x}{ax + b}$ линеаризация достигается заменой переменных $\eta = \dfrac{1}{y}$, $\xi = \dfrac{1}{x}$, а для функции $y = axe^{-bx}$ – заменой $\eta = \ln(\dfrac{y}{x})$, $\xi \equiv x$.

Пример 1.3. Получить формулу для нелинейной двухточечной интерполяции функции $y(x)$, если переменные можно преобразовать по формулам $\eta = \ln(y)$ и $\xi = x$.

Составим интерполяционный полином Ньютона на двухточечном шаблоне

$$\eta = \eta_0 + \frac{\eta_1 - \eta_0}{\xi_1 - \xi_0}(\xi - \xi_0).$$

В исходных переменных имеем

$$\ln(y) = \ln(y_0) + \frac{\ln(y_1) - \ln(y_0)}{x_1 - x_0}(x - x_0)$$

и окончательно

$$y = y_0 (y_1 / y_0)^{(x - x_0)/(x_1 - x_0)}.$$

1.3. Интерполяция сплайнами

Слово сплайн переводится как «гибкая линейка». Такую линейку можно использовать для проведения кривых через заданную совокупность точек, изгибая и придерживая ее так, чтобы ребро проходило через все точки на плоскости. Равновесие гибкой линейки описывается уравнением $\psi^{IV}(x) = 0$, т. е. интерполяционный полином на участке между каждой парой соседних точек имеет третью степень:

$$\psi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \quad (1.12)$$

$$x_{i-1} \le x \le x_i, \ 0 \le i \le N.$$

В узлах значения многочлена и интерполируемой функции совпадают:

$$\psi(x_{i-1}) = y_{i-1} = a_i, \quad (1.13)$$

$$\psi(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3, \quad (1.14)$$

$$h_i = x_i - x_{i-1}, \quad 1 \le i \le N.$$

Число таких уравнений меньше числа неизвестных в два раза. Недостающие уравнения получают, приравнивая во внутренних узлах первые и вторые производные, вычисляемые по коэффициентам на соседних участках:

$$\psi'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$

$$\psi''(x) = 2c_i + 6d_i(x - x_{i-1}), \quad x_{i-1} \le x \le x_i,$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2, \tag{1.15}$$

$$c_{i+1} = c_i + 3d_i h_i, \quad 1 \le i \le N-1. \tag{1.16}$$

Недостающие условия можно получить, полагая, например, что вторая производная равна нулю на концах участка интерполирования:

$$\psi''(x_0) = 0, \quad c_1 = 0, \tag{1.17}$$

$$\psi''(x_N) = 0, \quad c_N + 3d_N h_N = 0. \tag{1.18}$$

Уравнения (1.13)–(1.18) позволяют определить все $4N$ неизвестных коэффициентов: a_i, b_i, c_i, d_i $(1 \le i \le N)$.

Решение полученной системы уравнений можно сильно упростить, если привести ее к специальному виду.

Используя уравнение (1.13), можно получить все коэффициенты a_i. Из (1.16) и (1.18) следует:

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \qquad 1 \le i \le N-1, \tag{1.19}$$

$$d_N = -\frac{c_N}{3h_N}. \tag{1.20}$$

Из (1.14) и (1.19)

$$b_i = \frac{y_i - y_{i-1}}{h_i} - h_i \frac{c_{i+1} - 2c_i}{3}, \qquad 1 \le i \le N-1. \tag{1.21}$$

Из (1.14) и (1.20)

$$b_N = \frac{y_N - y_{N-1}}{h_N} - h_N \frac{2c_N}{3}. \tag{1.22}$$

Исключим теперь из (1.15) величины b_i и b_{i+1} с учетом (1.21), наращивая во втором случае индекс на единицу, а величину d_i – с

учетом (1.19). В результате получим систему уравнений для определения коэффициентов c_i:

$$c_1 = 0,$$

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right), \quad (1.23)$$

$$2 \le i \le N-1,$$

$$c_{N+1} = 0.$$

После нахождения коэффициентов c_i остальные коэффициенты определяют по следующим формулам:

$$a_i = y_{i-1}, \quad 1 \le i \le N,$$

$$b_i = \frac{y_i - y_{i-1}}{h_i} - h_i\frac{c_{i+1} - 2c_i}{3}, \quad 1 \le i \le N-1,$$

$$b_N = \frac{y_N - y_{N-1}}{h_N} - h_N\frac{2c_N}{3},$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad 1 \le i \le N-1,$$

$$d_N = -\frac{c_N}{3h_N}.$$

Осталось выяснить, как решать систему (1.23). Матрица этой системы трехдиагональна, т. е. все ее элементы равны нулю, кроме тех, которые находятся на главной и двух соседних диагоналях. Такие системы удобно решать методом прогонки. Суть метода в следующем.

Применение метода исключения Гаусса для решения системы уравнений с трехдиагональной матрицей приводит к тому, что система уравнений преобразуется к виду, когда в каждом уравнении содержится только два неизвестных и при обратном ходе одно из этих неизвестных выражается через другое. Поэтому применительно к (1.23) можно записать:

$$c_i = \xi_{i+1}c_{i+1} + \eta_{i+1}, \quad (1.24)$$

где ξ_{i+1}, η_{i+1} – некоторые не известные пока прогоночные коэффициенты;

$$c_{i-1} = \xi_i c_i + \eta_i.$$

Подставляя последнее выражение в (1.23) и преобразуя, получим

$$c_i = -\frac{h_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)} c_{i+1} + \frac{f_i - h_{i-1}\eta_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}. \qquad (1.25)$$

Сравнивая (1.24) и (1.25), получим

$$\xi_{i+1} = -\frac{h_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}, \quad \eta_{i+1} = \frac{f_i - h_{i-1}\eta_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}. \qquad (1.26)$$

В этих формулах введено обозначение

$$f_i = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right).$$

Из условия $c_1 = 0$ следует $\xi_2 = 0$, $\eta_2 = 0$.

Теперь алгоритм решения (1.23) выглядит следующим образом: по формулам (1.26) при известных ξ_2, η_2, равных нулю, вычисляют прогоночные коэффициенты ξ_{i+1}, η_{i+1} $(2 \le i \le N)$ (прямой ход); затем по формулам (1.24) при условии $c_{N+1} = 0$ определяют все c_i (обратный ход).

1.4. Многомерная интерполяция

В различных приложениях широко используют двумерные и трехмерные таблицы. Например, теплофизические свойства различных веществ зависят от температуры и давления, а оптические характеристики – еще и от длины волны излучения.

При многомерной интерполяции из-за громоздкости таблиц необходимо брать достаточно большие шаги по аргументам, т. е. сетка узлов, на которой строят таблицу, получается довольно грубой. Поэтому требуется вводить преобразование переменных

$\eta = \eta(y)$, $\xi = \xi(x)$, $\varphi = \varphi(z)$, подбирая подходящие формулы. При удачном выборе таких формул можно использовать в новых переменных интерполяционный полином невысокой степени.

Осуществляя многомерную интерполяцию, следует помнить, что расположение узлов не может быть произвольным. Например, при интерполяции полиномом первой степени $P_1(x, y)$ узлы не должны лежать на одной прямой в плоскости. Действительно, определитель системы трех уравнений

$$z_i = a + bx_i + cy_i, \quad 1 \le i \le 3, \tag{1.27}$$

записывается в виде

$$\Delta = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2). \tag{1.28}$$

Условие размещения трех точек на одной прямой выглядит следующим образом:

$$\frac{x_2 - x_1}{y_2 - y_1} = \frac{x_3 - x_2}{y_3 - y_2}.$$

После простых преобразований имеем из этого условия

$$x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) = 0, \tag{1.29}$$

т. е. если узлы лежат на одной прямой, то определитель (1.28) обращается в нуль и построить полином $P_1(x, y)$ вида (1.27) невозможно. Проверять условия подобного типа достаточно сложно, поэтому на практике целесообразно строить регулярные сетки, как правило, прямоугольные и равномерные, когда узлы являются точками пересечения двух взаимно перпендикулярных систем параллельных прямых. На этой сетке проводят простую последовательную интерполяцию: сначала по строкам, а затем по столбцам.

При последовательной интерполяции завышается степень интерполяционного полинома. При треугольной конфигурации расположения узлов степень многочлена будет минимальной. Многочлен n-й степени в форме Ньютона в этом случае можно представить как обобщение одномерного варианта записи:

$$P_n(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n-i} z(x_0, \ldots, x_i, y_0, \ldots, y_j) \prod_{p=0}^{i-1} (x - x_p) \prod_{q=0}^{j-1} (y - y_q). \tag{1.30}$$

Пример 1.4. Записать многочлен Ньютона первой и второй степени для двумерной интерполяции функции $z = z(x, y)$.

Из (1.30) получаем

$$P_1(x, y) = z(x_0, y_0) + z(x_0, y_0, y_1)(y - y_0) + z(x_0, x_1, y_0)(x - x_0);$$

$$z(x_0, y_0, y_1) = \frac{z(x_0, y_0) - z(x_0, y_1)}{y_0 - y_1},$$

$$z(x_0, x_1, y_0) = \frac{z(x_0, y_0) - z(x_1, y_0)}{x_0 - x_1};$$

$$P_2(x, y) = z(x_0, y_0) + z(x_0, y_0, y_1)(y - y_0) +$$

$$+ z(x_0, y_0, y_1, y_2)(y - y_0)(y - y_1) + z(x_0, x_1, y_0)(x - x_0) +$$

$$+ z(x_0, x_1, y_0, y_1)(x - x_0)(y - y_0) + z(x_0, x_1, x_2, y_0)(x - x_0)(x - x_1).$$

В ряде случаев приходится использовать нерегулярные сетки. Тогда ограничиваются интерполяционным полиномом первой степени $z = a + bx + cy$, его коэффициенты находят по трем узлам, выбираемым в окрестности точки интерполяции:

$$z_i = a + bx_i + cy_i, \quad 1 \le i \le 3.$$

Коэффициенты a, b, c нет необходимости вычислять, так как первый столбец является линейной комбинацией трех других столбцов. В результате из этих столбцов можно составить определитель, который будет равен нулю. Раскрывая этот определитель по первому столбцу, получим зависимость $z = z(x, y)$.

1.5. Применение математических пакетов для интерполяции

Одномерная интерполяция в Matlab может быть выполнена с использованием функции **interp1**, которая реализует три способа интерполяции: по соседним узлам, когда значение функции в заданной промежуточной точке принимается равным ее значению в ближайшем узле, полиномом первой степени (линейная) и с использованием кубического сплайна. Пример 1.5 иллюстрирует применение функции **interp1** для интерполяции некоторого набора

данных **y** (черные кружки на рис. 1.1), сгенерированных датчиком случайных чисел **rand** и соотнесенных с вектором **x** значений аргумента. На этом же графике показан результат применения нелинейной интерполяции с логарифмическим преобразованием переменных **x** и **y**. Интересно отметить, что, согласно рис. 1.1, в ряде случаев сплайн-интерполяция может дать заметную погрешность на концах области интерполяции (правая граница области на рисунке).

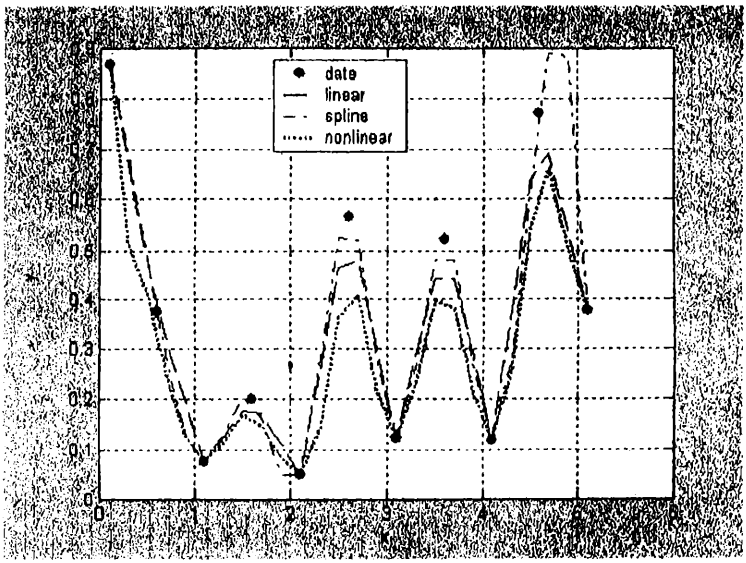

Рис. 1.1. Применение функции **interp1** из пакета Matlab для интерполяции функции одной переменной

Для двумерной интерполяции в Matlab применяется функция **interp2**. Она также обеспечивает три способа интерполяции: по соседним элементам, билинейную интерполяцию и интерполяцию бикубическими сплайнами. Применение данной функции показано в примере 1.6 для двух последних способов интерполяции. Исходные данные сгенерированы с помощью функции

$$z(x) = \sin(3/2\pi x)\cos(3/2\pi y)(1-x^2)(1-y^2).$$

Результаты работы программы показаны на рис. 1.2.

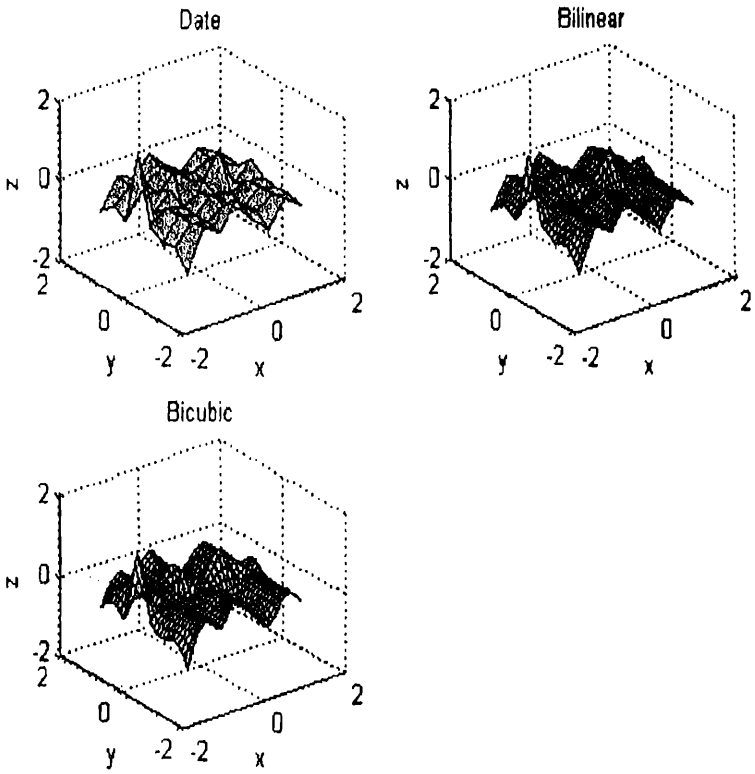

Рис. 1.2. Применение функции **interp2** из пакета Matlab для интерполяции функции двух переменных

Пример 1.5. Интерполяция функции одной переменной в Matlab.

```
%Создание вектора значений аргумента для формирования вектора
%значений интерполируемой функции y
x=[0.1:0.5:5.1];
%Создание вектора значений интерполируемой функции y
y=rand(1,11);
%Вывод графика y(x) (черные кружки)
plot(x,y,'k o ');
```

```
%Формирование вектора значений аргумента, для которых выполняется
%интерполяция
z=[0.1:0.2:5.1];
%Вызов функции interp1 при разных способах интерполяции
y1=interp1(x,y,z,'linear');
y2=interp1(x,y,z,'spline');
%Нелинейная интерполяция в логарифмических переменных
x=log(x);
y=log(y);
hold on;
y3=exp(interp1(x,y,log(z),'linear'));
%Вывод графиков результатов интерполяции
plot(z,y1,'-- k',z,y2,'-. k',z,y3,': k');
grid;
Добавление легенды графика
legend('date','linear','spline','nonlinear');
```

Пример 1.6. Интерполяция функций двух переменных в Matlab.

```
%Вывод графика интерполируемой функции
[x,y]=meshgrid(-pi/2:0.4:pi/2);
z=sin(3/2*pi*x).*cos(3/2*pi*y).*(1-x.^2).*(1-y.^2);
subplot(2,2,1);
surf(x,y,z);
title('Date');
%Создание двумерной сетки значений аргумента для
%проведения процедуры интерполяции
[x1,y1]=meshgrid(-pi/2:0.1:pi/2);
%Вызов функции interp2
zbil=interp2(x,y,z,x1,y1,'bilinear');
%Вывод графиков результатов интерполяции
subplot(2,2,2);
surf(x1,y1,zbil);
title('Bilinear');
zbic=interp2(x,y,z,x1,y1,'bicubic');
subplot(2,2,3);
surf(x1,y1,zbic);
title('Bicubic');
```

Интерполяция функций одной переменной в Mathcad выполняется с помощью функций **linterp** (полиномом первой степени) и **interp** (кубическими сплайнами). Использование этих функций по-

казано в примере 1.7, а результаты представлены на рис. 1.3. Применение функции **interp** требует предварительного обращения к функциям **lspline, pspline, cspline,** которые возвращают вектор вторых производных для трех типов условий на концах интервала интерполяции: линейные, параболические и кубические.

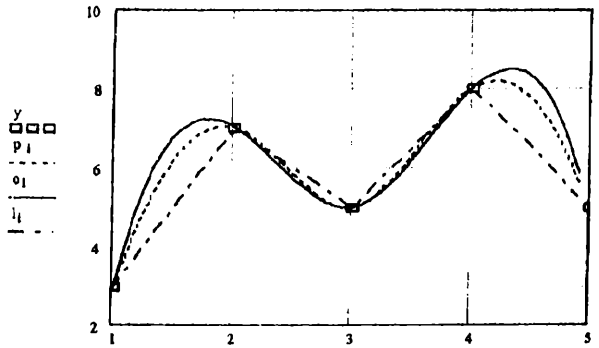

Рис. 1.3. Применение функций **linterp** и **interp** из пакета Mathcad для интерполяции фукции одной переменной

Пример 1.7. Интерполяция функций одной переменной в Mathcad.

$$\text{data} := \begin{pmatrix} 1 & 3 \\ 2 & 7 \\ 3 & 5 \\ 4 & 8 \\ 5 & 5 \end{pmatrix} \qquad x := \text{data}\langle 0 \rangle \qquad y := \text{data}\langle 1 \rangle$$

$$\text{fint}(t) := \text{linterp}(x, y, t)$$

$$s2 := \text{pspline}(x, y) \qquad \text{fp}(t) := \text{interp}(s2, x, y, t)$$

$$s3 := \text{cspline}(x, y) \qquad \text{fc}(t) := \text{interp}(s3, x, y, t)$$

$$i := 1 \ldots 50$$

$$t_i := \frac{4 \times (i-1)}{50} + 1 \qquad l_i := \text{fint}(t_i)$$

$$p_i := \text{fp}(t_i)$$

$$c_i := \text{fc}(t_i)$$

2. СРЕДНЕКВАДРАТИЧНОЕ ПРИБЛИЖЕНИЕ

При интерполировании функции строят некоторую новую функцию, совпадающую с заданной в фиксированных узлах. В ряде случаев целесообразно приближать функции не по точкам, а в среднем, например, когда необходимо иметь аналитическую зависимость для широкого диапазона значений аргумента, а не только в окрестности некоторого узла, или когда значения функции в узлах определены неточно.

Пусть имеется множество функций $\varphi(x)$, принадлежащих линейному пространству функций. Под близостью в среднем интерполируемой и интерполирующей функций будем понимать результат оценки интеграла

$$I = \left\{ \int_a^b \rho(x)[y(x) - \varphi(x)]^2 dx \right\}^{\frac{1}{2}}, \qquad (2.1)$$

где $\rho(x)$ – весовая функция.

Такое приближение называют среднеквадратичным. Можно рассмотреть две задачи:

1) подобрать функцию $\varphi(x)$ так, чтобы выполнялось неравенство

$$I \le \varepsilon; \qquad (2.2)$$

2) найти наилучшее приближение, т. е. такую функцию $\bar{\varphi}(x)$, чтобы было справедливым соотношение

$$\int_a^b \rho(x)[y(x) - \bar{\varphi}(x)]^2 dx = \inf \int_a^b \rho(x)[y(x) - \varphi(x)]^2 dx. \qquad (2.3)$$

Займемся поиском наилучшего приближения.

Разложим функцию $\varphi(x)$ по системе линейно независимых функций $\varphi_k(x)$:

$$\varphi(x) = \sum_{k=1}^{n} a_k \varphi_k(x). \tag{2.4}$$

В дальнейшем для сокращения записи будем пользоваться определением скалярного произведения в пространстве функций L_2:

$$(f, \varphi) = \int_a^b \rho(x) f(x) \varphi(x) dx.$$

Подставляя (2.4) в условие (2.3), получим

$$((y-\varphi),(y-\varphi)) = (y, y) - 2\sum_{k=1}^{n} a_k (y, \varphi_k) +$$

$$+ \sum_{k=1}^{n} \sum_{m=1}^{n} a_k a_m (\varphi_k, \varphi_m) = \min.$$

Дифференцируя это выражение по a_k и приравнивая производные нулю, найдем

$$\sum_{m=1}^{n} (\varphi_k, \varphi_m) a_m = (y, \varphi_k), \; 1 \le k \le n. \tag{2.5}$$

Определитель этой системы есть определитель Грама функций $\varphi_k(x)$. В силу их линейной независимости этот определитель не равен нулю. Следовательно, из системы (2.5) можно найти коэффициенты a_k, определяющие функцию $\varphi(x)$, согласно (2.4), и минимизирующие интеграл (2.3) от погрешности $|y(x) - \varphi(x)|$. Таким образом, наилучшее среднеквадратичное приближение существует, и оно единственно.

При использовании ортонормированной системы функций $\varphi_k(x)$ система (2.5) упрощается: $a_k = (y, \varphi_k) = \int_a^b \rho(x) y(x) \varphi_k(x) dx$, т. е. a_k являются коэффициентами Фурье, а наилучшее приближение есть ряд Фурье, обрываемый на каком-то члене.

В тех случаях, когда функции $\varphi_k(x)$ не ортогональны, при $n \to \infty$ определитель Грама уменьшается, приближаясь к нулю. Тогда система (2.5) становится плохо обусловленной, и ее решение дает большую погрешность. В этой ситуации обычно берут не более пяти-шести членов в сумме (2.4).

В качестве $\varphi_k(x)$ чаще всего используют полиномы Лежандра, Чебышева, Лагерра, Эрмита, ортогональные с заданным весом.

Рассмотрим частный случай, когда необходимо найти наилучшее приближение функции, заданной таблично. Для вещественных функций, заданных на конечном множестве точек, скалярное произведение определяется формулой

$$(f, \varphi) = \sum_{i=1}^{N} \rho_i f(x_i) \varphi(x_i), \quad \rho_i > 0, \tag{2.6}$$

где N – число заданных узлов.

Условие наилучшего среднеквадратичного приближения записывается следующим образом:

$$\sum_{i=1}^{N} \rho_i [y(x_i) - \varphi(x_i)]^2 = \min. \tag{2.7}$$

Полагая $\varphi(x) = \sum_{k=1}^{n} a_k \varphi_k(x)$, где $n < N$, и подставляя этот многочлен в (2.7), придем к системе (2.5), в которой скалярные произведения вычисляют согласно (2.6). Описанная процедура аппроксимации носит название метода наименьших квадратов.

Наиболее употребительный вариант метода наименьших квадратов соответствует случаю степенного вида функций $\varphi_k(x)$, т. е. $\varphi_k(x) = x^k$, причем $0 \le k \le n$.

Система уравнений (2.5) при этом принимает вид

$$\sum_{m=0}^{n} (x^k,\ x^m) a_m = (y,\ x^k),\quad 0 \le k \le n, \tag{2.8}$$

где $(x^k,\ x^m) = \sum_{i=1}^{N} \rho_i x_i^{k+m}$; $(y,\ x^k) = \sum_{i=1}^{N} \rho_i y_i x_i^k$.

Пример 2.1. Методом наименьших квадратов аппроксимировать сеточную функцию линейной зависимостью вида $\varphi(x) = a_0 + a_1 x$.

В данном случае $n = 1$. Тогда система уравнений (2.8) имеет вид

$$(x^0,\ x^0) a_0 + (x^0,\ x^1) a_1 = (y,\ x^0),$$

$$(x^1,\ x^0) a_0 + (x^1,\ x^1) a_1 = (y,\ x^1).$$

Скалярные произведения в полученной системе записываются следующим образом:

$$(x^0,\ x^0) = \sum_{i=1}^{N} \rho_i,\quad (x^1,\ x^0) = \sum_{i=1}^{N} \rho_i x_i,\quad (x^1,\ x^1) = \sum_{i=1}^{N} \rho_i x_i^2,$$

$$(y,\ x^0) = \sum_{i=1}^{N} \rho_i y_i,\quad (y,\ x^1) = \sum_{i=1}^{N} \rho_i y_i x_i.$$

Окончательно

$$a_0 = \frac{\sum_{i=1}^{N} \rho_i y_i \sum_{i=1}^{N} \rho_i x_i^2 - \sum_{i=1}^{N} \rho_i x_i \sum_{i=1}^{N} \rho_i x_i y_i}{\sum_{i=1}^{N} \rho_i \sum_{i=1}^{N} \rho_i x_i^2 - \left(\sum_{i=1}^{N} \rho_i x_i\right)^2},$$

$$a_1 = \frac{\sum_{i=1}^{N} \rho_i \sum_{i=1}^{N} \rho_i x_i y_i - \sum_{i=1}^{N} \rho_i x_i \sum_{i=1}^{N} \rho_i y_i}{\sum_{i=1}^{N} \rho_i \sum_{i=1}^{N} \rho_i x_i^2 - \left(\sum_{i=1}^{N} \rho_i x_i\right)^2}.$$

Система функций x^k не ортогональна, поэтому при больших n задача (2.8) плохо обусловлена, в связи с чем на практике ограничиваются значениями $n \le 5$.

Применение математических пакетов для аппроксимации функций

Среднеквадратичное приближение в Matlab может быть выполнено с использованием функции **polyfit**. В примере 2.2 и на рис. 2.1 показан результат применения **polyfit** для аппроксимации данных (черные точки), заданных в нескольких точках с помощью функции

$$f(x) = x^2 \cos(2\pi / 5x).$$

Аппроксимация проводится четырьмя полиномами разных степеней: первой, второй, четвертой и шестой. На рис. 2.1 видно, как меняется характер приближения по мере увеличения степени полинома.

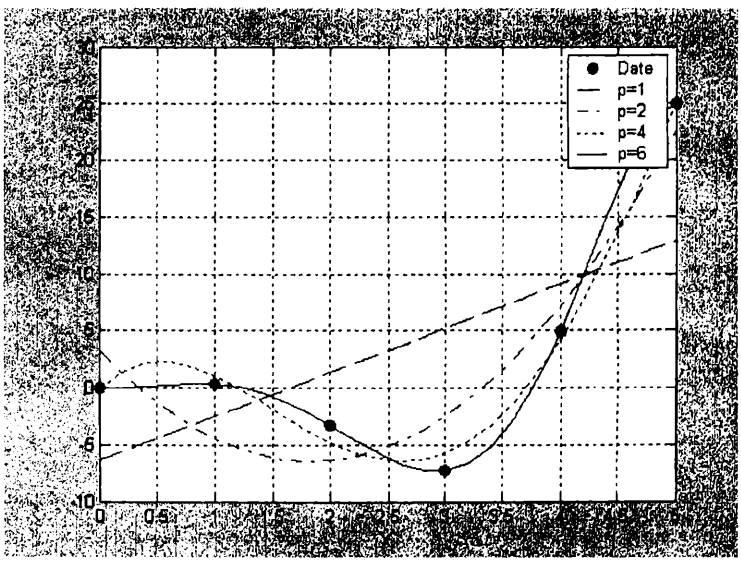

Рис. 2.1. Применение функции **polyfit** для среднеквадратичного приближения функции

Пример 2.2. Среднеквадратичное приближение функций одной переменной в Matlab.

```
x=[0:5];
y=x.^2.*cos(2*pi/5.*x);
pol_1=polyfit(x,y,1);
pol_2=polyfit(x,y,2);
pol_4=polyfit(x,y,4);
pol_6=polyfit(x,y,6);
z=[0:0.2:5];
pp_1=polyval(pol_1,z);
pp_2=polyval(pol_2,z);
pp_4=polyval(pol_4,z);
pp_6=polyval(pol_6,z);
plot(x,y,'ko',z,pp_1,'--k',z,pp_2,'-.k',z,pp_4,': ',z,pp_6,'- k');
legend('Date','p=1','p=2','p=4','p=6');
```

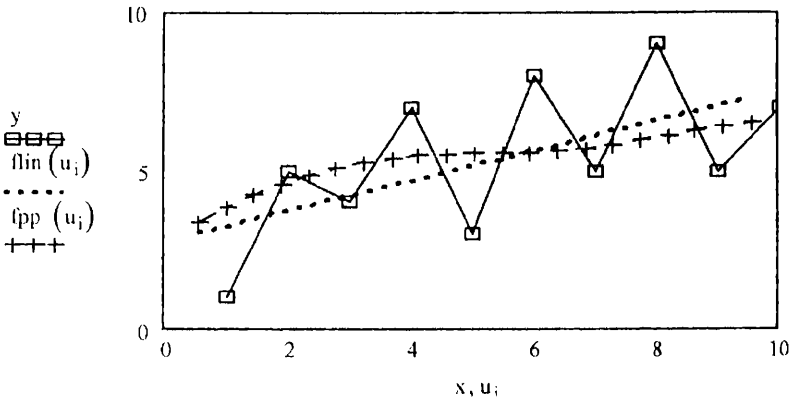

Рис. 2.2. Применение функций Mathcad для среднеквадратичного приближения функции

Среднеквадратичное приближение в Mathcad может быть осуществлено несколькими способами. Аппроксимация полиномом первой степени $\varphi(x) = ax + b$, при этом для определения коэффициентов b и a используются соответственно функции Mathcad **intercept** и **slope**. Приближение линейной комбинацией произвольных функций – **linfit**. Полиномиальная аппроксимация полиномом произвольной степени – функция **interp**. Нелинейная ап-

проксимация произвольной функцией $f(x,\ k_1,\ k_2,\ k_3...)$ от параметров k_1, k_2, k_3 ... – **genfit**. В примере 2.3 показано применение функций **intercept**, **slope** и **linfit** для среднеквадратичного приближения заданной таблицы, а на рис. 2.2 – соответствующие графики результатов аппроксимации.

Пример 2.3. Среднеквадратичное приближение функций одной переменной в Mathcad.

Задание таблицы исходных данных и векторов аргумента и интерполируемой функции

$$\text{data} := \begin{pmatrix} 1 & 1 \\ 2 & 5 \\ 3 & 4 \\ 4 & 7 \\ 5 & 3 \\ 6 & 8 \\ 7 & 5 \\ 8 & 9 \\ 9 & 5 \\ 10 & 7 \end{pmatrix} \quad x := \text{data}^{\langle 0 \rangle} \quad y := \text{data}^{\langle 1 \rangle}$$

Аппроксимация полиномом первой степени

$$a := \text{slope}(x, y) \quad b := \text{intercept}(x, y)$$
$$\text{flin}(x) := a \cdot x + b$$

Использование линейной комбинации функций

$$F(t) := \begin{pmatrix} \sin(t) \\ t^{0.2} \\ \cos(t) \end{pmatrix} \qquad A := \text{linfit}(x, y, F) \qquad A = \begin{pmatrix} -0.064 \\ 4.014 \\ -0.196 \end{pmatrix}$$

$$i := 0..20 \qquad u_i := 1 + (i - 1) \cdot \frac{9}{20}$$

$$fp(t) := F(t) \cdot A \qquad fp(4) = 5.474 \qquad fp(9) = 6.382$$

$$fpp(x) := A_0 \cdot \sin(x) + A_1 \cdot x^{0.2} + A_2 \cdot \cos(x)$$

3. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ ФУНКЦИЙ ОДНОЙ ПЕРЕМЕННОЙ

Во многих случаях применение формулы Ньютона–Лейбница для вычисления определенного интеграла затруднительно или невозможно из-за того, что первообразная оказывается слишком сложной или вообще не может быть выражена через элементарные функции. Кроме того, подынтегральная функция в задачах моделирования часто задается в табличной форме, и тогда само понятие первообразной теряет смысл. В этой ситуации проблема разрешается с использованием численных методов. Вначале рассмотрим интегрирование функций с использованием формул Гаусса, обладающих наивысшей алгебраической точностью. Затем получим другие часто используемые в вычислительной практике формулы (трапеций, средних, Симпсона).

3.1. Квадратурная формула Гаусса

Пусть функция задана на стандартном интервале [–1; 1]. Задача состоит в том, чтобы подобрать точки $t_1, t_2, ..., t_n$ и коэффициенты $A_1, A_2, ..., A_n$ так, чтобы квадратурная формула

$$\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} A_i f(t_i) \qquad (3.1)$$

была точной для всех полиномов наивысшей возможной степени. Из приведенного ниже способа нахождения узлов t_i и коэффициентов A_i следует, что эта наивысшая степень равна $N = 2n - 1$.

Запишем полином в виде $f(t) = \sum_{k=0}^{2n-1} a_k t^k$. Легко показать, что для того, чтобы формула (3.1) была точна для полинома, необхо-

димо и достаточно, чтобы она была точна для степенных функций t^k при $k = 0, 1, 2, ..., 2n-1$. Действительно, полагая, что

$$\int_{-1}^{1} t^k dt = \sum_{i=1}^{n} A_i t_i^k, \qquad k = 0,\ 1,\ 2,\ ...,\ 2n-1, \tag{3.2}$$

получим

$$\int_{-1}^{1} f(t)dt = \int_{-1}^{1} \sum_{k=0}^{2n-1} a_k t^k dt = \sum_{k=0}^{2n-1} a_k \int_{-1}^{1} t^k dt =$$

$$= \sum_{k=0}^{2n-1} a_k \sum_{i=1}^{n} A_i t_i^k = \sum_{i=1}^{n} A_i \sum_{k=0}^{2n-1} a_k t_i^k = \sum_{i=1}^{n} A_i f(t_i).$$

Таким образом, система (3.2) дает $2n$ соотношений для определения $2n$ неизвестных A_i и t_i. При этом

$$\int_{-1}^{1} t^k dt = \frac{1-(-1)^{k+1}}{k+1} = \begin{cases} \dfrac{2}{k+1} & \text{при } k \text{ четном;} \\ 0 & \text{при } k \text{ нечетном.} \end{cases}$$

Итак, A_i и t_i находят из системы $2n$ уравнений:

$$\begin{aligned} &\sum_{i=1}^{n} A_i = 2, \\ &\sum_{i=1}^{n} A_i t_i = 0, \\ &\sum_{i=1}^{n} A_i t_i^2 = \frac{2}{3}, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \\ &\sum_{i=1}^{n} A_i t_i^{2n-1} = 0. \end{aligned} \tag{3.3}$$

Система (3.3) нелинейная, и ее решение найти довольно трудно. Рассмотрим следующий прием нахождения A_i и t_i. Для этого нам понадобятся полиномы Лежандра

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n], \quad n = 0,\ 1,\ 2, \ldots$$

Эти полиномы обладают рядом полезных свойств.

1. $P_n(1) = 1$, $P_n(-1) = (-1)^n$, $\quad n = 0,\ 1,\ 2, \ldots$

2. $\int\limits_{-1}^{1} P_n(x)P_m(x)dx = \delta_{mn}N_m$, $\quad N_m = \dfrac{2}{2n+1}$.

3. Полином Лежандра $P_n(x)$ имеет n различных и действительных корней, расположенных на интервале $[-1; 1]$.

Составим по узлам интегрирования многочлен n-й степени

$$w_n(x) = \prod_{k=1}^{n}(x - x_k).$$

Функция $f(x) = w_n(x)P_m(x)$ при $m \le n-1$ есть многочлен степени не выше $2n-1$. Значит, для этой функции формула Гаусса справедлива:

$$\int\limits_{-1}^{1} w_n(x)\,P_m(x)\,dx = \sum_{i=1}^{n} A_i w_n(x_i)P_m(x_i) = 0,$$

так как $w_n(x_i) = 0$.

Разложим $w_n(x)$ в ряд по ортогональным многочленам Лежандра:

$$w_n(x) = \sum_{k=0}^{n} b_k P_k(x),$$

$$\int\limits_{-1}^{1} w_n(x)\,P_m(x)\,dx = \int\limits_{-1}^{1}\Big[\sum_{k=0}^{n} b_k P_k(x)\Big]P_m(x) = b_m N_m = 0,$$

$$m \le n-1,$$

т. е. все коэффициенты $b_m = 0$ при $m \le n-1$. Значит $w_n(x)$ с точностью до численного множителя совпадает с $P_n(x)$. Таким образом, узлами формулы Гаусса являются нули многочлена Лежандра степени n.

Зная значение t_i, из линейной теперь системы первых n (3.3) легко найти коэффициенты A_i ($i = 1, 2, ..., n$). Определитель этой системы есть определитель Вандермонда.

Формулу $\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} A_i f(t_1)$, в которой t_i – нули полинома Лежандра $P_n(t)$, а A_i определяют из (3.3), называют квадратурной формулой Гаусса.

Пример 3.1. Вывести квадратурную формулу Гаусса для случая трех узлов ($n = 3$).

Полином Лежандра третьей степени

$$P_3(t) = \frac{1}{2}(5t^3 - 3t).$$

Корни полинома

$$t_1 = -\sqrt{\frac{3}{5}}, \quad t_2 = 0, \quad t_3 = \sqrt{\frac{3}{5}}.$$

Из (3.3) имеем

$$A_1 + A_2 + A_3 = 2;$$

$$-\sqrt{\frac{3}{5}}A_1 + \sqrt{\frac{3}{5}}A_3 = 0;$$

$$\frac{3}{5}A_1 + \frac{3}{5}A_3 = \frac{2}{3}.$$

Отсюда

$$A_1 = A_3 = \frac{5}{9}, \quad A_2 = \frac{8}{9}.$$

Тогда

$$\int_{-1}^{1} f(t)dt = \frac{1}{9}[5f(-\sqrt{3/5}) + 8f(0) + 5f(\sqrt{3/5})].$$

При вычислении интеграла с неединичными пределами $\int_a^b f(x)dx$ д
применения квадратурной формулы Гаусса необходимо выполнить пр
образование переменной:

$$x = \frac{b+a}{2} + \frac{b-a}{2}t.$$

Получим

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b+a}{2} + \frac{b-a}{2}t)dt;$$

$$\int_a^b f(x)dx = \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i),$$

где

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}t_i, \quad i = 1, 2, \ldots, n,$$

а t_i – нули полинома Лежандра $P_n(t)$, т. е. $P_n(t_i) = 0$.

Остаточный член формулы Гаусса с n узлами выражается формулой

$$R_n = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi).$$

Отсюда, в частности, следует

$$R_2 = \frac{1}{135}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi),$$

$$R_3 = \frac{1}{15750}\left(\frac{b-a}{2}\right)^7 f^{(6)}(\xi),$$

$$R_4 = \frac{1}{3472875}\left(\frac{b-a}{2}\right)^9 f^{(8)}(\xi),$$

$$R_8 = \frac{1}{648984486150}\left(\frac{b-a}{2}\right)^{13} f^{(12)}(\xi).$$

3.2. Другие формулы численного интегрирования

Ставится задача вычисления определенного интеграла

$$F = \int_a^b f(x)dx. \tag{3.4}$$

При построении нижеприведенных формул численного интегрирования используется общая идея замены подынтегральной функции $f(x)$ на интерполяционный многочлен, который легко интегрируется. Поскольку коэффициенты полинома линейным образом выражаются через значения интегрируемой функции в узлах, то имеет место соотношение

$$f(x) \approx \sum_{i=1}^{n} f(x_i)\varphi_i(x) + r(x), \tag{3.5}$$

где n – количество узлов интерполяции на отрезке интегрирования $[a, b]$; $\varphi_i(x)$ – многочлены степени n; x_i – заданные узлы интерполяции; $r(x)$ – остаточный член.

Подставляя (3.5) в (3.4), получают формулу численного интегрирования, (она называется квадратурной формулой)

$$\begin{aligned} F &= \sum_{i=0}^{n} A_i f(x_i) + R, \\ A_i &= \int_a^b \varphi_i(x)dx, \qquad R = \int_a^b r(x)dx, \end{aligned} \tag{3.6}$$

где A_i – вес квадратурной формулы, а R – погрешность или остаточный член формулы.

Понятно, что формула (3.6) точна для полинома степени n, а следовательно, и для всех степеней x от 0 до n, т. е. справедливы соотношения, использованные ранее при получении системы (3.3):

$$\int_a^b x^k dx = \sum_{i=0}^{n} A_i x_i^k \quad \text{при} \quad k = 0,\ 1,\ 2,\ ....,\ n. \tag{3.7}$$

При этом

$$\int_a^b x^k dx = \frac{b^{k+1} - a^{k+1}}{k+1}.$$

Отсюда, задаваясь количеством и расположением узлов, можно, применяя (3.7), вычислить веса квадратурной формулы.

Например, при $n = 0$ и $x_0 = \frac{a+b}{2}$ получим $b - a = A_0$, т. е. квадратурная формула имеет вид

$$\int_a^b f(x)dx = (b-a) f\left(\frac{a+b}{2}\right). \tag{3.8}$$

Эта формула называется формулой средних.

При $n = 1$, $x_0 = a$ и $x_1 = b$ получаем следующую систему уравнений:

$$b - a = A_0 + A_1,$$
$$\frac{b^2 - a^2}{2} = A_0 a + A_1 b.$$

Откуда $A_0 = A_1 = \frac{b-a}{2}$ и квадратурная формула имеет вид

$$\int_a^b f(x)dx = \frac{(b-a)}{2}\left[f(a) + f(b)\right]. \tag{3.9}$$

Формула (3.9) называется формулой трапеций.

Точно так же может быть получена формула Симпсона

$$\int_a^b f(x)dx = \frac{(b-a)}{6}\left[f(x_0)+4f(x_1)+f(x_2)\right], \qquad (3.10)$$

для которой

$$n=2,\ x_0=a,\ x_1=\frac{a+b}{2},\ x_2=b.$$

Эту процедуру можно продолжить, получая новые формулы со все большим количеством узлов. Однако можно данную процедуру проделать иначе. Используя интерполяционный полином Лагранжа, т. е. подставляя в (3.5) в качестве функций $\varphi_i(x)$ лагранжевы коэффициенты $L_i^{(n)}(x)$ и проведя интегрирование в формуле (3.6), вычислим A_i:

$$A_i=(b-a)H_i,$$

где H_i – коэффициенты Котеса;

$$H_i=\frac{(-1)^{n-i}}{ni!\ (n-i)!}\int_0^n \frac{q^{[n+1]}}{q-i}\,dq, \qquad (3.11)$$

$$q^{[n+1]}=q(q-1)...(q-n).$$

Квадратурная формула в общем виде запишется следующим образом:

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{n} H_i f(x_i). \qquad (3.12)$$

При этом $x_i=a+ih,\ h=\frac{b-a}{n}$ и справедливы соотношения $\sum_{i=0}^{n} H_i=1,\ H_i=H_{n-1}$. Получим, например, с помощью (3.11) и (3.12) выведенную выше формулу Симпсона ($n=2$)

$$H_0=\frac{1}{2\cdot 2}\int_0^2 \frac{q(q-1)(q-2)}{q}dq=\frac{1}{4}\int_0^2 (q^2-3q+2)dq=\frac{1}{6},$$

$$H_1 = -\frac{1}{2 \cdot 1}\int_0^2 q(q-2)\,dq = -\frac{1}{2}\int_0^2 (q^2 - 2q)\,dq = \frac{2}{3},$$

$$H_2 = \frac{1}{2 \cdot 2}\int_0^2 q(q-1)\,dq = \frac{1}{4}\int_0^2 (q^2 - q)\,dq = \frac{1}{6}.$$

Подставляя коэффициенты Котеса в (3.12), получим (3.10). Существуют таблицы коэффициентов Котеса вплоть до значения $n = 10$.

Приведем сводку ряда простейших формул, используемых в практике численного интегрирования, при постоянном расстоянии между узлами (шаге сетки).

Формула трапеций имеет вид

$$\int_a^b f(x)\,dx \approx h\left(\frac{1}{2} f_0 + f_1 + \ldots + f_{N-1} + \frac{1}{2} f_N\right)$$

с погрешностью

$$R \approx -\frac{1}{12} h^2 \int_a^b f''(x)\,dx = O(h^2), \qquad (3.13)$$

$$h = x_i - x_{i-1} = \text{const},$$

где R – асимптотическая погрешность формулы; f_i – значения интегрируемой функции в узлах.

Формула средних имеет вид

$$\int_a^b f(x)\,dx \approx h \sum_{i=1}^{N} f_{i-\frac{1}{2}}$$

с погрешностью

$$R \approx \frac{1}{24} h^2 \int_a^b f''(x)\,dx = O(h^2). \qquad (3.14)$$

Формула Симпсона имеет вид

$$\int_a^b f(x)dx \approx \frac{h}{3}\sum_{i=0}^{\frac{N}{2}-1}(f_{2i} + 4f_{2i+1} + f_{2i+2})$$

с погрешностью

$$R \approx -\frac{1}{180}h^4\int_a^b f^{(4)}(x)dx = O(h^4). \qquad (3.15)$$

3.3. Применение математических пакетов для вычисления интегралов

Численное интегрирование функций одной переменной в Matlab выполняется с использованием функций **quad** и **quadl**. Первая основана на методе Симпсона с автоматическим выбором шага, вторая – на формулах Гаусса – Лобатто. Для использования этих функций необходимо предварительно создать файл – функцию, определяющую подынтегральное выражение. Точностью вычислений можно управлять, указывая ее среди параметров функций **quad** и **quadl**. По умолчанию точность равна 10^{-3}. В примере 3.2 показано, как вычисляется определенный интеграл

$$I = \int_{-1}^{1} e^{-x^2}\cos(x)\,dx$$

от функции, формируемой с помощью файл-функции **ff**.

Пример 3.2. Интегрирование функции одной переменной в Matlab.

```
int_s=quad('ff',-1,1,1e-5);
int_g=quadl('ff',-1,1,1e-5);
function f=ff(x);
f=exp(-x.^2).*cos(x);
```

В Mathcad вычисление определенных интегралов проводится путем записи выражений, максимально соответствующих обычной математической записи с использованием знака интеграла, выби-

раемого из палитры математических символов или нажатием клавиши &. Точность вычислений может изменяться путем задания значения переменной TOL. В примере 3.3 и на рис. 3.1 приведено решение следующей задачи: построить график зависимости от параметра t интеграла

$$f(t) = \int_0^3 \exp[-(t^2 + x^2)] \cdot \sin(6t)dx.$$

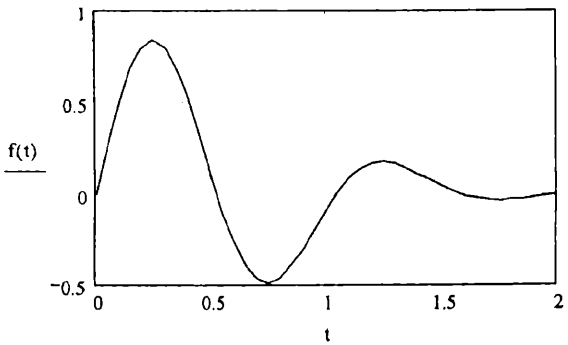

Рис. 3.1. Вычисление определенного интеграла в Mathcad

Пример 3.3. Интегрирование функции одной переменной в Mathcad.

$$f(t) := \int_0^3 \exp[-(t^2 + x^2)] \cdot \sin(6 \cdot t)dx$$

t:=0,0.52...2

4. КРАТНЫЕ ИНТЕГРАЛЫ

4.1. Метод ячеек

Рассмотрим двукратный интеграл по прямоугольной области

$$\int_c^d \int_a^b f(x,y)dx\,dy \approx S\, f(\overline{x},\overline{y}), \qquad (4.1)$$

где $S=(b-a)(d-c)$; $\overline{x}=\dfrac{a+b}{2}$; $\overline{y}=\dfrac{c+d}{2}$.

Для повышения точности разобьем область на прямоугольные ячейки:

$$\int_c^d \int_a^b f(x,y)dx\,dy \approx \sum_i S_i f(\overline{x}_i,\overline{y}_i). \qquad (4.2)$$

Оценим погрешность интегрирования. Разложим подынтегральную функцию в ряд Тейлора:

$$f(x,y)=f(\overline{x},\overline{y})+(x-\overline{x})f'_x+(y-\overline{y})f'_y+\frac{1}{2}(x-\overline{x})^2 f''_{xx}+$$

$$+\frac{1}{2}(x-\overline{x})(y-\overline{y})f''_{xy}+\frac{1}{2}(y-\overline{y})^2 f''_{yy}+\ldots,$$

где все производные берутся в центре ячейки. Для формулы (4.1) погрешность

$$R=\int_c^d \int_a^b f(x,y)dx\,dy - S\, f(\overline{x},\,\overline{y}) \approx \frac{1}{24}S[(\,b-a)^2 f''_{xx}+(d-c)^2 f''_{yy}].$$

Пусть теперь в обобщенной квадратурной формуле (4.2) стороны прямоугольника разбиты соответственно на N и M равных частей. Тогда погрешность этой формулы для единичной ячейки

$$R_i \approx \frac{1}{24} S_i [(\frac{b-a}{N})^2 f''_{xx} + (\frac{d-c}{M})^2 f''_{yy}].$$

Суммируя погрешность по всем ячейкам, получим суммарную погрешность

$$R_\Sigma = \frac{1}{24} [(\frac{b-a}{N})^2 \iint\limits_G f''_{xx}\, dx\, dy + (\frac{d-c}{M})^2 \iint\limits_G f''_{yy}\, dx\, dy] = O(N^{-2} + M^{-2}),$$

т. е. формула имеет второй порядок точности.

Обобщим формулу ячеек на более сложные области. Используем (4.1), где под S понимается площадь области, а $\overline{x}$, $\overline{y}$ рассматриваются как координаты центра тяжести сложной области:

$$\overline{x} = \frac{1}{S} \iint\limits_G x\, dx\, dy, \quad \overline{y} = \frac{1}{S} \iint\limits_G y\, dx\, dy.$$

Практическую ценность этот подход имеет только для областей простой формы (треугольник, правильный многоугольник, трапеция), центр тяжести и площадь которых легко определить. Понятно, что данный подход годится и для области, ограниченной ломаной линией, так как ее всегда можно разбить на прямоугольники и треугольники.

Если граница области G криволинейная, то формулу (4.2) применяют, накладывая на область G прямоугольную сетку. Все ячейки разделяют на внутренние и граничные. Площадь внутренней ячейки равна произведению ее сторон. Площадь граничной ячейки вычисляют приближенно, заменяя в пределах данной ячейки истинную границу области хордой. Значения этих площадей подставляют в (4.2).

Погрешность формулы (4.2) при этом будет такой. В каждой внутренней ячейке ошибка составляет $O(N^{-2})$, в каждой граничной ячейке относительная ошибка есть $O(N^{-1})$, так как центр тяжести прямоугольной ячейки не совпадает с центром тяжести входящей в интеграл части. Но граничных ячеек примерно в N раз меньше, чем внутренних. Поэтому общая погрешность будет $O(N^{-2})$, т. е. имеем второй порядок точности.

Можно граничные ячейки вообще не включать в сумму. Погрешность при этом составит $O(N^{-1})$.

4.2. Последовательное интегрирование

Рассмотрим интеграл по прямоугольной области, разбитой сеткой на ячейки:

$$I = \int_c^d \int_a^b f(x,\ y)dx\,dy = \int_a^b F(x)dx,$$

где

$$F(x) = \int_c^d f(x, y)dy.$$

Каждый однократный интеграл вычисляют на данной сетке по квадратурным формулам

$$I = \sum_i A_i F(x_i), \quad \text{где} \quad F(x_i) = \sum_j \overline{A}_j f(x_i,\ y_j).$$

Тогда

$$I = \sum_i \sum_j A_i \overline{A}_j f(x_i,\ y_j) = \sum_i \sum_j g_{ij} f(x_i,\ y_j).$$

Для разных направлений можно использовать квадратурные формулы разных порядков точности (трапеций, прямоугольников, средних, Симпсона и т. д.).

Можно использовать формулы Гаусса, тогда

$$g_{ij} = \frac{1}{4}(b-a)(d-c)\gamma_i\gamma_j,$$

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\xi_i; \qquad y_j = \frac{c+d}{2} + \frac{d-c}{2}\xi_j, \qquad 1 \le i;\ j \le n,$$

где $\xi_i, \xi_j, \gamma_i, \gamma_j$ – нули многочленов Лежандра и веса формулы Гаусса.

Теперь пусть область интегрирования ограничена непрерывными однозначными кривыми

$$y = \varphi(x),\ \ y = \psi(x)\ \ (\varphi(x) < \psi(x))$$

и двумя вертикалями $x = a$ и $x = b$.

Имеем

$$I = \iint\limits_G f(x,y)\,dx\,dy = \int\limits_a^b dx \int\limits_{\varphi(x)}^{\psi(x)} f(x,y)\,dy = \int\limits_a^b F(x)\,dx,$$

где

$$F(x) = \int\limits_{\varphi(x)}^{\psi(x)} f(x,\ y)\,dy.$$

Отсюда

$$I = \sum_{i=1}^{n} A_i F(x_i), \quad F(x_i) = \int\limits_{\varphi(x_i)}^{\psi(x_i)} f(x_i,\ y)dy \approx \sum_{j=1}^{m_i} B_{ij} f(x_i,\ y_j),$$

$$I = \iint\limits_G f(x,\ y)\ dx\,dy = \sum_{i=1}^{n}\sum_{j=1}^{m_i} A_i B_{ij} f(x_i, y_j),$$

где A_i, B_{ij} – известные постоянные.

Пример 4.1. Получить кубатурную формулу типа формулы Симпсона.

Пусть сначала областью интегрирования будет прямоугольник. Разобьем каждую сторону данного прямоугольника на два интервала, т. е. введем на каждой границе области сетку с тремя узлами. Соответствующие шаги сетки

$$h = \frac{b-a}{2}, \quad k = \frac{d-c}{2}.$$

Проведем цепочку преобразований:

$$\iint\limits_G f(x,y)dx\,dy = \int\limits_a^b dx \int\limits_c^d f(x,y)\,dy =$$

$$= \frac{k}{3}\left[\int\limits_a^b f(x,\ y_0)dx + 4\int\limits_a^b f(x,\ y_1)dx + \int\limits_a^b f(x,\ y_2)dx\right].$$

Вновь применим к каждому интегралу формулу Симпсона:

$$\iint\limits_G f(x,y)dxdy = \frac{kh}{9}\{[f(x_0,y_0) + 4f(x_1,y_0) + f(x_2,y_0)] +$$

$$+4[f(x_0,y_1)+4f(x_1,y_1)+f(x_2,y_1)]+$$

$$+[f(x_0,y_2)+4f(x_1,y_2)+f(x_2,y_2)]\}$$

или

$$\iint_G f(x,y)\,dx\,dy=\frac{kh}{9}\{f(x_0,y_0)+f(x_2,y_0)+f(x_0,y_2)+f(x_2,y_2)+$$

$$+4[f(x_1,y_0)+f(x_0,y_1)+f(x_2,y_1)+f(x_1,y_2)]+16f(x_1,y_1)\}.$$

Таким образом, получим кубатурную формулу Симпсона.

Для повышения точности интегрирования вводят более мелкую сетку. Если область интегрирования криволинейная, то в самом простом варианте можно построить прямоугольник R, стороны которого параллельны осям координат, и ввести вспомогательную функцию

$$f^*(x,y)=\begin{cases} f(x,y), & (x,y)\in G,\\ 0, & (x,y)\in R-G. \end{cases}$$

В таком случае, очевидно

$$\iint_G f(x,y)\,dx\,dy=\iint_G f^*(x,y)\,dx\,dy.$$

Последний интеграл может быть вычислен по общей кубатурной формуле.

4.3. Применение математических пакетов для вычисления интегралов

Вычисление кратных интегралов в Matlab может быть выполнено с помощью функции **dblquad**. Как и для функций **quad** и **quadl** здесь точностью вычислений тоже можно управлять. Использование **dblquad** требует создания файл-функции с двумя параметрами, описывающей подынтегральное выражение. При этом при обращении к **dblquad** следует помнить, что среди ее параметров, определяющих пределы интегрирования, вначале указываются пределы внутреннего интеграла. В Mathcad вычисления кратных интегралов выполняются использованием соответствующего количества раз математического символа интегрирования.

Рассмотрим в качестве примера использования указанных пакетов решение следующей задачи.

Пример 4.2. Определить степень черноты ε полупрозрачного однородного по объему цилиндра бесконечной длины, заполненного плазмой с оптической плотностью $\tau = kR$, где k, R – коэффициент оптического поглощения плазмы и внутренний радиус цилиндра соответственно. Построить график зависимости $\varepsilon(\tau)$ в диапазоне изменения $\tau = 0...10$.

Искомая степень черноты определяется с помощью двойного интеграла

$$\varepsilon = \frac{4}{\pi} \int_0^{\pi/2} d\varphi \int_0^{\pi/2} [1 - \exp(-kl)] \cos\theta \sin\theta d\theta d\varphi,$$

где $l = \dfrac{2R\cos\theta}{1 - \sin^2\theta\cos^2\varphi}$.

Реализация задачи в Matlab и Mathcad представлена ниже. На рис. 4.1 искомый график построен средствами Matlab, а на рис. 4.2 – средствами Mathcad.

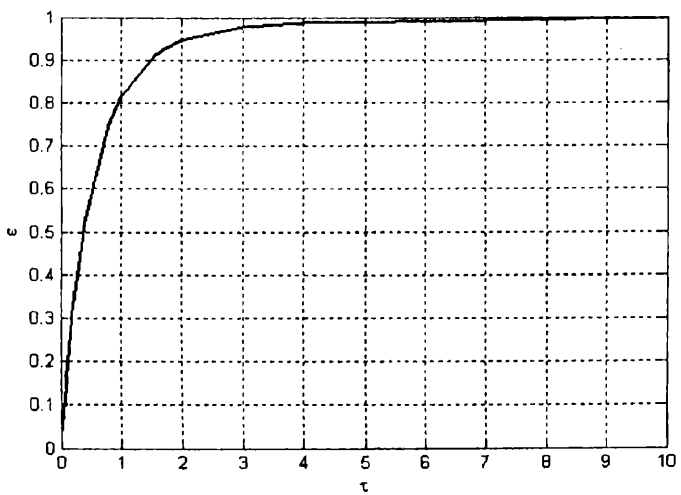

Рис. 4.1. Применение функции **dblquad** из пакета Matlab для вычисления двойных интегралов. Зависимость $\varepsilon(\tau)$

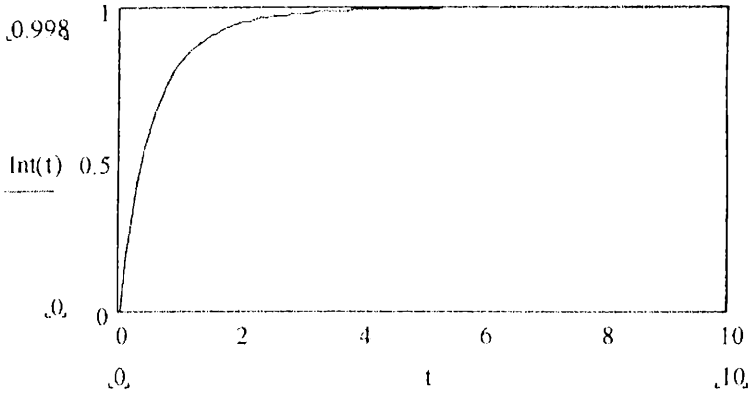

Рис. 4.2. Результат решения задачи в Mathcad

Интегрирование функции двух переменных. Решение задачи в Matlab.

```
global tau;
t=[0.01 0.02 0.05 0.07 0.1 0.4 0.8 1 1.5 1.7 2 3 4 10];
for i=1:15
   tau=t(i);
   int(i)=4/pi.*dblquad('f',1e-7,pi/2,0,pi/2);
end;
plot(t,int);
grid on;

function ff=f(fi,tet);
global tau;
ff=(1-exp(-2.*tau.*cos(tet)./
(1-sin(tet).^2.*cos(fi).^2))).*cos(tet).*sin(tet);
```

Решение задачи в Mathcad.

t:=0,0.1..1

$$\mathrm{Int}(t) := \frac{4}{\pi}\int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\left(1-\exp\left(-2t\frac{\cos(\mathrm{tet})}{1-\sin(\mathrm{tet})^2\cos(\mathrm{fi})^2}\right)\right)\cos(\mathrm{tet})\sin(\mathrm{tet})\,d\mathrm{tet}\,d\mathrm{f}.$$

5. ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ. ЗАДАЧА КОШИ

5.1. Общие замечания

Обыкновенными дифференциальными уравнениями называются уравнения с одной независимой переменной. Если независимых переменных больше, чем одна, то уравнение называется дифференциальным уравнением с частными производными.

С помощью обыкновенных дифференциальных уравнений строятся модели движения систем взаимодействующих частиц, электротехнических процессов в электрических цепях, кинетики химических реакций, процессов заселения уровней энергии в высокотемпературных средах и многих других объектов и процессов.

К задачам, описываемым обыкновенными дифференциальными уравнениями, сводятся некоторые задачи, построенные на уравнениях в частных производных, когда многомерное уравнение позволяет провести разделение переменных (например, при вычислении энергетического спектра частиц в полях определенной симметрии).

Обыкновенное дифференциальное уравнение любого порядка при помощи замены переменных может быть сведено к системе уравнений первого порядка. Рассмотрим в связи с последним следующий пример.

Дифференциальное уравнение третьего порядка

$$a(x)\frac{d^3v}{dx^3}+b(x)\frac{d^2v}{dx^2}+c(x)\frac{dv}{dx}+d(x)v=f(x)$$

заменой переменных

$$\frac{d^2v}{dx^2}=v_2,\quad \frac{dv}{dx}=v_1$$

приводится к следующей системе дифференциальных уравнений:

$$\frac{dv}{dx} = v_1,\ \frac{dv_1}{dx} = v_2,$$

$$a(x)\frac{dv_2}{dx} = -b(x)v_2 - c(x)v_1 - d(x)v + f(x).$$

В общем виде преобразование выглядит следующим образом: ифференциальное уравнение n-го порядка

$$v^{(n)}(x) = \varphi\left(x, v, v', v'', \ldots, v^{(n-1)}\right)$$

аменой переменных

$$v^{(k)} \equiv v_k$$

водится к системе n уравнений первого порядка

$$v_k' = v_{k+1},\quad 0 \le k \le n-2,$$
$$v_{n-1}'(x) = \varphi\left(x, v_0, v_1, v_2, \ldots, v_{n-1}\right),$$

де обозначено $v_0 \equiv v$.

В соответствии с изложенным выше далее будут рассматриваться системы уравнений первого порядка:

$$v_k'(x) = \varphi_k\left(x, v_1, v_2, \ldots, v_n\right),\ 1 \le k \le n.$$

Решение системы n-го порядка зависит от n параметров $c_1, c_2, \ldots, c_n$. Для выделения единственного решения необходимо использование дополнительных условий для искомой функции. В зависимости от того, каким образом ставятся данные условия, различают три типа задач для обыкновенных дифференциальных уравнений: задача Коши, краевая задача и задача на собственные значения.

В задаче Коши все дополнительные условия ставятся в одной точке:

$$v_k(x_0) = v_{k,0},\ 1 \le k \le n.$$

Решение отыскивается в некотором интервале $x_0 \le x \le x_l$.

Если правые части φ_k уравнений непрерывны в некоторо окрестности начальной точки $(x_0, v_{1,0}, v_{2,0}, ..., v_{n,0})$ и удовлетво ряют условию Липшица по переменным v_k, то решение задач Коши существует, единственно и непрерывно зависит от координат начальной точки, т. е. задача является корректной. Условие Липшица формулируется следующим образом:

$$\left|\varphi_k\left(x, v_{1,l}, v_{2,l}, \ldots, v_{n,l}\right) - \varphi_k\left(x, v_{1,m}, v_{2,m}, \ldots, v_{n,m}\right)\right| \le$$
$$\le L\left\{\left|v_{1,l} - v_{1,m}\right| + \left|v_{2,l} - v_{2,m}\right| + \ldots + \left|v_{n,l} - v_{n,m}\right|\right.$$

для любых точек $\left(x, v_{1,l}, v_{2,l}, \ldots, v_{n,l}\right), \left(x, v_{1,m}, v_{2,m}, \ldots, v_{n,m}\right)$.

5.2. Методы решения

Можно выделить три метода решения обыкновенных дифференциальных уравнений: точные, аналитические приближенные численные.

Точные методы предусматривают получение решения в виде комбинации элементарных функций или в виде квадратур от последних. Возможности точных методов ограничены.

Приближенные методы сводятся к построению последовательности функций $w_n(x)$, имеющих пределом искомую функцию $v(x)$. Обрывая эту последовательность на каком-то номере k, получают приближенное решение.

Наиболее универсальными методами решения являются численные. Их основной недостаток – возможность получения только частного решения.

Следует иметь в виду следующее обстоятельство: успех применения численного метода в значительной степени зависит от обусловленности задачи, т. е. задача должна быть хорошо обусловлена, а именно малые изменения начальных условий должны приводить к малому изменению решения. В противном случае (слабой устойчивости) малые погрешности в начальных данных или погрешности численного метода могут приводить к большим погрешностям в решении.

Далее будут рассмотрены алгоритмы решения задачи Коши на примере одного уравнения первого порядка $v'(x)=\varphi(x,v)$. Обобщение на случай системы n уравнений осуществляется заменой $v(x)$ на $\overline{v}(x)$ и $\varphi(x,v)$ на $\overline{\varphi}(x,\overline{v})$, где

$$\overline{v}(x)=\begin{vmatrix} v_1 \\ v_2 \\ \cdot \\ v_n \end{vmatrix}, \qquad \overline{\varphi}(x,\overline{v})=\begin{vmatrix} \varphi_1 \\ \varphi_2 \\ \cdot \\ \varphi_n \end{vmatrix}.$$

5.2.1. Метод Пикара

Данный метод является одним из приближенных методов решения рассматриваемого класса задач. Идея метода чрезвычайно проста и сводится к использованию последовательных приближений для решения интегрального уравнения, к которому приводится исходное дифференциальное уравнение. Пусть поставлена задача Коши:

$$v'(x)=\varphi(x,v(x)), \tag{5.1}$$

$$x_0 \le x \le x_l,$$

$$v(x_0)=v_0.$$

Проинтегрируем выписанное уравнение:

$$v(x)=v_0+\int_{x_0}^{x}\varphi(t,\ v(t))dt. \tag{5.2}$$

Процедура последовательных приближений метода Пикара реализуется согласно следующей схеме:

$$y_s(x)=v_0+\int_{x_0}^{x}\varphi(t,\ y_{s-1}(t))dt, \tag{5.3}$$

причем $y_0(t)=v_0$ (i – номер итерации).

Пример 5.1. Решить методом Пикара уравнение

$$v'(x) = x^3 + v^3,$$
$$v(0) = 0.$$

Решение этого уравнения не выражается через элементарные функции:

$$y_1(x) = 0 + \int_0^x t^3 dt = \frac{x^4}{4},$$

$$y_2(x) = 0 + \int_0^x [t^3 + (\frac{t^4}{4})^3]dt = \frac{x^4}{4}(1 + \frac{1}{4^2 \cdot 13} x^9),$$

$$y_3(x) = 0 + \int_0^x \{t^3 + [\frac{t^4}{4}(1 + \frac{1}{4^2 \cdot 13} t^9)]^3\}dt =$$

$$= \frac{x^4}{4} + \frac{x^{13}}{4^3 \cdot 13}\left(1 + \frac{3}{4^2 \cdot 22} x^9 + \frac{3}{4^4 \cdot 13 \cdot 31} x^{18} + \frac{1}{4^6 \cdot 13^2 \cdot 40} x^{27}\right)$$

и т. д.

Видно, что при $x \le 1$ ряд быстро сходится. Метод удобен, если интегралы можно взять аналитически.

Докажем сходимость метода Пикара. Пусть в некоторой ограниченной области $g(x, v)$ правая часть $\varphi(x, v)$ непрерывна и, кроме того, удовлетворяет условию Липшица по переменной v, т. е.

$$|\varphi(x, v_1) - \varphi(x, v_2)| \le L|v_1 - v_2|,$$

где L – некоторая константа.

В силу ограниченности области $g(x, v)$ имеют место неравенства

$$|x - x_0| \le l_1, \quad |v - v_0| \le V.$$

Вычтем из (5.3) формулу (5.2) и получим для модулей правой и левой частей

$$|y_s(x) - v(x)| = \left| \int_{x_0}^x \varphi(t,\ y_{s-1}(t))\, dt - \int_{x_0}^x \varphi(t, v(t))\, dt \right|,$$

или

$$\left|y_s(x)-v(x)\right| \le \int_{x_0}^{x} \left|\varphi(t, y_{s-1}(t)) - \varphi(t, v(t))\right| dt.$$

Используя условие непрерывности Липшица, получим

$$\left|z_s(x)\right| \le L \int_{x_0}^{x} \left|z_{s-1}(t)\right| dt, \tag{5.4}$$

где $z_s(x) = y_s(x) - v(x)$ – погрешность приближенного решения.

Последовательное применение формулы (5.4) при $s = 1, 2, \ldots$ с учетом того, что $\left|z_0(x)\right| = \left|y_0 - v(x)\right| \le V$, дает следующую цепочку соотношений:

$$\left|z_1(x)\right| \le LV\left|x - x_0\right|,$$

$$\left|z_2(x)\right| \le \frac{1}{2} L^2 V \left|(x-x_0)^2\right|,$$

$$\left|z_s(x)\right| \le \frac{1}{s!} L^s V \left|(x-x_0)^s\right|.$$

Поскольку $\left|x - x_0\right| \le l_1$, то имеем

$$\left|z_s(x)\right| \le \frac{1}{s!} L^s V l_1^s = \frac{1}{s!} V (L l_1)^s.$$

Заменяя $s!$ по формуле Стирлинга, окончательно получим оценку погрешности приближенного решения:

$$\left|z_s(x)\right| \le \frac{V}{\sqrt{2\pi s}} \left(\frac{e l_1 L}{s}\right)^s. \tag{5.5}$$

Из (5.4) следует, что при $s \to \infty$ модуль погрешности $\left|z_s(x)\right| \to 0$, т. е. приближенное решение равномерно сходится к точному.

5.2.2. Методы Рунге–Кутта

Данные методы являются численными. На практике применяются методы Рунге–Кутта, обеспечивающие построение разностных схем (методов) различного порядка точности. Наиболее упот-

ребительны схемы (методы) второго и четвертого порядков. Их мы и рассмотрим ниже.

Предварительно введем некоторые понятия и определения. Сеткой на отрезке $[a, b]$ называется фиксированное множество точек этого отрезка ω_N. Функция, определенная в данных точках, называется сеточной функцией. Координаты точек x_i удовлетворяют условиям

$$a = x_0 < x_1 < x_2 < \ \ldots \ < x_{N-2} < x_{N-1} < x_N = b.$$

Точки $x_i \in \omega_N$ являются узлами сетки. Равномерной сеткой на $[a, b]$ называется множество точек

$$\omega_h = \{x_i = a + ih\}, \qquad i = 0, 1, 2, \ldots, N,$$

где $h = \dfrac{b-a}{N}$ – шаг сетки.

При решении дифференциальных уравнений приближенным методом основным является вопрос о сходимости. Применительно к разностным методам традиционно более употребительно понятие сходимости при $h \to 0$. Обозначим значения сеточной функции через y_i, значения точного решения дифференциального уравнения (5.1) в узле i – через $v(x_i)$ (значения y_i являются приближенными значениями $v(x_i)$). Сходимость при $h \to 0$ означает следующее. Фиксируем точку x и строим совокупность сеток ω_h таким образом, что $h \to 0$ и $x_i = a + ih = x$ (при этом $i \to \infty$). Тогда считают, что численный метод сходится в точке x, если $|y_i - v(x_i)| \to 0$ при $h \to 0$ и $x_i = x$. Метод сходится на отрезке $[a, b]$, если он сходится в каждой точке $x \in [a, b]$. Говорят, что метод имеет p-й порядок точности, если можно найти такое число $p > 0$, что $|y_i - v(x_i)| = O(h^p)$ при $h \to 0$.

Введем далее понятие невязки, или погрешности аппроксимации разностного уравнения, заменяющего заданное дифференциальное уравнение на решении исходного уравнения, т. е. невязка ψ_i представляет собой результат подстановки точного решения уравнения (5.1) $v(x)$ в разностное уравнение. Например, (5.1) можно заменить следующим простейшим разностным уравнением:

$$\frac{y_{i+1}-y_i}{h}-\varphi(x_i,\ y_i)=0,\qquad i=0,\ 1,\ 2,\ \ldots,\ y_0=v_0\,.$$

Тогда невязка определится следующим выражением:

$$\psi_i=-\frac{u_{i+1}-u_i}{h}+\varphi(x_i,\ u_i)\,.$$

Приближенное решение не совпадает, вообще говоря, с u_i, поэтому невязка ψ_i в i-й точке не равна нулю. Вводят следующее определение: численный метод аппроксимирует исходное дифференциальное уравнение, если невязка $\psi_i\to 0$ при $h\to 0$, и имеет p-й порядок аппроксимации, если $\psi_i=O(h^p)$. Доказывается, что порядок точности численного метода решения дифференциального уравнения совпадает с порядком аппроксимации при достаточно общих предположениях.

Теперь перейдем к анализу схем Рунге–Кутта. Сначала обратимся к схемам второго порядка точности. Используя формулу Тейлора, решение дифференциального уравнения (5.1) можно представить в виде

$$v_{n+1}=v_n+h_n v_n'+\frac{1}{2}h_n^2 v_n''+\ldots, \tag{5.6}$$

где обозначено $v_n=v(x_n)$, $v_n'=v'(x_n)$, $h_n=x_{n+1}-x_n$.

Согласно (5.1), $v_n'=\varphi(x_n,\ v_n)$, $v_n''=\varphi_x'(x_n,\ v_n)+\varphi_v'(x_n,v_n)\times$ $\times\varphi(x_n,v_n)$. Далее удерживаем только выписанные члены ряда. Представим вторую производную следующим образом:

$$v_n''=(v_n')'=\frac{\varphi(\tilde{x},\ \tilde{v})-\varphi(x_n,\ v_n)}{\Delta x},$$

где $\tilde{x}$, $\tilde{v}$ – пока неизвестные величины. Пусть

$$\tilde{x}=x_n+\gamma h\,,\ \tilde{v}=v_n+\delta h.$$

Обозначим приближенное значение решения в узле с номером n через y_n (именно это решение будет получаться после того, как мы ограничим ряд членами с порядком не выше второго).

Имеем

$$y_{n+1} = y_n + h_n\varphi(x_n,\, y_n) + \frac{1}{2}h_n^2\left[\frac{\varphi(x_n + \gamma h_n,\, y_n + \delta h_n) - \varphi(x_n,\, y_n)}{\Delta x}\right] =$$

$$= y_n + h_n[\beta\varphi(x_n,\, y_n) + \alpha\varphi(x_n + \gamma h_n,\; y_n + \delta h_n)] \;.$$

Введенные здесь параметры α, β, γ и δ подлежат определению. Разлагая правую часть в ряд Тейлора до линейных членов и приводя подобные члены, получим последовательно

$$y_{n+1} = y_n + h_n\{\,\beta\varphi(x_n, y_n) + \alpha[\varphi(x_n, y_n) +$$

$$+\varphi'_x(x_n,\, y_n)\gamma h_n + \varphi'_y(x_n,\, y_n)\delta h_n]\,\} =$$

$$= y_n + (\alpha+\beta)h_n\varphi(x_n,\, y_n) + \alpha h_n^2[\gamma\varphi'_x(x_n,\, y_n) + \delta\varphi'_y(x_n,\, y_n)]. \quad (5.7)$$

Условием выбора параметров α, β, γ и δ поставим близость выражения (5.7) ряду (5.6); тогда

$$\alpha+\beta=1, \quad \alpha\gamma=\frac{1}{2}, \quad \alpha\delta=\frac{1}{2}\varphi(x_n, y_n).$$

Один параметр остается свободным. Пусть это будет α, тогда

$$\beta=1-\alpha, \;\; \gamma=\frac{1}{2\alpha}, \;\; \delta=\frac{1}{2\alpha}\varphi(x_n,\, y_n),$$

и окончательно из (5.7) с учетом найденных отношений для β, γ и δ получим

$$y_{n+1} = y_n + h_n\{(1-\alpha)\varphi(x_n, y_n) + \\ +\alpha\varphi(x_n + \frac{1}{2\alpha}h_n,\; y_n + \frac{h_n}{2\alpha}\varphi'(x_n,\, y_n)\,\}. \quad (5.8)$$

Соотношение (5.8) описывает однопараметрическое семейство двучленных формул Рунге–Кутта.

В специальной литературе доказывается, что если $\varphi(x, y)$ непрерывна и ограничена вместе со своими вторыми производными, то приближенное решение схемы (5.8) равномерно сходится к точному решению с погрешностью $O(\max\; h_n^2)$, т. е. схема (5.8) обладает вторым порядком точности.

В практике расчетов используют формулы (5.8) при значениях параметра $\alpha = \frac{1}{2}$, $\alpha = 1$. Рассмотрим эти варианты.

Случай, когда $\alpha = \frac{1}{2}$. Из (5.8) выводим

$$y_{n+1} = y_n + \frac{h}{2}\left[\varphi(x_n, y_n) + \varphi\left(x_n + h_n, y_n + h_n\varphi(x_n, y_n)\right)\right]. \quad (5.9)$$

Применение формулы (5.9) сводится к следующей последовательности шагов.

1. Вычисляется грубо значение функции $\overline{y}_{n+1}$ (по схеме ломаных):

$$\overline{y}_{n+1} = y_n + h_n\varphi(x_n, y_n).$$

2. Определяется наклон интегральной кривой в точке (x_{n+1}, y_{n+1}):

$$\overline{y}'_{n+1} = \varphi(x_{n+1}, \overline{y}_{n+1}).$$

3. Находится среднее значение производной функции на шаге h_n:

$$y'_{n+\frac{1}{2}} = \frac{1}{2}\left[\varphi(x_n, y_n) + \overline{y}'_{n+1}\right].$$

4. Рассчитывается, наконец, значение функции в $(n+1)$-м узле:

$$y_{n+1} = y_n + hy'_{n+\frac{1}{2}}.$$

Данная схема имеет специальное название «предиктор-корректор».

Случай, когда $\alpha = 1$.

Согласно (5.8), получаем

$$y_{n+1} = y_n + h_n\varphi[x_n + \frac{h_n}{2}, y_n + \frac{h_n}{2}\varphi(x_n, y_n)].$$

Задача решается посредством следующих шагов.

1. Вычисляется значение функции в половинном узле:

$$y_{n+\frac{1}{2}} = y_n + \frac{h_n}{2}\varphi(x_n,\, y_n)\,.$$

2. Определяется значение производной в узле $n+\frac{1}{2}$:

$$y'_{n+\frac{1}{2}} = \varphi(x_n + \frac{h_n}{2},\, y_{n+\frac{1}{2}})\,.$$

3. Находится значение функции в $(n+1)$-м узле:

$$y_{n+1} = y_n + h_n\, y'_{n+\frac{1}{2}}.$$

Помимо рассмотренных выше двучленных схем широкое р пространение в практике расчетов имеют схемы Рунге–Кутта ч вертого порядка точности. Ниже даются без вывода соответ вующие формулы:

$$y_{n+1} = y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6\,,$$

$$k_1 = h_n\varphi(x_n,\, y_n)\,,\quad k_2 = h_n\varphi(x_n + h_n/2,\; y_n + k_1/2)\,, \qquad (5.$$

$$k_3 = h_n\varphi(x_n + h_n/2,\; y_n + k_2/2)\,,\quad k_4 = h_n\varphi(x_n + h_n,\; y_n + k_3)\,.$$

Схемы с бо́льшим числом членов практически не применя ся. Пятичленные формулы обеспечивают четвертый порядок т ности, шестичленные формулы имеют шестой порядок, но их в весьма сложен.

Погрешности приведенных схем Рунге--Кутта определяю максимальными значениями соответствующих производн Оценку погрешностей легко получить для частного случая вы правой части дифференциального уравнения

$$\varphi(x,\, y) \equiv \varphi(x).$$

В этом варианте решение уравнения может быть сведено к кв ратуре и все схемы разностного решения переходят в формулы ч ленного интегрирования. Например, схема (5.9) принимает вид

$$y_{n+1} = y_n + \frac{h_n}{2}\left[\varphi(x_n) + \varphi(x_{n+1})\right]\,,$$

т. е. имеет форму метода трапеций, а схема (5.10) переходит в схему

$$y_{n+1} = y_n + \frac{h_n}{2}\left[\varphi(x_n) + 4\varphi(x_n + h_n/2) + \varphi(x_n + h_n)\right],$$

представляющую собой формулу Симпсона с шагом $\frac{h_n}{2}$.

Оценки погрешности формул трапеций и Симпсона известны (см. раздел 3.2). Например, мажорантные оценки погрешности указанных формул даются следующими выражениями:

$$R_{\text{трап}} \le \frac{x_l - x_0}{12} \max(h_n^2) \max(\varphi''),$$

$$R_{\text{симп}} \le \frac{x_l - x_0}{2880} \max(h_n^4) \max(\varphi^{\text{IV}}).$$

Из приведенных формул видно, что точность схем Рунге–Кутта достаточно высока.

Выбор той или иной из приведенных схем для решения конкретной задачи определяется следующими соображениями. Если функция $\varphi(x, y)$ в правой части уравнения непрерывна и ограничена, а также непрерывны и ограничены ее четвертые производные, то наилучший результат достигается при использовании схемы (5.10); в том случае, когда функция $\varphi(x, y)$ не имеет названных выше производных, предельный (четвертый) порядок схемы (5.10) не может быть достигнут и целесообразным оказывается применение более простых схем .

Помимо схем Рунге–Кутта практический интерес представляют многошаговые методы, которые можно описать следующей системой уравнений:

$$\frac{a_0 y_n + a_1 y_{n-1} + \ldots + a_m y_{n-m}}{h} = b_0\varphi_n + b_1\varphi_{n-1} + \ldots + b_m\varphi_{n-m}, \quad (5.11)$$

где $n = m, m+1, \ldots$; a_k, b_k – числовые коэффициенты, $k = 0, 1, 2, \ldots, m$, $a_0 \neq 0$.

Согласно данному уравнению, расчет начинается со значения $n = m$. В этом случае получается соотношение вида

$$\frac{a_0 y_m + a_1 y_{m-1} + \ldots + a_m y_0}{h} = b_0 \varphi_m + b_1 \varphi_{m-1} + \ldots + b_m \varphi_0, ,$$

т. е. для начала счета надо иметь m начальных значений y_i, $i = 0, 1, 2, \ldots, m-1$. Эти значения y_i приходится вычислять каким-либо другим методом, например методом Рунге–Кутта. В необходимости использовать разные методы счета состоит неудобство многошаговых методов.

Среди многошаговых методов наиболее распространен метод Адамса, схема реализации которого следует из (5.11) при $a_0 = -a_1 = 1$ и $a_k = 0$ для $k = 2, 3, \ldots, m$:

$$\frac{y_n - y_{n-1}}{h} = \sum_{k=0}^{m} b_k \varphi_{n-k}.$$

При $b_0 = 0$ метод Адамса оказывается явным, а при $b_0 \neq 0$ – неявным.

5.2.3. Неявные методы

Введем понятие устойчивости разностного метода. Для этого рассмотрим уже упоминавшееся разностное уравнение многошагового метода

$$\sum_{k=0}^{m} \frac{a_k}{h} y_{n-k} = \sum_{k=0}^{m} b_k \varphi(x_{n-k}, y_{n-k}), \quad n = m, m+1, \ldots \quad (5.12)$$

Однородное разностное уравнение, соответствующее (5.12), имеет вид

$$\sum_{k=0}^{m} a_k y_{n-k} = 0. \quad (5.13)$$

Говорят, что уравнение (5.13) устойчиво по начальным данным, если существует постоянная M, не зависящая от n, такая, что при любых начальных данных $y_0, y_1, \ldots, y_{m-1}$ имеет место неравенство

$$|y_n| \le M \cdot \max_{0 \le j \le m-1} |y_j|, \quad n = m, m+1, \ldots$$

Вопрос устойчивости по начальным данным решается путем рассмотрения корней так называемого характеристического уравнения, получаемого из (5.13), если решение этого уравнения искать в виде $y_{n-k} = q^{n-k}$. Подставляя данное y_{n-k} в (5.13) и сокращая на q^{n-m}, получим характеристическое уравнение для нахождения q:

$$a_0 q^m + a_1 q^{m-1} + \ldots + a_{m-1} q + a_m = 0. \qquad (5.14)$$

Справедлива следующая теорема: для устойчивости уравнения (5.13) по начальным данным необходимо и достаточно, чтобы выполнялось условие корней, а именно, все корни $q_1, q_2, \ldots, q_m$ характеристического уравнения должны располагаться внутри или на границе единичного круга комплексной плоскости, причем на границе не должно быть кратных корней.

Доказывается следующее утверждение: пусть $0 \le nh \le T$, выполнено условие корней, $|y_i - v(x_i)| \to 0$ при $h \to 0$, $i = 0, 1, 2, \ldots, m-1$, и разностное уравнение (5.12) аппроксимирует исходное дифференциальное уравнение (5.1). Тогда решение разностной задачи (5.12) сводится при $h \to 0$ к решению исходной задачи (5.1). Говоря другими словами, из аппроксимации и устойчивости по начальным данным следует сходимость на ограниченном отрезке $[0, T]$.

Сформулированное условие устойчивости, базирующееся на анализе расположения корней характеристического уравнения (5.14), является весьма общим. Конкретизируем вопрос об устойчивости разностного уравнения применительно к асимптотически устойчивым решениям уравнения (5.1). Пусть $\varphi(x, v(x)) = \lambda v(x)$, $\lambda < 0$, т. е.

$$\frac{dv}{dx} = \lambda v(x). \qquad (5.15)$$

Решение этого уравнения асимптотически устойчиво, т. е. при любых $x > 0$ справедлива оценка

$$|v(x+h)| \le |v(x)|. \qquad (5.16)$$

Логично потребовать, чтобы и разностное уравнение давало решение, обладающее свойством (5.16). Используя явный метод Эйлера первого порядка аппроксимации, получим разностный аналог (5.15)

$$\frac{y_{n+1} - y_n}{h} = \lambda y_n, \quad n = 0, 1, 2, \ldots, \qquad (5.17)$$

или

$$y_{n+1} = (1 + h\lambda) y_n, \quad \text{т. е.} \quad q = 1 + h\lambda.$$

Оценка (5.16) будет выполнена для (5.17) только в том случае, если $|q| \le 1$, так как тогда $|y_{n+1}| \le |y_n|$. Из $|q| \le 1$ следует ограничение на шаг h: $0 \le h \le \dfrac{2}{|\lambda|}$.

Разностный метод (5.12) называется абсолютно устойчивым, если устойчивость имеет место при любых $h > 0$, и условно устойчивым, если она может быть обеспечена только введением ограничений на шаг h.

В качестве примера абсолютно устойчивого метода традиционно рассматривается неявный метод Эйлера, имеющий первый порядок аппроксимации:

$$\frac{y_{n+1} - y_n}{h} = \lambda y_{n+1}. \qquad (5.18)$$

Из (5.18) следует

$$y_{n+1} = \frac{y_n}{1 - h\lambda} = \frac{y_n}{1 + h|\lambda|},$$

т. е. $|q| = \dfrac{1}{1 + h|\lambda|} < 1$ всегда, при любых $h > 0$.

Условная устойчивость приводит к необходимости выбирать малые значения шага h, что является недостатком явного метода. Неявный метод, лишенный данного ограничения, имеет другой довольно существенный недостаток, обусловленный необходимостью решения на каждом шаге алгебраического уравнения (или системы уравнений, в общем случае нелинейных).

Запишем разностное уравнение (5.12) для задачи (5.15):

$$\sum_{k=0}^{m}(a_k-\mu b_k)y_{n-k}=0,\quad n=m,\ m+1,\ldots, \tag{5.19}$$

где $\mu=h\lambda$ – в общем случае комплексный параметр.

Характеристическое уравнение для (5.19) имеет вид

$$\sum_{k=0}^{m}(a_k-\mu b_k)q^{m-k}=0. \tag{5.20}$$

При малых μ корни (5.20) близки к корням (5.13).

Областью устойчивости метода (5.12) называется множество точек комплексной плоскости $\mu=h\lambda$, для которых данный метод, примененный к уравнению специального вида (5.15), является устойчивым.

Для явного метода Эйлера условие устойчивости $|1+\mu|\le 1$ при комплексном $\mu=\mu_0+i\mu_1$ ($\mu_0=\mathrm{Re}\,\mu$, $\mu_1=\mathrm{Im}\,\mu$) выглядит следующим образом: $(\mu_0+1)^2+\mu_1^2\le 1$, т. е. областью устойчивости является круг единичного радиуса, центр которого находится в точке $(-1;\,0)$ комплексной плоскости.

Для неявного метода Эйлера условие $\dfrac{1}{|1-\mu|}\le 1$ соответствует неравенству $(1-\mu_0)^2+\mu_1^2\ge 1$, т. е. областью устойчивости является внешняя область круга единичного радиуса с центром в точке $(1;\,0)$.

Разностный метод называется A-устойчивым, если область его устойчивости включает левую полуплоскость $\mathrm{Re}\,\mu<0$ (или $h\,\mathrm{Re}\,\lambda<0$). Следует обратить внимание на то, что уравнение (5.15) асимптотически устойчиво при $\mathrm{Re}\,\lambda<0$. Следовательно, A-устойчивый разностный метод является абсолютно устойчивым (т. е. устойчивым при любых $h>0$), если устойчиво решение исходного дифференциального уравнения.

Из приведенного выше рассмотрения видно, что неявный метод Эйлера обладает свойством A-устойчивости, а явный метод – нет.

Рассмотрим еще один неявный метод более высокого порядка аппроксимации (второго):

$$\frac{y_{n+1}-y_n}{h}=\frac{1}{2}\left[\varphi(x_{n+1},\ y_{n+1})+\varphi(x_n,\ y_n)\right]. \qquad (5.21)$$

Этот метод получается заменой интеграла от правой части (5.1) на длине шага по формуле трапеций. Применительно к уравнению (5.15) метод (5.21) выглядит следующим образом:

$$y_{n+1}=\frac{1+0,5\mu}{1-0,5\mu}y_n,$$

т. е. $\left|\frac{1+0,5\mu}{1-0,5\mu}\right|\le 1$, если $\mathrm{Re}\,\mu\le 0$, т. е. метод (5.21) относится к A- устойчивым.

Существует доказательство следующих положений:

– среди методов (5.12) не существует явных A-устойчивых методов;

– среди неявных линейных многошаговых методов нет A-устойчивых методов, имеющих порядок точности выше второго.

A-устойчивые разностные схемы весьма эффективны при решении так называемых жестких систем уравнений, так как эти методы не накладывают ограничений на шаг h. Рассмотрим подробнее это утверждение.

Система обыкновенных дифференциальных уравнений

$$\frac{d\bar{v}}{dx}=A\bar{v} \qquad (5.22)$$

с не зависящей от x матрицей $A(m\times m)$ называется жесткой, если $\mathrm{Re}\,\lambda_k<0,\ k=1,2,\ldots m$ и отношение $s=\dfrac{\max\limits_{1\le k\le m}\left|\mathrm{Re}\,\lambda_k\right|}{\min\limits_{1\le k\le m}\left|\mathrm{Re}\,\lambda_k\right|}$ велико, где λ_k – собственные числа матрицы A. Величина s называется числом жесткости. Если матрица A зависит от x, то и λ_k зависят от x, тогда вводят переменное число жесткости

$$s(x) = \frac{\max\limits_{1 \le k \le m} \left| \mathrm{Re}\,\lambda_k(x) \right|}{\min\limits_{1 \le k \le m} \left| \mathrm{Re}\,\lambda_k(x) \right|}$$

и оперируют величиной $\sup(x)$ на отрезке интегрирования.

Отличительной особенностью жестких систем является наличие в их решении как быстро, так и медленно убывающих компонент. При $x > 0$ решение системы практически определяется медленно убывающей компонентой, однако, если воспользоваться явными разностными методами, то быстро убывающая составляющая будет отрицательно влиять на устойчивость, и в результате весь расчет необходимо вести с малым шагом интегрирования. При использовании же неявных методов ограничения на шаг сняты, и его величину определяют из условия достижения нужной точности, не заботясь особо об устойчивости.

При решении жестких систем дифференциальных уравнений хорошо зарекомендовал себя метод Гира, который относится к чисто неявным многошаговым разностным методам, общая формула которых выглядит следующим образом:

$$\sum_{k=0}^{m} a_k y_{n-k} = h\varphi(x_n, y_n),$$

т. е. рассматривается частный вариант метода (5.12), когда $b_1 = b_2 = \ldots = b_m = 0$, а $b_0 = 1$.

При $m = 1$ и $a_0 = 1$, $a_1 = -1$ имеем $y_n - y_{n-1} = h\varphi(x_n, y_n)$, т. е. неявный метод Эйлера. При $m = 2$ и $m = 3$ методы выглядят следующим образом:

$$\frac{3}{2} y_n - 2y_{n-1} + \frac{1}{2} y_{n-2} = h\varphi(x_n, y_n), \qquad (5.23)$$

$$\frac{11}{6} y_n - 3y_{n-1} + \frac{3}{2} y_{n-2} - \frac{1}{3} y_{n-3} = h\varphi(x_n, y_n). \qquad (5.24)$$

Разностное уравнение (5.23) имеет второй порядок точности, а (5.24) – третий. Чтобы найти область устойчивости метода, следу-

ет записать аналогичные уравнения для дифференциального уравнения (5.15). Например, (5.23) примет вид

$$\frac{3}{2}y_n - 2y_{n-1} + \frac{1}{2}y_{n-2} = \mu y_n.$$

Соответствующее характеристическое уравнение запишется следующим образом

$$(\frac{3}{2} - \mu)q^2 - 2q + \frac{1}{2} = 0. \qquad (5.25)$$

Наша задача определить область комплексной плоскости $\mu = \mu_0 + i\mu_1$, в точках которой оба корня (5.25) по модулю меньше единицы. Оказывается, что эта область целиком располагается в левой плоскости и метод (5.23) является A-устойчивым.

Метод (5.24) относится к так называемым $A(\alpha)$-устойчивым методам.

5.3. Применение математических пакетов для решения задачи Коши

В Matlab решение задачи Коши может быть выполнено с помощью нескольких функций: **ode45** (реализует метод Рунге–Кутта четвертого и пятого порядков точности), **ode23** (основана на формулах Рунге–Кутта второго и третьего порядков), **ode113** (базируется на методе Адамса–Бэшфорта–Милтона), **ode15s** (основана на многошаговом методе Гира), **ode23s** (одношаговый метод Розенброка второго порядка точности). Решатели (солверы) **ode45**, **ode23**, **ode113** применяются при решении нежестких систем уравнений или систем с небольшой жесткостью. Решение жестких систем выполняется с помощью солверов **ode15s** и **ode23s**. Функция **ode23s** дает более низкую точность, чем **ode15s**, которая позволяет изменять порядок точности. Все функции допускают задание параметров для повышения эффективности вычислений. Назначение этих параметров требует отчетливого понимания методов численного решения систем обыкновенных дифференциальных уравнений. Все солверы ищут решение с точностью 10^{-3}. Точность расчета можно изменять, используя дополнительный параметр **options**, получаемый с помощью функции **odeset**.

В Mathcad для решения задачи Коши также может быть использовано несколько функций: **rkfixed** (реализует метод Рунге – Кутта четвертого порядка точности с фиксированным шагом), **Rkadapt** (основана на методе Рунге–Кутта с переменным шагом), **Bulstoer** (базируется на методе Булирша–Штера).

Рассмотрим в качестве примера использование функций Matlab и Mathcad для решения следующей задачи Коши.

Пример 5.2. Решить систему электротехнических уравнений, описывающих разрядный контур, включающий активное сопротивление R_k, индуктивность L_k и емкость C_k.

Систему электротехнических уравнений необходимо привести к стандартному виду

$$\frac{d\,\overline{y}}{dt} = \overline{f}(t,\ \overline{y}(t)),$$

где $\overline{y}$ и $\overline{f}$ – векторы решений и правых частей уравнений.

Имеем

$$\frac{dI}{dT} = \frac{U - R_k I}{L_k},$$

$$\frac{dU}{dt} = \frac{I}{C_k}.$$

Начальные условия:

$$t = 0,\ \ I = I_0,\ \ U = U_0.$$

Здесь I, U – ток и напряжение на конденсаторе.

Реализация задачи в пакетах Mathcad и Matlab представлена ниже. На рис. 5.1, 5.2 средствами соответствующих пакетов построены графики найденного решения.

Решение задачи Коши в Matlab.

```
global Lk Ck Rk;
%Задание параметров контура
Lk=60e-6;
Ck=150e-6;
Rk=1;
%Задание вектора начальных условий
y0=[0;3000];
%Задание точности вычислений
```

```
options=odeset('reltol',1e-5);
%Обращение к функции ode45
[t,y]=ode45('el',[0 700e-6],y0,options);
%Создание графика решения системы
plot(t,y(:,1),'k-',t,y(:,2),'k-.');
grid on;
%Вывод легенды графика
legend('I','U');
xlabel('t,c');
ylabel('I,A  U,B');

function f=el(t,y);
%Файл – функция, формирующая вектор правых частей
global Lk Ck Rk;
y1=(y(2)-Rk.*y(1))./Lk;
y2=-y(1)./Ck;
f=[y1;y2];
```

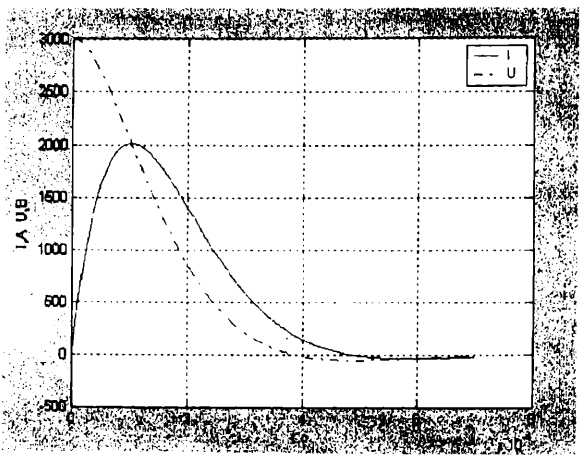

Рис. 5.1. Решение задачи Коши в Matlab

Решение задачи Коши в Mathcad.

Задание исходных данных

$$Lk := 60 \cdot 10^{-6} \quad Ck := 150 \cdot 10^{-6} \quad Rk := 1 \quad y := \begin{pmatrix} 0 \\ 3000 \end{pmatrix}$$

Задание вектора правых частей и вызов функции Rkadapt

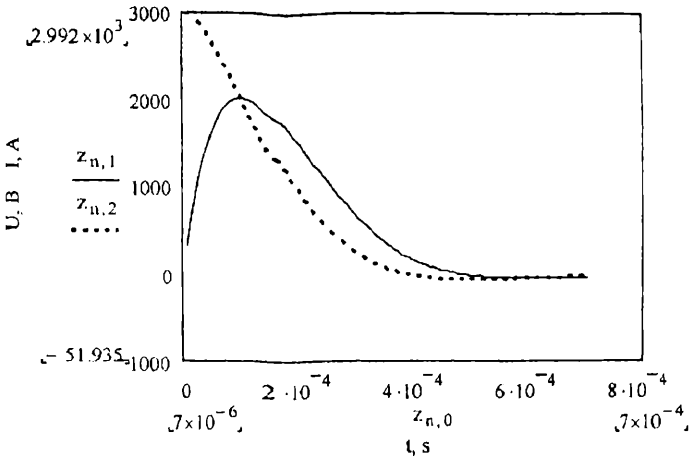

Рис. 5.2. Решение задачи Коши в Mathcad

$$F(t, y) := \begin{bmatrix} \dfrac{(y_1 - Rk\, y_0)}{Lk} \\ \dfrac{-(y_0)}{Ck} \end{bmatrix} \qquad n := 1 .. 100$$

$z := Rkadapt\ (y, 0, 700 \cdot 10^{-6}, 100, F)$

6. ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ. КРАЕВЫЕ ЗАДАЧИ

6.1. Постановка задачи

Стандартная постановка краевой задачи для обыкновенных дифференциальных уравнений выглядит следующим образом:

$$v_k'(x) = \varphi_k(x, v_1, v_2, ..., v_n),\ 1 \le k \le n,$$

а дополнительные условия ставятся более чем в одной точке отрезка интегрирования уравнений (понятно, что в этом случае порядок системы не может быть меньше второго):

$$\psi_k(v_1(\xi_k), v_2(\xi_k), ..., v_n(\xi_k)) = \eta_k,\quad 1 \le k \le p,$$

$$x_0 \le \xi_k \le x_l.$$

Общая классификация методов решения краевых задач та же, что и в случае задачи Коши: существуют точные, аналитические, приближенные и численные методы. О точных методах сказано ранее. Среди приближенных методов можно указать методы Ритца, Галеркина, метод рядов Фурье.

Численные способы решения краевых задач представлены на практике двумя методами: стрельбы и разностным.

6.2. Метод стрельбы

Согласно данному методу, краевая задача сводится к задаче Коши, решение которой осуществляется одним из уже описанных методов.

В качестве примера рассмотрим систему двух дифференциальных уравнений достаточно общего вида с краевыми условиями

$$v'(x) = \varphi(x, v, w), \qquad (6.1$$

$$w'(x) = \eta(x, v, w), \tag{6.2}$$

$$g\big(v(a), w(a)\big) = 0, \tag{6.3}$$

$$\psi\big(v(b), w(b)\big) = 0, \tag{6.4}$$

$$a \le x \le b.$$

Назначаем произвольно $v(a) = \xi$, тогда уравнение (6.3) примет вид $g\big(\xi, w(a)\big) = 0$, откуда в принципе находится $w(a) = \chi(\xi)$.

Итак, краевая задача свелась к задаче Коши с начальными условиями

$$v(a) = \xi, \quad w(a) = \chi(\xi).$$

После численного интегрирования выписанной системы уравнений будет получено решение $v(x, \xi)$ и $w(x, \xi)$, т. е. найденные функции будут зависеть от параметра ξ, произвольно заданного в начале решения.

В силу произвольности выбора правое краевое условие (6.4) не будет удовлетворено, т. е. $\psi\big(v(b, \xi), w(b, \xi)\big) = \bar{\psi}(\xi) \neq 0$.

Меняя параметр ξ, надо добиться того, чтобы $\bar{\psi}(\xi)$ обратилась в нуль, т. е. необходимо отыскать корень уравнения

$$\bar{\psi}(\xi) = 0. \tag{6.5}$$

Самый простой подход здесь состоит в применении метода половинного деления. Алгоритм расчета при этом будет следующим.

Выполняют расчеты с несколькими, вообще говоря, произвольными значениями ξ, имея в виду получить в итоге значения $\bar{\psi}(\xi)$, разные по знаку. Как только данный результат будет достигнут, корень функции (6.5) оказывается локализованным. Далее методом половинного деления находится искомое значение корня (6.5). Для определения значения $\bar{\psi}(\xi_i)$ на каждом шаге данного метода решается система (6.1)–(6.4).

Процесс можно несколько детерминировать, если применить метод секущих. При этом только первые два решения системы

(6.1)–(6.4) находятся с наугад выбранными значениями y_i и ξ_2. Все последующие уточнения параметра ξ производят по формуле метода секущих

$$\xi_{s+1} = \xi_s - \frac{(\xi_s - \xi_{s-1})\bar{\psi}(\xi_s)}{\bar{\psi}(\xi_s) - \bar{\psi}(\xi_{s-1})}.$$

Следует помнить, что метод секущих хорошо сходится только вблизи корня, т. е. само получение результата сильно зависит от того, насколько удачным оказалось начальное приближение.

В качестве примера реализации изложенного выше алгоритма расчета рассмотрим решение методом стрельбы линейных дифференциальных уравнений

$$v'(x) = a_1(x)v(x) + b_1(x)w(x) + c_1(x), \tag{6.6}$$

$$w'(x) = a_2(x)v(x) + b_2(x)w(x) + c_2(x), \tag{6.7}$$

с линейными же краевыми условиями:

$$a \le x \le b,$$

$$e_1 v(a) + g_1 w(a) = d_1, \tag{6.8}$$

$$e_2 v(a) + g_2 w(a) = d_2. \tag{6.9}$$

Следуя алгоритму метода стрельбы, вначале сводим краевую задачу к задаче Коши. Задаем

$$v(a) = \xi, \tag{6.10}$$

а из (6.8) находим

$$w(a) = \frac{d_1 - e_1\xi}{g_1}. \tag{6.11}$$

В силу линейности задачи Коши решение будет линейно зависеть от параметра ξ, поэтому функция $\bar{\psi}(\xi)$ будет линейной функцией. В этом случае точный корень данной функции может быть найден методом секущих за два шага:

$$\xi_3 = \xi_2 - \frac{(\xi_2 - \xi_1)\bar{\psi}(\xi_2)}{\bar{\psi}(\xi_2) - \bar{\psi}(\xi_1)}.$$

Таким образом, потребуется три раза решить задачу Коши, чтобы получить решение указанной линейной краевой задачи.

Метод стрельбы применим для решения линейных и нелинейных задач и позволяет использовать хорошо разработанные алгоритмы для задач Коши. Трудности могут появиться в ситуациях, когда краевая задача хорошо обусловлена, а соответствующая задача Коши плохо обусловлена. В этом случае целесообразно поставить начальные условия на другом конце отрезка и с него начать процедуру решения. При отрицательном результате и в этом случае необходимо перейти к разностным методам.

6.3. Разностный метод

Поставим краевую задачу для линейного дифференциального уравнения второго порядка

$$v''(x) - g(x)v(x) = f(x), \qquad (6.12)$$

$$a \le x \le b,$$

$$v(a) = c, \quad v(b) = d.$$

На отрезке $[a, b]$ строим сетку $\{x_i = x_0 + ih\}$, $i = 0, ..., N$, где h – шаг сетки. Заменяя вторую производную ее разностным аналогом, получим

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} - g_i y_i = f_i,$$

где $g_i = g(x_i)$, $f_i = f(x_i)$.

После преобразований придем к разностному уравнению следующего вида:

$$y_{i-1} - (2 + h^2 g_i) y_i + y_{i+1} = h^2 f_i, \qquad (6.13)$$

$$1 \le i \le N - 1.$$

Получили систему из $(N-1)$-го алгебраического уравнения, в которой неизвестными являются приближенные значения искомой функции в узлах – y_i. Вместе с граничными условиями число уравнений равно числу неизвестных. Решая эту систему уравнений, найдем все y_i.

Рассмотрим вопросы существования и единственности решения и сходимости приближенного разностного решения к точному.

Пусть $g(x) > 0$. Система (6.13) является системой линейных алгебраических уравнений. Коэффициент $g_N > 0$, поэтому матрица этой системы обладает свойством диагонального преобладания. В этом случае, как известно, решение линейной системы существует и оно единственно.

В качестве способа нахождения решения системы (6.13) может быть использован вариант метода Гаусса – метод прогонки, учитывая, что в данном случае матрица системы трехдиагональная.

Докажем сходимость. Пусть $g(x)$ и $f(x)$ дважды непрерывно дифференцируемы. Тогда разностное решение равномерно сходится к точному с погрешностью $O(h^2)$ при $h \to 0$.

Представим вторую производную от функции $v(x)$ в виде разностного аналога с учетом остаточного члена ряда Тейлора в форме Лагранжа:

$$v''(x_i) = \frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} - \frac{1}{12} h^2 v^{IV}(\xi_i),$$

где ξ_i удовлетворяет условию $x_{i-1} \le \xi_i \le x_{i+1}$.

Точное решение (6.12) будет удовлетворять следующему разностному уравнению:

$$v_{i-1} - (2 + h^2 g_i) v_i + v_{i+1} = h^2 f_i + \frac{h^4}{12} v^{IV}(\xi_i). \qquad (6.14)$$

Вычтем последнее уравнение из (6.13) и получим

$$z_{i-1} - (2 + h^2 g_i) z_i + z_{i+1} = -\frac{h^4}{12} v^{IV}(\xi_i), \qquad (6.15)$$

где $z_i = y_i - v_i$ – погрешность приближенного решения.

Перепишем (6.15) вместе с граничными условиями для погрешности в виде

$$(2 + h^2 g_i) z_i = z_{i-1} + z_{i+1} + \frac{h^4}{12} v^{IV}(\xi_i), \qquad (6.16)$$

$$z_0 = 0,\ z_N = 0.$$

Возьмем точку x_m такую, в которой $|z_i|$ достигает максимума (граничная точка не может быть точкой x_m). Принимая во внимание сформулированное выше условие $g_i > 0$, можем в точке x_m записать неравенство, следующее из (6.16):

$$(2 + h^2 g_m)|z_m| \le |z_{m-1}| + |z_{m+1}| + \frac{h^4}{12}\left|v^{\mathrm{IV}}(\xi_m)\right|.$$

Заменяя $|z_{m\pm1}|$ на $|z_m|$, можно только усилить неравенство, в итоге получается оценка

$$z_m \le \frac{h^2}{12}\left|\frac{v^{\mathrm{IV}}(\xi_m)}{g_m}\right| \le \frac{h^2}{12}\max\left|\frac{v^{\mathrm{IV}}(x)}{g(x)}\right|.$$

Отсюда вытекает утверждение, которое следовало доказать.

По поводу устойчивости задачи следует заметить следующее: при $g(x) > 0$ задача Коши плохо обусловлена, а разностная схема (6.13) нечувствительна к этой неустойчивости. В случае, когда $g(x) < 0$, не выполняется достаточное условие прогонки, однако в практических вычислениях данное обстоятельство, как правило, оказывается несущественным и не вызывает сложностей в получении решения.

Приведенная выше разностная схема (6.13) в случае нелинейной задачи усложняется. Если имеется нелинейное дифференциальное уравнение второго порядка

$$v''(x) = f(x, v(x)), \qquad (6.17)$$

$$v(a) = c,\ v(b) = d,$$

то разностная схема в результате тех же действий, что и при построении схемы (6.13), примет вид

$$y_{i-1} - 2y_i + y_{i+1} = h^2 f(x_i, y_i),\ 1 \le i \le N-1, \qquad (6.18)$$

$$y_0 = c,\ y_N = d.$$

Решение (6.18) удобно искать, проведя линеаризацию системы. Процедура линеаризации заключается в следующем. Записывают значение решения в i-й точке на s-й итерации в виде

$$y_i^{(s)} = y_i^{(s-1)} + \Delta_i^{(s)},$$

а функцию в правой части преобразуют с помощью ряда Тейлора:

$$f(x_i, y_i^{(s)}) = f(x_i, y_i^{(s-1)}) + f_y'(x_i, y_i^{(s-1)})\Delta_i^{(s)}.$$

После этого уравнение (6.17) преобразуется следующим образом:

$$(y_{i-1}^{(s-1)} + \Delta_{i-1}^{(s)}) - 2(y_i^{(s-1)} + \Delta_i^{(s)}) + (y_{i+1}^{(s-1)} + \Delta_{i+1}^{(s)}) =$$

$$= h^2 f(x_i, y_i^{(s-1)}) + h^2 f_y'(x_i, y_i^{(s-1)})\Delta_i^{(s)}$$

или

$$\Delta_{i-1}^{(s)} - [2 + h^2 f_y'(x_i, y_i^{(s-1)})]\Delta_i^{(s)} + \Delta_i^{(s)} =$$

$$= h^2 f(x_i, y_i^{(s-1)}) - y_{i-1}^{(s-1)} + 2y_i^{(s-1)} - y_{i+1}^{(s-1)}, \quad 1 \le i \le N-1, \qquad (6.19)$$

$$\Delta_0^{(s)} = 0, \quad \Delta_N^{(s)} = 0.$$

Полученная система решается прогонкой с применением итерационного процесса. Итерации сходятся квадратично. Если линеаризацию не использовать, то итерационная процедура организуется согласно схеме

$$y_{i-1}^{(s)} - 2y_i^{(s)} + y_{i+1}^{(s)} = h^2 f(x_i, y_i^{(s-1)}), \quad 1 \le i \le N-1, \qquad (6.20)$$

$$y_0^{(s)} = c, \quad y_N^{(s)} = d.$$

Здесь итерации сходятся, если $\frac{1}{8}(b-a)^2 M_s < 1$, где $M_s = \max|f_y'|$.

Метод линеаризации более удобен и быстрее приводит к результату, чем метод простых итераций (6.20).

При построении разностных схем в настоящем разделе рассматривались простейшие краевые условия первого рода. На прак-

тике часто используются более сложные краевые условия, так называемые краевые условия второго и третьего рода. В последнем случае запись краевого условия выглядит следующим образом:

$$v'(a) = \varphi(v(a)).$$

Простая аппроксимация производной односторонней разностью $v'(a) \approx \frac{v_1 - v_0}{h}$ дает слишком низкую точность расчета, так как порядок аппроксимации производной такой разностью – $O(h)$. Для повышения точности можно применить следующий прием: по формуле Тейлора

$$v(x_1) = v(x_0) + hv'(x_0) + \frac{1}{2}h^2 v''(x_0) + \ldots,$$

далее с помощью (6.17) заменим вторую производную в этом разложении, а с помощью краевого условия – первую $v'(x_0)$ и получим

$$v(x_1) = v(x_0) + h\varphi(v(x_0)) + \frac{1}{2}h^2 f(x_0, v(x_0)) + \ldots$$

Откуда

$$\frac{y_1 - y_0}{h} = \varphi(y_0) + \frac{1}{2}hf(x_0, y_0),$$

где через y_i как обычно обозначено приближенное решение дифференциального уравнения.

6.4. Применение математических пакетов для решения краевых задач

В Matlab решение краевых задач может быть проведено с помощью солвера **bvp4c**. Технология решения задачи при этом следующая. Составляется файл-функция правых частей системы уравнений первого порядка. По определенным правилам пишется файл-функция граничных условий. Проводится инициализация начального приближения и, наконец, вызывается солвер **bvp4c**.

На Mathcad технологическая цепочка несколько иная. Вначале методом стрельбы находятся недостающие начальные условия с

тем, чтобы перейти от решения краевой задачи к задаче Коши. Затем последняя решается тем или иным методом, указанным в предыдущей главе. Первая часть всей процедуры, связанная с определением начальных условий, реализуется с помощью функции **sbval**, которая, в свою очередь, использует в качестве параметров функции **load** и **score**. Параметрами функции **load** выступают левая граница интервала интегрирования и вектор недостающих граничных условий на этой границе. Значением вектор-функции **load** являются граничные условия на левой границе, причем часть этих условий задана в условии задачи, а для остальной части устанавливаются приближенные значения, которые будут уточняться при работе функции **sbval**. Значением вектор-функции **score** являются вектор разностей между получаемым на правой границе решением и значениями для искомых функций, определенными для правого граничного условия задачи, т. е. этот вектор служит для оценки степени близости получаемых на правой границе значений искомых функций к их значениям, заданным в постановке краевой задачи.

Пример 6.1. Решить краевую задачу

$$u''(x) + u(x) = -x,$$

$$u(0) = 0, \quad u(\frac{\pi}{2}) = 0.$$

Сводим заданное уравнение к системе уравнений первого порядка:

$$y_1' = y_2,$$

$$y_2' = -y_1 - x,$$

$$y_1(0) = 0, \quad y_1(\frac{\pi}{2}) = 0.$$

Сформулированная задача допускает аналитическое решение:

$$u(x) = \frac{\pi}{2}\sin(x) - x.$$

Ниже приведено решение задачи средствами Matlab и Mathcad и выполнено сравнение с точным решением на рис. 6.1, 6.2.

Решение краевой задачи в Matlab.

```
%Файл-программа
%Выполняем инициализацию начального приближения с помощью
%функции bvpinit. Первым параметром этой функции является
%вектор сетки, вторым - вектор начальных приближений для
%функций У1 и У2
initsol=bvpinit([0:pi/30:pi/2],[0 0]);
%Вызываем солвер bvp4c
s=bvp4c('fg','boundar',initsol);
%Строим график решения
plot(s.x,s.y(1,:),'k');
hold on;
%Строим график точного решения
x=[[0:pi/10:pi/2]];
plot(x,pi/2*sin(x)-x,'ko');
grid on;
%Добавляем легенду к графику
legend('bvp4c','exact');

function f=fg(x,y);
%Файл-функция правых частей
f=[y(2);-y(1)-x];

function f=boundar(ya,yb);
%Файл-функция граничных условий
f=[ya(1);yb(1)];
```

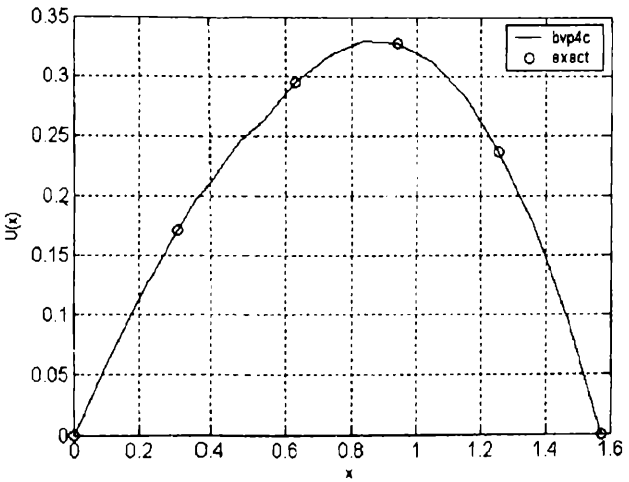

Рис. 6.1. Решение краевой задачи в Matlab

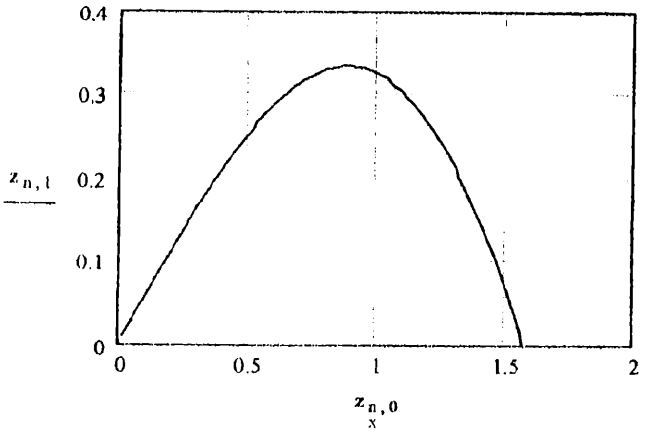

Рис. 6.2. Решение краевой задачи в Mathcad

Решение краевой задачи в Mathcad.

Задаем вектор правых частей и границы интервала интегрирования

$$F(x, y) := \begin{pmatrix} y_1 \\ -y_0 \quad -x \end{pmatrix} \qquad x\,min := 0 \qquad x\,max := \frac{\pi}{2}$$

Задаем приближенно недостающее значение для y1 на левой границе и формируем выражение для функции load

$$v_0 := 1 \qquad load\,(x\,min, v) := \begin{pmatrix} 0 \\ v_0 \end{pmatrix}$$

Определяем функцию score

score (xmax, y):= y – 0

Вызываем функцию sbval

u :=sbval (v, xmin, xmax, F, load, score)

Недостающее значение для производной (y1)

$u_0 = 0.571$

Задаем полный набор начальных условий для задачи Коши и воспользуемся функцией Rkadapt, основанной на методе Рунге–Кутта

$$n := 1..100 \qquad y := \begin{pmatrix} 0 \\ u_0 \end{pmatrix}$$

z:= Rkadapt (y, xmin, xmax, 100, F)

7. РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ И НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Рассматриваются методы решения систем линейных алгебраических уравнений, одного трансцендентного уравнения и систем нелинейных уравнений.

7.1. Решение систем линейных уравнений

Будут изучаться системы линейных уравнений

$$A\bar{x} = \bar{b}, \tag{7.1}$$

матрица которых A имеет определитель, не равный нулю, т. е. $\det A \neq 0$.

При численной реализации систем на компьютере существенное значение имеет вопрос устойчивости решения $\bar{x}$ к малым изменениям правой части $\bar{b}$ и элементов матрицы A. Для анализа этого вопроса представим систему (7.1) в виде $\bar{x} = A^{-1}\bar{b}$. Вариация $\delta\bar{x} = \delta A^{-1}\bar{b} + A^{-1}\delta\bar{b}$. С другой стороны, единичная матрица $E = AA^{-1}$ имеет вариацию $\delta E = 0$. Тогда справедливо соотношение $\delta E = \delta(AA^{-1}) = 0 = \delta A A^{-1} + A\delta A^{-1}$, откуда $\delta A^{-1} = -A^{-1}\delta A A^{-1}$. Подставив выражение для δA^{-1} в формулу для $\delta\bar{x}$, получим

$$\delta\bar{x} = -A^{-1}\delta A\bar{x} + A^{-1}\delta\bar{b} = A^{-1}(\delta\bar{b} - \delta A\bar{x}).$$

Таким образом, формально устойчивость по правой части и матрице имеет место, так как при $\det A \neq 0$ обратная матрица существует. Однако если элементы обратной матрицы велики, то при определенном виде погрешностей малые изменения элементов A и $\bar{b}$ могут сильно исказить решение. В данном случае система на-

зывается плохо обусловленной. Необходимый, но недостаточный признак плохой обусловленности заключается в том, что у таких систем $\det A \approx 0$. Для решения плохо обусловленных систем применяют различные методы регуляризации, которые здесь не рассматриваются.

Методы решения линейных систем делятся на прямые и итерационные. В прямых методах решение получается за строго фиксированное число действий, в итерационных методах строится соответствующий сходящийся процесс, окончание которого зависит от выбранных критериев.

Среди прямых методов наибольшее распространение имеет метод исключения Гаусса. Этот метод основан на приведении A к виду верхней треугольной матрицы. При этом первая строка делится на элемент своего первого столбца (коэффициент при первом неизвестном x_1), затем умножается на элемент первого столбца второй строки и вычитается из нее, умножается на элемент первого столбца третьей строки и тоже вычитается из нее и т. д. В итоге обнуляется первый столбец, начиная со второй строки. Затем рассматривается часть полученной матрицы с элементами $a_{i,j}^{(1)}$, у которой исключены первая строка и первый столбец, и с ней проделывается точно такая же процедура и т. д. После $(n-1)$-го шага обнуляются все элементы матрицы, лежащие ниже главной диагонали, и на главной диагонали последней строки стоит элемент $a_{n,n}^{(n-1)}$. Указанная процедура приведения матрицы к верхнему треугольному виду называется прямым ходом. В обратном ходе из последнего уравнения определяется последнее неизвестное x_n, из предпоследнего уравнения при найденном x_n вычисляется x_{n-1} и т. д. вплоть до определения x_1 (здесь n – количество уравнений системы).

Если при выполнении прямого хода на главной диагонали появляется нуль или малое число, то следует переставить строки так, чтобы переместить на главную диагональ наибольший по модулю элемент столбца (выбор главного элемента). Метод Гаусса требует произвести примерно $\frac{2}{3}n^3$ арифметических действий.

Отметим, что метод Гаусса можно использовать в тех случаях, когда все угловые миноры матрицы системы A отличны от нуля,

при этом можно выполнить разложение матрицы A на нижнюю треугольную матрицу с ненулевыми диагональными элементами и верхнюю треугольную матрицу с единичной диагональю.

Рассмотренный метод можно применить для вычисления определителя. В прямом ходе метода многократно выполняется операция сложения одной строки с другой, взятой с некоторым множителем. Такая операция, как известно, не меняет определителя. При выборе главного элемента приходится переставлять строки, что меняет знак определителя. В итоге определитель может быть вычислен по формуле

$$\det A = (-1)^k a_{11} \cdot a_{22}^{(1)} \cdot a_{33}^{(2)} \cdots \; a_{nn}^{(n-1)},$$

где k – число перестановок строк в процессе выполнения прямого хода.

Далее рассмотрим процедуру нахождения обратной матрицы. Пусть элементы обратной матрицы будут a_{ij}^{-1}. Тогда в соответствии с правилом умножения двух матриц $AA^{-1} = E$ можно записать

$$\sum_{k=1}^{n} a_{i\,k} a_{k\,m}^{-1} = \delta_{im},$$

где δ_{im} – символ Кронекера. Записывая n таких уравнений для $1 \le i \le n$ при фиксированных значениях $m = 1, 2 \ldots, n$, получаем n систем линейных уравнений для определения каждого из n столбцов обратной матрицы $a_{k\,m}^{-1}$. Данные системы уравнений отличаются столбцом правой части, в котором все элементы нулевые, кроме элемента с номером m, равного единице. При этом прямой ход метода Гаусса делается только один раз.

7.2. Решение уравнений с одним неизвестным

Применению любого численного метода для нахождения корней уравнения $f(x) = 0$ должно предшествовать исследование уравнения на предмет определения количества, характера и расположения корней. После этого выбирается численный метод и в ходе итераций выполняется уточнение корней.

1. Наиболее надежный и алгоритмически простой метод определения корней – это метод половинного деления (дихотомия). В

этом методе после локализации корня на отрезке $[a, b]$, т. е. $f(a) \cdot f(b) \le 0$, проводится деление отрезка пополам и из двух половин выбирается та, на концах которой функция $f(x)$ имеет разные знаки. Затем выбранный отрезок снова делится пополам. Процедура повторяется до тех пор, пока не будет выполнен критерий сходимости, в качестве которого удобно использовать соотношение

$$|x_1 - x_2| \le \varepsilon \cdot |\overline{x}|,$$

где x_1, x_2, ε, $\overline{x}$ – значения аргумента x на концах отрезка, относительная точность расчета и текущее значение корня соответственно.

Метод половинного деления применим для любых непрерывных функций, включая недифференцируемые. Скорость сходимости легко прогнозируется, например за 10 итераций отрезок локализации корня уменьшится в 2^{10} раз, т. е. точность расчета корня повышается примерно в 1000 раз. К недостаткам метода следует отнести невозможность обобщения метода на системы уравнений, его неприменимость для нахождения корней четной кратности, неопределенность с тем, к какому корню сойдется процесс при наличии на участке локализации нескольких корней.

2. В методе простых итераций уравнение $f(x) = 0$ заменяют эквивалентным ему уравнением $x = \psi(x)$, например выбирая функцию ψ в виде

$$\psi(x) = x + \chi(x) f(x),$$

где $\chi(x)$ – произвольная непрерывная знакопостоянная функция.

Итерационный процесс строится следующим образом:

$$x_{n+1} = \psi(x_n),$$

где n – номер итерации.

Исследуем условия сходимости, полагая, что точное значение корня равно $\overline{x}$. Используя разложение функции $\psi(x)$ в ряд Тейлора, найдем

$$x_{n+1} - \overline{x} = \psi(x_n) - \psi(\overline{x}) = (x_n - \overline{x})\psi'(\xi), \quad \xi \in (x_n, \overline{x}).$$

Если всюду $|\psi'(x)| \le q < 1$, то отрезки $|x_n - x|$ убывают и последовательность x_n сходится к $\bar{x}$ при любом начальном приближении. Если модуль производной от $\psi(x)$ меньше единицы только в некоторой окрестности корня, то итерации сходятся только при выборе начального приближения достаточно близко к корню. Если процесс сходится и $\psi'(x) < 0$, то последовательные приближения будут попеременно оказываться с разных сторон корня, и его истинное значение будет находиться в интервале $(x_n,\ x_{n-1})$. При $\psi'(x) > 0$ схождение к корню будет монотонным.

3. Одним из наиболее популярных методов отыскания корня является метод Ньютона. Формула метода получается разложением в ряд Тейлора функции $f(x)$ с удержанием в разложении только линейных членов:

$$f(x_n) + (x_{n+1} - x_n) f'(x_n) = 0,$$

откуда значение корня на $n + 1$ итерации определяется по формуле

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

Метод Ньютона можно рассматривать как частный случай метода простых итераций, в котором правая часть имеет специальный вид

$$\psi(x) = x - \frac{f(x)}{f'(x)}.$$

Тогда легко установить, что итерации сходятся, если всюду выполняется неравенство

$$|f(x) f''(x)| < (f'(x))^2,$$

иначе сходимость будет иметь место не при любом начальном приближении, а только в некоторой окрестности корня, так как $\psi'(\bar{x}) = f(\bar{x}) f''(\bar{x}) / f'(\bar{x})^2 = 0$ в виду того, что $f(\bar{x}) = 0$.

Скорость сходимости вблизи простого корня оценивается точно так же, как и в методе простых итераций, с учетом того, что здесь $\psi'(\bar{x}) = 0$:

$$x_{n+1} - \overline{x} = \frac{1}{2}(x_n - \overline{x})^2 \psi''(\xi),$$

т. е. скорость сходимости вблизи корня квадратичная.

4. Если в методе Ньютона заменить производную разностным аналогом, то получится формула метода секущих, который в отличие от двух предыдущих одношаговых методов относится к двухшаговым методам:

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

В методе секущих скорость сходимости меньше, чем в методе Ньютона, и оценивается по формуле

$$x_{n+1} - \overline{x} = a^{0,62}(x_n - \overline{x})^{1,62}.$$

Здесь $a = \dfrac{f''(\overline{x})}{2f'(\overline{x})}$.

5. Можно построить трехшаговый метод, который называется методом парабол. Расчетные соотношения получаются, если записать интерполяционный полином Ньютона по трем узлам и приравнять его нулю:

$$f(x_n) + (x - x_n) f(x_n, x_{n-1}) +$$

$$+ (x - x_n)(x - x_{n-1}) f(x_n, x_{n-1}, x_{n-2}) = 0.$$

Для определения очередного приближения ищутся корни квадратного уравнения

$$ay^2 + by + c = 0,$$

где

$$y = x - x_n, \quad a = f(x_n, x_{n-1}, x_{n-2}),$$
$$b = a(x_n - x_{n-1}) + f(x_n, x_{n-1}), \quad c = f(x_n).$$

Выбирая меньший по модулю корень квадратного уравнения, определяют очередное приближение $x_{n+1} = x_n + y$.

Для начала счета надо, вообще говоря, произвольно задать три начальных значения корня x_0, x_1, x_2. Скорость сходимости у метода парабол

$$x_{n+1} - \overline{x} \infty (x_n - \overline{x})^{1,84},$$

т. е. меньше, чем в методе Ньютона.

Важным преимуществом метода парабол является возможность нахождения с его помощью комплексных корней, например при определении всех корней многочлена высокой степени.

7.3. Системы нелинейных уравнений

Система нелинейных уравнений

$$f_k(x_1, x_2, ..., x_n) = 0, \quad k = 1, ..., n$$

решается обычно методом Ньютона. При этом проводится линейное разложение в ряд Тейлора каждой функции, получается линейная система уравнений, которая решается итерационно одним из известных методов, например методом Гаусса.

Линеаризованная система имеет вид

$$\sum_{k=1}^{n} \frac{df_k(x_1^{(s)}, x_2^{(s)}, ..., x_n^{(s)})}{dx_i} \Delta x_i^{(s)} = -f_k(x_1^{(s)}, x_2^{(s)}, ..., x_n^{(s)}).$$

Из этой системы уравнений на s-итерации находятся приращения $\Delta x_i^{(s)}$, после чего очередное приближение вычисляется по формуле

$$x_i^{(s+1)} = x_i^{(s)} + \Delta x_i^{(s)}.$$

Если определитель линеаризованной системы, составленный из частных производных заданных функций, не равен нулю, то метод сходится в достаточно ограниченной окрестности корней системы.

7.4. Применение математических пакетов для решения задач

Рассмотрим решение нескольких задач на основе рассмотренных выше методов с использованием технологии программирования в Matlab.

Пример 7.1. Определить рабочее давление в цилиндре, заполненном неоднородной плазмой с известным радиальным распределением температуры, по заданному давлению наполнения холодного газа при комнатной температуре. Полученное распределение концентрации тяжелых частиц и температурный профиль, привязанные к радиальной координате, записать в текстовый файл.

Сформулированная задача возникает при моделировании нестационарных процессов в излучающих импульсных газовых разрядах, стабилизированных цилиндрической оболочкой. Цилиндрическая трубка заполняется газом (например, ксеноном) при комнатной температуре T^0 и давлении p_0 и герметизируется. В процессе прохождения тока через газ температура последнего увеличивается, появляется профиль температур $T(z)$, происходит ионизация атомов с образованием ионов различной кратности ионизации, при этом давление в рабочем состоянии возрастает до p. Именно это давление и требуется определить.

Зависимость концентрации тяжелых частиц от температуры и рабочего давления представлена двумерной таблицей. Температурный профиль задается формулой

$$T(z) = (T_w - T_0)z^n + T_0,$$

где T_w, T_0, n – задаваемые параметры профиля (температура на оси цилиндра, на границе, показатель степени, определяющий крутизну профиля).

Все указанные данные находятся в текстовом файле и должны быть, таким образом, прочитаны из этого файла. Данные в файле представлены в следующем виде (концентрации тяжелых частиц представлены в единицах 10^{18} см$^{-3}$ для ксенонового наполнения):

To,K	Tw,K	n	po	Tn,K
8000	2000	4	0.5	300
	Давления, ат			
	5	15	25	
T,K	Концентрации частиц			
2000	18.34	55.03	91.71	
3000	12.23	36.69	61.14	
4000	9.17	27.50	45.86	
5000	7.35	22.05	36.75	
6000	6.12	18.37	30.62	
7000	5.23	15.73	26.23	
8000	4.55	13.71	22.87	

9000	3.97	12.04	20.15
10000	3.43	10.56	17.77
11000	2.88	9.15	15.53
12000	2.37	7.77	13.36
13000	1.93	6.50	11.32
14000	1.6	5.41	9.52
15000	1.38	4.59	8.07
16000	1.23	3.9	6.99

Расчетное уравнение выводится из условия сохранения числа тяжелых частиц (ядер) (атомов и ионов) на единицу длины в исходном (холодном) и в рабочем (горячем) состояниях:

$$\frac{p_0}{kT^0} = 2\int_0^1 n_t(z,\ p)zdz,$$

где z – безразмерная радиальная координата (отношение радиальной координаты к внутреннему радиусу разрядного объема); k – константа Больцмана, n_t – концентрация тяжелых частиц ($см^{-3}$).

Для определения p необходимо решить приведенное уравнение, являющееся уравнением с одним неизвестным вида

$$f(p) = 2\int_0^1 n_t(z,\ p)zdz - \frac{p_0}{kT^0} = 0.$$

После нормирования концентраций на величину 10^{18} и подстановки значения константы k уравнение примет вид

$$2\int_0^1 n_t(z,\ p)zdz - \frac{7242p_0}{T^0} = 0,$$

где давление p_0 берется в атмосферах.

Будем решать данное уравнение методом половинного деления. Для вычисления интеграла применим метод Симпсона, интерполяцию по двумерной таблице проведем с помощью функции **interp2**.

Ввод данных из файла и запись в файл, двумерная интерполяция, дихотомия и интегрирование функции одной переменной:

```
%Открытие файла исходных данных и файла для записи
%результата
ff=fopen('C:\PROFFI\MATLAB\TEACH\EXAM_1\dat.txt','rt');
```

```
fres=fopen('C:\PROFFI\MATLAB\TEACH\EXAM_1\res.txt','wt');
%Ввод данных из файла
s=fgetl(ff);
t0=fscanf(ff,'%g',1);
tw=fscanf(ff,'%g',1);
ns=fscanf(ff,'%g',1);
p0=fscanf(ff,'%g',1);
tn=fscanf(ff,'%g',1);
s=fgetl(ff);s=fgetl(ff);
press=fscanf(ff,'%g',[1,3]);
s=fgetl(ff);
s=fgetl(ff);
c=zeros(1,4);
w=zeros(4,15);
w=fscanf(ff,'%g',[4,15]);
c=w';
%Подготовка массива концентраций для процедуры двумерной
%интерполяции
cw=c;
cw(:,1)=[];
%Ввод количества узлов в формуле численного
%интегрирования
n=input('Enter number point at radius – ');
%Расчет температурного профиля
t=zeros(1,n);
z=zeros(1,n);
h=1/(n-1);
z=[0:h:1];
t=(tw-t0).*z.^ns+t0;
%Расчет рабочего давления
%Задание диапазона поиска искомого корня
p1=5;
p2=25;
pt=(p1+p2)/2;
%Определение вектор-столбца радиального распределения
%концентраций
cr=interp2(press,c(:,1),cw,p1,t,'bilinear');
%Получение подынтегрального выражения
con=cr'.*z;
%Определение значения функции на левой границе интервала
int1=2.*simps(n,con,h)-p0*7242/tn;
```

```
%Цикл, реализующий метод половинного деления
while abs(p2-p1)>pt*1e-4
cr=interp2(press,c(:,1),cw,pt,t,'bilinear');
con=cr'.*z;
int=2*simps(n,con,h)-p0*7242/tn;
if (int.*int1)<0
   p2=pt;
else
   p1=pt;
end;
pt=(p1+p2)/2;
end; %цикла while
%Вывод найденного значения давления на экран
disp('pt=');
disp(pt);
%Запись информации в файл
w1=[z;t;cr'];
fprintf(fres,'    z        T,K     n,cm-3\n');
fprintf(fres,' %5.3e   %5.0f   %5.3e\n',w1);w1=[t;cr'];
fclose(fres);
function s=simps(n,y,h);
%Файл-функция для вычисления интеграла от таблично
%заданной функции методом Симпсона
a=0;
for i=1:(n-1)/2
   a=a+y(2*i-1)+4*y(2*i)+y(2*i+1);
end;
s=a*h/3;
```

Файл результатов имеет следующую структуру:

z	T,K	n,cm-3
0.000e+000	8000	7.857e+000
5.000e-002	8000	7.857e+000
1.000e-001	7999	7.858e+000
1.500e-001	7997	7.861e+000
2.000e-001	7990	7.868e+000
2.500e-001	7977	7.884e+000
3.000e-001	7951	7.914e+000
3.500e-001	7910	7.962e+000
4.000e-001	7846	8.036e+000
4.500e-001	7754	8.143e+000

5.000e-001	7625	8.293e+000
5.500e-001	7451	8.496e+000
6.000e-001	7222	8.762e+000
6.500e-001	6929	9.129e+000
7.000e-001	6559	9.691e+000
7.500e-001	6102	1.039e+001
8.000e-001	5542	1.151e+001
8.500e-001	4868	1.307e+001
9.000e-001	4063	1.559e+001
9.500e-001	3113	2.046e+001
1.000e+000	2000	3.159e+001

Значение p при указанных исходных данных составляет 0,86 МПа.

Пример 7.2. Найти концентрации частиц в плазме трехкомпонентной смеси заданных газов при заданных давлении и температуре. Предусмотреть возможность графического представления результатов в зависимости от температуры и давления.

Для решения задачи записывается следующая система уравнений термодинамики равновесной слабонеидеальной плазмы:

$$\frac{n_e n_{Y,\,i+1}}{n_{Y,\,i}} = K_{Y,\,i}, \quad i = I,\ II, \tag{7.3}$$

$$\frac{n_e n_{Z,\,i+1}}{n_{Z,\,i}} = K_{Z,\,i}, \quad i = I,\ II, \tag{7.4}$$

$$\frac{n_e n_{X,\,i+1}}{n_{X,\,i}} = K_{X,\,i}, \quad i = I,\ II, \tag{7.2}$$

$$n_e = \sum_{j=X,Y,Z} \sum_{i=II}^{III} z_i n_{j,\,i}, \tag{7.5}$$

$$\frac{P}{kT} = n_e + \sum_{j=X,Y,Z} \sum_{i=I}^{III} n_{j,\,i} - \frac{\chi_D^3}{24\pi}, \tag{7.6}$$

$$\sum_{i=I}^{III} n_{Y,\,i} = a_1 \sum_{i=I}^{III} n_{Z,\,i}, \tag{7.7}$$

$$\sum_{i=I}^{III} n_{Y,i} = a_2 \sum_{i=I}^{III} n_{X,i} \tag{7.8}$$

$$K_{j,i} = 2\frac{Q_{j,i+1}}{Q_{j,i}}\left(\frac{2\pi\ m_e kT}{h^3}\right)^{3/2} \exp(-(E_{j,i} - \Delta E_i)/kT), \tag{7.9}$$

$$i = I,\ II,\quad j = X, Y, Z,$$

$$\Delta E_i = kT \cdot \ln \frac{\left(1 + z_{i+1}^2\ \frac{\overline{\Gamma}}{2}\right)\left(1 + \frac{\overline{\Gamma}}{2}\right)}{1 + z_i^2 \frac{\overline{\Gamma}}{2}},\quad i = I,\ II\ , \tag{7.10}$$

$$\overline{\Gamma}^2 = \left(\frac{e^2}{kT}\right)^3 4\pi \left[\frac{n_e}{1 + \frac{\overline{\Gamma}}{2}} + \sum_{j=X,Y,Z} \sum_{i=II}^{III} \frac{n_{j,i} z_i^2}{\left(1 + z_i^2 \frac{\overline{\Gamma}}{2}\right)}\right], \tag{7.11}$$

$$\overline{\Gamma} = \frac{\chi_D e^2}{kT}.$$

В этих формулах введены следующие обозначения: $P, T, K_{X,i}$, $K_{Y,i}, K_{Z,i}, z_i, n_e, n_{X,i}, n_{Y,i}, n_{Z,i}$ – давление, температура плазмы, константы равновесия, заряд частиц различной кратности ионизации, концентрация электронов и частиц i-го сорта соответственно, $z_I = 0,\ z_{II} = 1,\ z_{III} = 2$; римские цифры I, II, III – номера нейтральных атомов, ионов первой и второй кратностей ионизации;

буквы X, Y, Z – компоненты смеси;

$a_1,\ a_2$ – значения соотношения тяжелых частиц;

$Q_{j,i},\ E_{j,i},\ \Delta E_i$ – статистическая сумма, потенциал ионизации индивидуальных частиц сорта (j, i) и снижение потенциала ионизации частиц в плазме вследствие действия внутренних микрополей;

$k,\ h,\ m_e$ – константы Больцмана, Планка и масса электрона.

Исходными данными задачи являются массивы $P, T, Q_{j,i}\ E_{j,i}$, а также значения $a_1,\ a_2$.

В результате расчетов должна быть получена таблица значений концентраций $n_e,\ n_{X,i},\ n_{Y,i},\ n_{Z,i}$ и параметра $\overline{\Gamma}$ для всех заданных значений T при каждом заданном P.

Выписанная система уравнений нелинейная. Для ее решения может быть применен описанный выше метод линеаризации Ньютона.

Предварительно целесообразно сделать следующие замены переменных:

$$n_e^* = n_e \cdot 10^{-18},\ \nu = \ln n_e^*,\ x_i = \ln n_{X,i}^*,\ y_i = \ln n_{Y,i}^*\ z_i = \ln n_{Z,i}^*,$$
$$i = I,\ II,\ III.$$

После линеаризации уравнений (7.2)–(7.8) получается достаточно громоздкая система линейных уравнений $A\overline{w} = \overline{b}$. Вектор w сформирован следующим образом:

$$\overline{w} = \{\nu\ y_1\ y_2\ y_3\ z_1\ z_2\ z_3\ x_1\ x_2\ x_3\}^T.$$

Матрица этой системы A дана в табл. 2, где S – номер итерации.

Вектор правых частей

$$\overline{b} = \{f_1,\ f_2, ...,\ f_{10}\},$$

где

$$f_1 = \ln K_{Y,I},\quad f_2 = \ln K_{Y,II},\quad f_3 = \ln K_{Z,I},\quad f_4 = \ln K_{Z,II},$$

$$f_5 = \ln K_{X,I},\ f_6 = \ln K_{X,II},$$

$$f_7 = \exp(\nu^{S-1})(\nu^{S-1} - 1) +$$
$$+\sum_{i=1}^{3}\left[\exp(y_i^{S-1})(y_i^{S-1} - 1) + \exp(z_i^{S-1})(z_i^{S-1} - 1) + \exp(x_i^{S-1})(x_i^{S-1} - 1)\right] +$$
$$+\frac{7242\,P}{T} + 0{,}285 \cdot 10^{-11}\quad (\overline{\Gamma}\quad T)^3,$$

$$f_8 = \sum_{i=1}^{3}\left[\exp(y_i^{S-1})(y_i^{S-1} - 1) + a_1 \exp(z_i^{S-1})(1 - z_i^{S-1})\right],$$

$$f_9 = \sum_{i=1}^{3}\left[\exp(y_i^{S-1})(y_i^{S-1} - 1) + a_2 \exp(x_i^{S-1})(1 - x_i^{S-1})\right],$$

$$f_{10} = \exp(\nu^{S-1})(\nu^{S-1} - 1) + \exp(y_2^{S-1})(1 - y_2^{S-1}) + 2\exp(y_3^{S-1})(1 - y_3^{S-1}) +$$
$$+\exp(z_2^{S-1})(1 - z_2^{S-1}) + 2\exp(z_3^{S-1})(1 - z_3^{S-1}) + \exp(x_2^{S-1})(1 - x_2^{S-1}) +$$
$$+2\exp(x_3^{S-1})(1 - x_3^{S-1}).$$

Таблица 2

1	-1	1	0	0	0	0	0	0	0
1	0	-1	1	0	0	0	0	0	0
1	0	0	0	-1	1	0	0	0	0
1	0	0	0	0	-1	1	0	0	0
1	0	0	0	0	0	0	-1	1	0
1	0	0	0	0	0	0	0	-1	1
$\exp(v^{S-1})$	$\exp(y_1^{S-1})$	$\exp(y_2^{S-1})$	$\exp(y_3^{S-1})$	$\exp(z_1^{S-1})$	$\exp(z_2^{S-1})$	$\exp(z_3^{S-1})$	$\exp(x_1^{S-1})$	$\exp(x_2^{S-1})$	$\exp(x_3^{S-1})$
0	$\exp(y_1^{S-1})$	$\exp(y_2^{S-1})$	$\exp(y_3^{S-1})$	$-a_1\exp(z_1^{S-1})$	$-a_1\exp(z_2^{S-1})$	$-a_1\exp(z_3^{S-1})$	0	0	0
0	$\exp(y_1^{S-1})$	$\exp(y_2^{S-1})$	$\exp(y_3^{S-1})$	0	0	0	$-a_2\exp(x_1^{S-1})$	$-a_2\exp(x_2^{S-1})$	$-a_2\exp(x_3^{S-1})$
$\exp(v^{S-1})$	0	$-\exp(y_2^{S-1})$	$-2\exp(y_3^{S-1})$	0	$-\exp(z_2^{S-1})$	$-2\exp(z_3^{S-1})$	0	$-\exp(x_2^{S-1})$	$-2\exp(x_3^{S-1})$

Файл исходных данных сформирован для тройной смеси Cs – Hg –Xe следующим образом:

```
ncomp kt  p    tt   a1  a2  eps     nde
3    15  5   4000  1   1.5 0.0001   10
wde (Cs) снижения потенциала ионизации в массиве статсумм атома Y
0.00 0.0057 0.0139 0.0506 0.1024 0.1870 0.3187 0.4445 0.5500 1.088
sty0  (Cs) статсумма атомов компонента Y
 1000  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00 2.00
 2000  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00  2.00 2.00
 3000  2.03  2.03  2.03  2.03  2.03  2.03  2.03  2.03  2.03 2.03
 4000  2.17  2.17  2.17  2.16  2.16  2.16  2.16  2.16  2.15 2.15
 5000  2.54  2.53  2.52  2.50  2.49  2.47  2.46  2.44  2.42 2.39
 6000  3.35  3.29  3.23  3.17  3.11  3.05  3.00  2.92  2.85 2.75
 7000  4.84  4.67  4.50  4.30  4.15  3.97  3.82  3.61  3.45 3.20
 8000  7.19  6.81  6.43  5.99  5.66  5.27  4.95  4.51  4.20 3.73
 9000 10.54  9.82  9.10  8.28  7.67  6.96  6.38  5.61  5.07 4.31
10000 14.89 13.71 12.52 11.17 10.17  9.02  8.10  6.88  6.04 4.91
11000 20.21 18.44 16.65 14.61 13.12 11.41 10.05  8.29  7.09 5.53
12000 26.43 23.92 21.41 18.55 16.46 14.08 12.21  9.81  8.20 6.16
13000 32.65 29.11 25.91 23.07 20.36 17.23 14.79 11.70 10.05 7.27
14000 40.23 35.69 31.57 27.82 24.36 20.38 17.30 13.43 11.39 7.98
15000 48.39 42.73 37.61 32.84 28.57 23.67 19.90 15.20 12.73 8.673
sty12 (Cs) статсуммы первого и второго ионов компонента  Y
1000   1.0  5
2000   1.0  5
3000   1.0  5
4000   1.0  5
5000   1.0  5
6000   1.0  5
7000   1.0  5
8000   1.0  5
9000   1.0  5
10000  1.0  5
11000  1.0  5
12000  1.0  5
13000  1.0  5
14000  1.0  5
15000  1.0  5
stz (Hg) статсуммы атома, первого и второго ионов компонента  Z
```

1000 1.000 2.0 1.0
2000 1.000 2.0 1.0
3000 1.000 2.0 1.0
4000 1.000 2.0 1.0
5000 1.000 2.0 1.0
6000 1.001 2.001 1.0
7000 1.002 2.004 1.002
8000 1.006 2.011 1.005
9000 1.016 2.023 1.011
10000 1.034 2.041 1.023
11000 1.070 2.068 1.041
12000 1.131 2.103 1.067
13000 1.232 2.148 1.103
14000 1.389 2.205 1.147
15000 1.619 2.273 1.203
stx (Xe) статсуммы атома, первого и второго ионов компонента X
1000 1.000 4.0 5.0
2000 1.000 4.0 5.0
3000 1.000 4.0 5.0
4000 1.000 4.045 5.153
5000 1.000 4.096 5.312
6000 1.001 4.160 5.512
7000 1.002 4.229 5.738
8000 1.006 4.301 5.979
9000 1.001 4.371 6.227
10000 1.003 4.439 6.475
11000 1.008 4.504 6.720
12000 1.020 4.566 6.959
13000 1.044 4.625 7.191
14000 1.090 4.682 7.415
15000 1.165 4.737 7.630
ey (Cs) потенциал ионизации атома и первого иона компонента Y
3.89 25.1
ez (Hg) потенциал ионизации атома и первого иона компонента Z
10.43 18.75
ex (Xe) потенциал ионизации атома и первого иона компонента X
12.13 21.2
k_press количество значений давления плазмы
6
press массив давлений
3 5 7 9 11 15

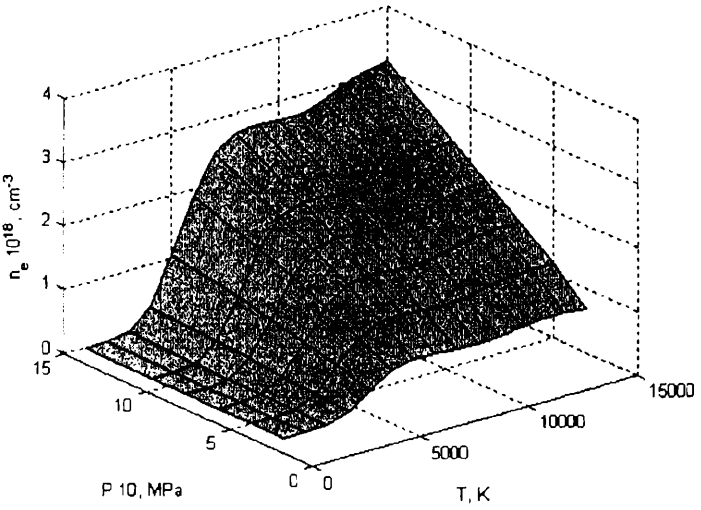

Рис. 7.1. Двумерный график зависимости концентрации электронов от P и T

Алгоритм решения задачи следующий: учитывается, что надо получить таблицу результатов для некоторого набора температур и давлений; для каждого давления вначале рассчитываются начальные приближения, этот расчет проводится при начальной из заданного диапазона температуре, которая достаточно низка, поэтому концентрации ионов в смеси ничтожны, в ней преобладают нейтральные частицы, а снижение потенциала ионизации можно не учитывать. В этом случае удается получить аналитическое решение для концентрации атомов всех компонент и электронов, причем концентрации ионов задать малыми величинами. В дальнейшем в качестве начальных приближений для расчета при очередной температуре берется сошедшееся решение, найденное при предыдущей температуре. Далее рассчитываются элементы матрицы системы A и вектор правых частей $\overline{b}$. Решается система линейных уравнений. При найденных концентрациях из уравнения (7.11) определяется параметр $\overline{\Gamma}$ и по (7.10) рассчитываются снижения потенциала ионизации. Затем уточняются константы равновесия (7.9) , и вновь решается система уравнений. Критерий окончания итераций

$$\left|\frac{w_i^S - w_i^{S-1}}{w_i^S}\right| < \varepsilon, \quad i = 1, ..., 10,$$

где, как и ранее, S – номер итерации.

Ниже представлена реализация алгоритма на Matlab, а на рис. 7.1 дан график зависимости концентрации электронов от температуры и давления.

Ввод данных из файла и запись в файл, решение системы нелинейных уравнений, интерполяция, дихотомия, двумерная графика:

```
%Открытие файлов исходных данных и результатов счета
ff=fopen('C:\PROFFI\MATLAB\TEACH\S\dat_sost.txt','rt');
fres=fopen('C:\PROFFI\MATLAB\TEACH\S\res_sost.txt','wt');
 global tem res0;
%Ввод данных из файла
s=fgetl(ff);
ncomp=fscanf(ff,'%g',1);
kt=fscanf(ff,'%g',1);
p=fscanf(ff,'%g',1);
tt=fscanf(ff,'%g',1);
a1=fscanf(ff,'%g',1);
a2=fscanf(ff,'%g',1);
eps=fscanf(ff,'%g',1);
nde=fscanf(ff,'%g',1);
s=fgetl(ff);s=fgetl(ff);
wde=zeros(1,10);
wde=fscanf(ff,'%g',[1,nde]);
s=fgetl(ff);s=fgetl(ff);
sty0=zeros(kt,nde+1);
w=zeros(nde+1,kt);
w=fscanf(ff,'%g',[nde+1,kt]);
sty0=w';
s=fgetl(ff);s=fgetl(ff);
clear w;
sty12=zeros(kt,3);
w=zeros(3,kt);
w=fscanf(ff,'%g',[3,kt]);
sty12=w';
s=fgetl(ff);s=fgetl(ff);
clear w;
stz=zeros(kt,4);
w=zeros(4,kt);
w=fscanf(ff,'%g',[4,kt]);
stz=w';
```

```
s=fgetl(ff);s=fgetl(ff);
stx=zeros(kt,4);
w=fscanf(ff,'%g',[4,kt]);
stx=w';
s=fgetl(ff);s=fgetl(ff);
ey=zeros(1,2);
ez=zeros(1,2);
ex=zeros(1,2);
ey=fscanf(ff,'%g',[1,2]);
s=fgetl(ff);s=fgetl(ff);
ez=fscanf(ff,'%g',[1,2]);
s=fgetl(ff);s=fgetl(ff);
ex=fscanf(ff,'%g',[1,2]);
s=fgetl(ff);s=fgetl(ff);
k_press=fscanf(ff,'%g',1);
s=fgetl(ff);s=fgetl(ff);
 press=fscanf(ff,'%g',[1,k_press]);
 fclose(ff);
%Конец ввода данных----------
 nu=3.*ncomp+1;
 result=zeros(kt,nu+1,k_press);
 az=2*2.415e-3;
 res=zeros(10,1);
 res0=zeros(10,1);
 kp=zeros(1,10);
 f=zeros(10,1);
 temper=zeros(1,kt);
%Запись поясняющей информации в файл результатов
 fprintf(fres,'          RESULT , 1E+18  \n');
for p1=1:k_press      %цикл по давлению
 p=press(p1);
 fprintf(fres,'              \n');
 fprintf(fres,'       PRESSURE=%5.3f\n',p);
 fprintf(fres,'TEMPER ELECTR   Y0   Y1   Y2  Z0');
 fprintf(fres,'   Z1   Z2   X0   X1   X2');
 fprintf(fres,'      G   De0,eV  De1,eV \n');
 fprintf(fres,'              \n');
%Определение начального приближения
 ny=7.242e+3.*p./sty0(1,1)./(1+1./a1+1./a2);
 nz=ny/a1;
 nx=ny/a2;
```

```
 res0(2)=log(ny);
 res0(5)=log(nz);
 res0(8)=log(nx);
 wu=ey(1)*11603/sty0(1,1);
 kp(1)=az*sty12(1,2)/sty0(1,2)*sty0(1,1)^1.5*exp(-wu);
 res0(1) =log(sqrt(ny*kp(1)));
 res0(3) =res0(1);
 res0(4) =-50;
 res0(6) =-25;
 res0(7) =-50;
 res0(9) =-25;
 res0(10)=-50;
 gw     =0;
 alfa   =0;
for i=1:kt       %цикл по температуре
 tem=sty0(i,1);
 temper(i)=tem;
 flag=0;
while flag==0  %итерационный цикл решения системы
   flag=1;
%Расчет снижения потенциала ионизации
for j=1:2
 de(j)=8.61e-5*tem*log((1+j*j*gw/2)*(1+gw/2)/(1+(j-1)*...
 (j-1)*gw/2));
end;
%Интерполяция статсумм по снижению потенциала ионизации
  wdy=zeros(1,10);
  wdy=sty0(i,2:nde+1);
f1=interp1(wde,wdy,de(1),'linear');
%Расчет констант равновесия
for j=1:2
  wu=(ey(j)-de(j)).*11603./tem;
  if (wu>60)
  wu=60;
  end;
  if j==1
  kp(1)=az.*sty12(i,2)/f1.*tem^1.5.*exp(-wu);
  end;
  if j==2
  kp(2)=az.*sty12(i,3)./sty12(i,2).*tem^1.5.*exp(-wu);
  end;
```

```
end;
for j=1:2
  wu=(ez(j)-de(j)).*11603./tem;
  if (wu>60)
  wu=60;
  end;
  kp(j+2)=az.*stz(i,j+2)./stz(i,j+1).*tem^1.5.*exp(-wu);
end;
for j=1:2
  wu=(ex(j)-de(j))*11603/tem;
  if (wu>60)
  wu=60;
  end;
  kp(j+4)=az.*stx(i,j+2)./stx(i,j+1).*tem^1.5.*exp(-wu);
end;
%Заполнение вектора правых частей системы уравнений
  f(1:6)=log(kp(1:6));
  g1=0;g2=0;g3=0;
for j=1:3
  g1=g1+exp(res0(j+1)).*(res0(j+1)-1);
  g2=g2+exp(res0(j+4)).*(res0(j+4)-1);
  g3=g3+exp(res0(j+7)).*(res0(j+7)-1);
end;
  f(7)=exp(res0(1)).*(res0(1)-
1)+g1+g2+g3+7.242e+3.*p./tem+alfa;
  f(8)=g1-a1*g2;
  f(9)=g1-a2*g3;
  f(10)=exp(res0(1)).*(res0(1)-1)+exp(res0(3)).*(1-
res0(3))+2.*exp(res0(4)).*(1-res0(4))+exp(res0(6)).*(1-
res0(6))+2.*exp(res0(7)).*(1-res0(7))+exp(res0(9)).*(1-
res0(9))+2.*exp(res0(10)).*(1-res0(10));
%Заполнение матрицы системы уравнений
  a=zeros(nu,nu);
  a(1:6,1)=1;
  a(1,2)=-1; a(1,3)=1;
  a(2,3)=-1; a(2,4)=1;
  a(3,5)=-1; a(3,6)=1;
  a(4,6)=-1; a(4,7)=1;
  a(5,8)=-1; a(5,9)=1;
  a(6,9)=-1; a(6,10)=1;
  g1=exp(res0(1));
```

```
 a(7,1)=g1; a(10,1)=g1;
 g1=exp(res0(2)); g2=exp(res0(3)); g3=exp(res0(4));
 a(7:9,2)=g1;
 a(7:9,3)=g2;
 a(7:9,4)=g3;
 a(10,3)=-g2; a(10,4)=-2.*g3;
 g1=exp(res0(5)); g2=exp(res0(6)); g3=exp(res0(7));
 a(7,5)=g1; a(7,6)=g2; a(7,7)=g3;
 a(8,5)=-a1.*g1; a(8,6)=-a1.*g2; a(8,7)-a1.*g3;
 a(10,6)=-g2;  a(10,7)=-2.*g3;
 g1=exp(res0(8)); g2=exp(res0(9)); g3=exp(res0(10));
 a(7,8)=g1; a(7,9)=g2; a(7,10)=g3;
 a(9,8)=-a2.*g1; a(9,9)=-a2.*g2; a(9,10)=-a2.*g3;
 a(10,9)=-g2; a(10,10)=-2.*g3;
%Решение системы уравнений
 res=a\f;
%Проверка условия окончания итераций
for k=1:nu
 if(abs(res0(k)-res(k))> eps*abs(res(k)))
 flag=0;
 break;
 end;
end;
for l=1:nu
 res0(l)=res(l);
 if (res0(l)<-170)
 res0(l)=-170;
 end;
end;
%Определение параметра Г
 gw=fzero('GG',[0 15]);
 alfa=0.285e-11*(gw*tem)*(gw*tem)*(gw*tem);
end %цикла while
 res=exp(res);
%Заполнение массива результатов (концентрации от T и P)
 result(i,1:nu,p1)=res;
 result(i,nu+1,p1)=gw;
%Выдача результатов в файл
 fprintf(fres,'%5.0f',tem);
for k=1:nu
 fprintf(fres,' %4.3e',res(k));
```

```
end;
  fprintf(fres,' g=%5.3f   %5.3f   %5.3f',gw,de(1),de(2));
  fprintf(fres,'\n');
end;  %цикла по температуре
end;  %цикла по давлению
fclose(fres);
clear z;
%Запрос и ввод номера компонента для вывода графика
kv=input('Enter number component for Graphics – ');
z=zeros(k_press,kt);
%Выдача графика
[x,y]=meshgrid(temper,press);
for i=1:kt
for p1=1:k_press
  z(p1,i)=result(i,kv,p1);
end;
end;
surfl(x,y,z);
%shading interp;

function y = GG(g);
%Функция для определения параметра Г
global tem res0
w=exp(res0(3))./(1+g/2)+exp(res0(4))*4./(1+4*g/2)+...
exp(res0(6))./(1+g/2)+exp(res0(7))*4./(1+4*g/2)+...
exp(res0(9))./(1+g/2)+exp(res0(10))*4./(1+4*g/2);
y=5.87e+10./(tem*tem*tem).*(exp(res0(1))./(1+g/2)+w)-g*g;
```

СПИСОК ЛИТЕРАТУРЫ

1. *Калиткин Н. Н.* Численные методы. М.: Наука, 1978. 512 с.

2. *Бахвалов Н.С., Жидков Н.П., Кобельников Г.М.* Численные методы. М.: Наука, 1987. 445с.

3. *Тихонов А.Н., Костомаров Д.П.* Вводные лекции по прикладной математике. М.: Наука, 1984. 192 с.

4. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений: Пер. с англ. М.: Мир, 1980. 177 с.

5. *Самарский А.А.* Теория разностных схем. М.: Наука, 1983. 548 с.

6. *Амосов А.А., Дубинский Ю.А., Копченова Н.В.* Вычислительные методы для инженеров: Учеб. пособие. М.: Высш. шк., 1994. 544 с.

7. *Арушанян О.Б., Залеткин С.Ф.* Численное решение обыкновенных дифференциальных уравнений на Фортране. М.: Изд-во Моск. ун-та, 1990. 336с.

8. *Самарский А.А., Гулин А.В.* Численные методы: Учеб. пособие для вузов. М.: Наука, 1989. 432 с.

9. *Мэтьюз Д.Г., Финк К. Д.* Численные методы. Использование MATLAB: Пер. с англ. М.: Издат. дом «Вильямс», 2001. 720 с.

10. *Мартынов Н.Н.* Введение в MATLAB 6. М.:КУДИЦ- ОБРАЗ, 2002. 352 с.

11. *Потемкин В.Г.* MATLAB 6: среда проектирования инженерных приложений. М.: Диалог – МИФИ, 2003. 448 с.

12. *Дьяконов В.П.* MathCAD 2000: Учебный курс. СПб.: Питер, 2000. 234 с.

ОГЛАВЛЕНИЕ

Владимир Михайлович Градов

Компьютерные технологии в практике математического моделирования

Часть 1

Учебное пособие

Редактор *А.В. Сахарова*
Корректор *О.Ю. Соколова*
Компьютерная верстка *О.В. Беляевой*

Подписано в печать 24.12.2004. Формат 60×84/16. Бумага офсетная.
Печ. л. 6,75. Усл. печ. л. 6,28. Уч.-изд. л. 6,05. Тираж 100 экз. Изд. № 43.
Заказ №6

Издательство МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская, 5.