Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра электронных приборов

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ

Учебное пособие
для студентов направления «Фотоника и оптоинформатика» и «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

2012

Саликаев Юрий Рафаельевич

Компьютерное моделирование и проектирование: учебное пособие для студентов направления «Фотоника и оптоинформатика» и «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства» / Ю.Р. Саликаев; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Томский государственный университет систем управления и радиоэлектроники, Кафедра электронных приборов. - 2-е изд. - Томск: ТУСУР, 2012. – 95 с

В процессе изучения дисциплины студенты должны овладеть численными методами решения задач линейной алгебры и задач математической физики, ясно представлять алгоритмы, положенные в основу используемого программного обеспечения для решения таких задач, уметь решать задачи, связанные с анализом технических объектов, а также грамотно использовать все возможности ПК.

Предназначено для студентов очной и заочной форм, обучающихся по направлению «Фотоника и оптоинформатика» по дисциплине «Компьютерное моделирование и проектирование оптических систем» и напрвления «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства») по дисциплине «Компьютерное моделирование и проектирование электронных приборов и устройств».

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и
радиоэлектроники»

Кафедра электронных приборов

УТВЕРЖДАЮ
Зав.кафедрой ЭП
_____С.М. Шандаров
«___» _____ 2012 г.

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ

Учебное пособие
для студентов направления «Фотоника и оптоинформатика» и
«Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

Разработчик
_____Ю.Р. Саликаев
«____»_____2012 г

2012

Содержание

1 Математическое моделирование и вычислительный эксперимент

Метод исследования, основанный на построении и анализе с помощью ЭВМ математических моделей (ММ) изучаемого объекта, называют *вычислительным экспериментом*.

Пусть, например, требуется исследовать какой-то физический объект, явление, процесс. Тогда схема вычислительного эксперимента выглядит так, как показано на рис. 1.1. Формулируются основные законы, управляющие данным *объектом исследования* (I) и строится соответствующая *ММ* (II), представляющая обычно запись этих законов в форме системы уравнений (алгебраических, дифференциальных, интегральных и т.д.).

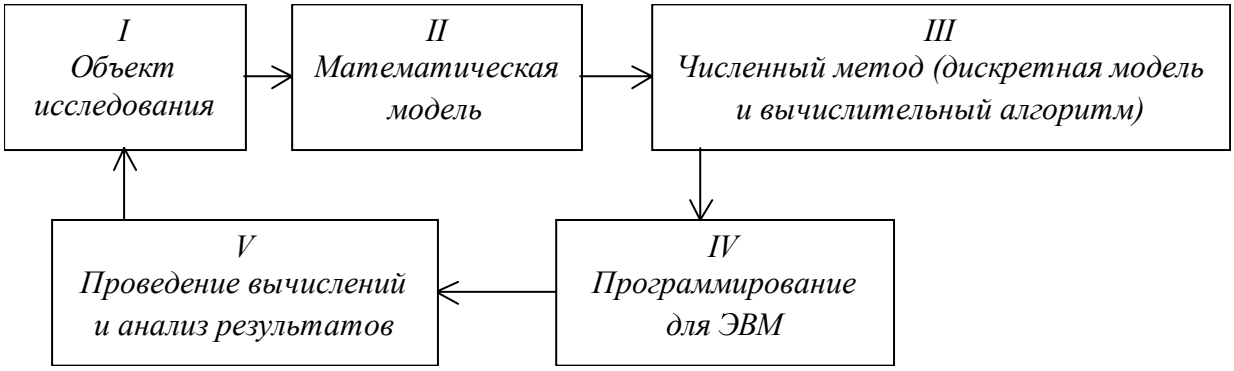

Рисунок 1.1 – Схема вычислительного эксперимента

Для ее решения требуется привлечение ЭВМ и, как следствие, развитие численных методов (см. III на рис. 11). Под *численным методом* (ЧМ) здесь понимается такая интерпретация ММ («дискретная модель»), которая доступна для реализации на ЭВМ. Например, если ММ представляет собой дифференциальное уравнение, то численным методом может быть аппроксимирующее его разностное уравнение совместно с алгоритмом, позволяющим отыскать решение этого разностного уравнения. Результатом реализации численного метода на ЭВМ является число или таблица чисел. Отметим, что в настоящее время помимо собственно численных методов имеются также методы, которые позволяют проводить на ЭВМ аналитические выкладки. Однако аналитические методы для ЭВМ не получили пока достаточно широкого распространения.

Чтобы реализовать численный метод, необходимо составить *программу для ЭВ*М (см. IV на рис. 1.1) или воспользоваться готовой программой. После отладки программы наступает этап *проведения*

вычислений и анализа результатов (V). Полученные результаты изучаются с точки зрения их соответствия исследуемому явлению и при необходимости вносятся исправления в численный метод и уточняется ММ.

Такова в общих чертах схема вычислительного эксперимента. Его основу составляет триада: *модель - метод (алгоритм) -программа.*

Предметом данного учебного пособия является изложение вопросов, отражающих лишь один из этапов вычислительного эксперимента, а именно этап построения и исследования численного метода. Исходные задачи и их математическая постановка не рассматриваются, вопросам программирования и организации вычислений посвящен лабораторный практикум.

2 Устранимые и неустранимые погрешности

Процесс исследования исходного объекта методом математического моделирования и вычислительного эксперимента неизбежно носит приближенный характер, потому что на каждом этапе вносятся те или иные погрешности. Так, построение ММ связано с упрощением исходного явления, недостаточно точным заданием коэффициентов уравнения и других входных данных. По отношению к численному методу, реализующему данную ММ, указанные погрешности являются *неустранимыми*, поскольку они неизбежны в рамках данной модели.

При вычислении с помощью ЭВМ неизбежны *погрешности округлений*, связанные с ограниченностью разрядной сетки машины. Обычно после выполнения операции производится или округление результата, или простое отбрасывание лишних разрядов с целью экономии машинного времени.

Перевод чисел из одной системы счисления в другую также может быть источником погрешности из-за того, что основание одной системы счисления не является степенью основания другой (например, 10 и 2). Это может привести к тому, что в новой системе число становится иррациональным. Например, число 0,1 при переводе в двоичную систему счисления примет вид 0,1=0,00011001100… Может оказаться, что с шагом 0,1 нужно при вычислениях пройти отрезок [0,1] от $x=1$ до $x=0$; десять шагов не дадут точного значения $x=0$. По отношению к численному методу, реализующему данную ММ, указанные погрешности являются *неустранимыми*, поскольку они неизбежны при расчетах на данной ЭВМ.

При переходе от ММ к численному методу возникают погрешности, называемые *погрешностями метода*. Они связаны с тем, что всякий численный метод воспроизводит соответствующую численную модель

приближенно. Наиболее типичной погрешностью метода является *погрешность дискретизации.* Погрешность метода принято считать устранимой, ее обычно стараются довести до величины, в несколько раз меньшей степени погрешности исходных данных. Дальнейшее снижение погрешности не приведет к повышению точности результатов, а лишь приведет к увеличению стоимости расчетов из-за необоснованного увеличения объема вычислений.

3 Требования к вычислительным методам

Одной и той же математической задаче можно поставить в соответствие множество различных дискретных моделей. Однако далеко не все из них пригодны для практической реализации. Вычислительные алгоритмы, предназначенные для быстродействующих ЭВМ, должны удовлетворять многообразным и зачастую противоречивым требованиям. Сформулируем основные из этих требований в общих чертах.

Можно выделить две группы требований к численным методам. Первая группа связана с адекватностью дискретной модели исходной математической задаче, и вторая группа – с реализуемостью численного метода на ЭВМ. К первой группе относятся такие требования, как **устойчивость, корректность, сходимость.**

3.1 Устойчивость численного метода (ЧМ)

Рассмотрим неустранимые погрешности исходных данных. Поскольку вычислитель не может с ними бороться, то нужно хотя бы иметь представление об их влиянии на точность окончательных результатов. Конечно, мы вправе надеяться на то, что погрешность результатов имеет порядок погрешности исходных данных. Всегда ли это так? К сожалению, нет. Некоторые задачи весьма чувствительны к неточностям в исходных данных. Эта чувствительность характеризуется так называемой устойчивостью.

Пусть в результате решения задачи по исходному значению величины x находится значение искомой величины y. Если исходная величина имеет абсолютную погрешность Δx, то решение имеет погрешность Δy. Задача называется *устойчивой* по исходному параметру x, если решение y непрерывно от него зависит, т.е. малое приращение величины Δx приводит к малому приращению искомой величины Δy. Другими словами, малые погрешности в исходной величине приводят к малым погрешностям в результате расчетов.

Отсутствие устойчивости означает, что даже незначительные погрешности в исходных данных приводят к большим погрешностям в решении или вовсе к неверному результату. О подобных неустойчивых задачах говорят, что они *чувствительны* к погрешностям исходных данных.

Примером такой задачи является отыскание действительных корней многочлена вида

$$(x-a)^n = \varepsilon,\ \ 0 < \varepsilon << 1.$$

Изменение правой части на величину порядка ε приводит к погрешности корней порядка $\varepsilon^{1/n}$.

3.2 Корректные и некорректные задачи

Задача называется *поставленной корректно*, если для любых значений исходных данных из некоторого класса ее решение существует, единственно и устойчиво по исходным данным.

Рассмотренная выше в примере неустойчивая задача является некорректно поставленной. Применять для решения таких задач численные методы, как правило, нецелесообразно, поскольку возникающие в расчетах погрешности округлений будут сильно возрастать в ходе вычислений, что приведет к значительному искажению результатов.

Вместе с тем отметим, что в настоящее время развиты методы решения некоторых некорректных задач. Это в основном, так называемые *методы регуляризации*. Они основываются на замене исходной задачи корректно поставленной задачей, содержащей параметр, при стремлении которого к нулю решение этой задачи переходит в решение исходной задачи.

3.3 Неустойчивость ЧМ

Иногда при решении корректно поставленной задачи может оказаться неустойчивым метод ее решения и по этой причине может быть получен результат, не имеющий смысла.

Численный алгоритм (метод) называется *корректным* в случае существования и единственности численного решения при любых значениях исходных данных, а также в случае устойчивости этого решения относительно погрешностей исходных данных.

3.4 Понятие сходимости ЧМ

При анализе точности вычислительного процесса одним из важнейших критериев является сходимость численного метода. Она означает близость получаемого численного решения задачи к истинному решению. Строгие определения разных оценок близости могут быть даны лишь с привлечением аппарата функционального анализа. Здесь мы ограничимся некоторыми параметрами сходимости, необходимыми для понимания последующего материала.

Рассмотрим процесс сходимости итерационного процесса. Этот процесс состоит в том, что для решения некоторых задач и нахождения искомого значения, определяемого параметром (например, корни нелинейного уравнения), строится метод последовательных приближений. В результате многократных повторений этого процесса (или итераций) получаем последовательность значений $x_1, x_2, ..., x_n, ...$ Говорят, что эта последовательность сходится к точному решению $x = a$, если при неограниченном возрастании числа итераций предел этой последовательности существует и равен a: $\lim\limits_{n\to\infty} x_n = a$. В этом случае имеем сходящийся численный метод.

Другой подход к понятию сходимости используется в методах дискретизации. Эти методы заключаются в замене задачи с непрерывными параметрами на задачу, в которой значения функций вычисляются в фиксированных точках. Это относится, в частности, к численному интегрированию, решению дифференциальных уравнений и т.п. Здесь под *сходимостью метода* понимается стремление значения решения дискретной модели задачи к соответствующим значениям решения исходной задачи при стремлении к нулю параметра дискретизации (например, шага интегрирования).

При рассмотрении сходимости важными понятиями являются ее вид, порядок и другие характеристики. С общей точки зрения эти понятия рассматривать нецелесообразно; к ним будем обращаться при изучении тех или иных численных методов.

Таким образом, для получения решения задачи с необходимой точностью ее постановка должна быть корректной, а используемый численный метод должен обладать устойчивостью и сходимостью.

Вторая группа требований, предъявляемых к численным методам, характеризуется возможностью получить на ЭВМ решение соответствующей системы уравнений за приемлемое время. Основным препятствием для реализации корректно поставленного алгоритма является ограниченный объем оперативной памяти ЭВМ и ограниченные

ресурсы времени счета. Реальные вычислительные алгоритмы должны учитывать эти обстоятельства, т.е. они должны быть экономичными как по числу арифметических действий, так и по требуемому объему памяти.

4 Представление вещественных чисел в ЭВМ

Одним из источников вычислительных погрешностей является приближенное представление вещественных чисел в ЭВМ, обусловленное конечностью разрядной сетки. Хотя исходные данные представляются в ЭВМ с большой точностью, накопление погрешностей округления в процессе счета может привести к значительной результирующей погрешности, а некоторые алгоритмы могут оказаться и вовсе непригодными для реального счета на ЭВМ.

Рассмотрим способы представления чисел и связанные с ними погрешности округления.

При ручном счете пользуются десятичной системой счисления. Например, запись 103,67 определяет число

$$1\cdot 10^2 + 0\cdot 10^1 + 3\cdot 10^0 + 6\cdot 10^{-1} + 7\cdot 10^{-2}.$$

Здесь 10 – основание системы счисления, запятая отделяет дробную часть от целой, 1, 0, 3, 6, 7 – числа из базисного набора {0,1,2,3,4,5,6,7,8,9}, с помощью которого можно представить любое вещественное число.

ЭВМ работают, как правило, в двоичной системе, когда любое число записывается в виде последовательности нулей и единиц. Например, запись в 0,0101 в двоичной системе определяет число

$$0\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4}.$$

Как двоичная, так и десятичная системы относятся к позиционным системам счисления. В *позиционной системе с основанием r* запись

$$a = \pm a_n \cdot a_{n-1}\ldots a_0 a_{-1} a_{-2}\ldots \qquad (1)$$

означает, что

$$a = \pm(a_n r^n + a_{n-1}\cdot r^{n-1} + \ldots + a_0\cdot r^0 + a_{-1}\cdot r^{-1} + a_{-2}\cdot r^{-2} + \ldots).$$

Будем считать далее, что r – целое число, большее единицы. Каждое из чисел a_i может принимать одно из значений $\{0,1\ldots,r-1\}$. Числа a_i

называются *разрядами*, например: a_3 – третий разряд перед запятой, a_{-2} – второй разряд после запятой.

Запись вещественного числа в виде (1) называется также его представлением в форме *числа с фиксированной запятой*. В ЭВМ чаще всего используется представление чисел в форме с *плавающей запятой*, т.е. в виде

$$a = M \cdot r^p, \tag{2}$$

где r – основание системы счисления, p – целое число (положительное, отрицательное или нуль) и

$$r^{-1} \le |M| < 1. \tag{3}$$

Число M представляется в форме числа с фиксированной запятой и называется мантиссой числа a. В виде (2) можно единственным образом представить любое вещественное число кроме нуля. Единственность обеспечивается условием нормировки (3).

Например, число 103,67 в форме с плавающей запятой имеет вид $0.10367 \cdot 10^3$, т.е. M=0.10367, p=3. Двоичное число $0,0101 = 0,101 \cdot 2^{-1}$ имеет в двоичной системе мантиссу $M = 0,101$ и порядок $p = -1$.

Знак порядка	Порядок	Знак мантиссы	Мантисса
48	47 42	41	40 1

Рисунок 4.1 – Разрядная сетка ЭВМ

В ЭВМ для записи каждого числа отводится фиксированное число разрядов (*разрядная сетка*). Например, в ЭВМ с 48-разрядной сеткой двоичные разряды распределяются следующим образом: в разрядах с 1 по 40 помещается абсолютное значение мантиссы, в 41 разряде – знак мантиссы, в разрядах с 42 по 47 – абсолютная величина порядка, в 48 разряде – знак порядка (см. рис. 4.1). Отсюда легко найти диапазон чисел, представимых в такой ЭВМ. Поскольку максимальное значение порядка в двоичной системе равно 111111=63 и мантисса не превосходит единицы, то с помощью указанной разрядной сетки можно представить числа, абсолютная величина которых лежит примерно в диапазоне от 2^{-63} до 2^{63}, т.е. от 10^{-19} до 10^{19}.

Ту же 48-разрядную сетку можно использовать для представления чисел с фиксированной запятой. Пусть, например, разряды с 1 по 24 отводятся для записи дробной части числа и разряды с 25 по 47 – для

записи целой части числа. Тогда максимальное число, которое можно представить с помощью данной разрядной сетки, будет равно

Следовательно, в данном случае диапазон допустимых чисел в 10^{12} раз меньше, чем при использовании представления с плавающей запятой. Возможностью существенного увеличения диапазона допустимых чисел при той же разрядной сетке и объясняется преимущественное использование в ЭВМ представления чисел в форме с плавающей запятой. Комплексное число представляется в ЭВМ в виде пары вещественных чисел.

5 Округление чисел в ЭВМ

Будем считать в дальнейшем, что вещественные числа представляются в ЭВМ в форме с плавающей запятой. Минимальное положительное число M_0, которое может быть представлено в ЭВМ с плавающей запятой, называется *машинным нулем*. Мы видим, что для рассмотренной ЭВМ число $M_0 \approx 10^{-19}$. Число $M_\infty = M_0^{-1}$ называют *машинной бесконечностью*. Все вещественные числа, которые могут быть представлены в данной ЭВМ, расположены по абсолютной величине от M_0 до M_∞. Если в процессе счета какой-либо задачи появится вещественное число, меньшее по модулю, чем M_0, то ему присваивается нулевое значение. Так, на рассмотренной ЭВМ в результате перемножения двух чисел 10^{-11} и 10^{-10} получим нуль. При появлении в процессе счета вещественного числа, большего по модулю, чем M_∞, происходит так называемое переполнение разрядной сетки, после чего ЭВМ прекращает счет задачи. Отметим, что нуль и целые числа представляются в ЭВМ особым образом, так что они могут выходить за пределы диапазона $M_0 \div M_\infty$.

Из-за конечности разрядной сетки в ЭВМ можно представить точно не все числа из диапазона $M_0 \div M_\infty$, а лишь конечное множество чисел. Число a не представимое в ЭВМ точно, подвергается *округлению*, т.е. оно заменяется близким ему числом $\tilde{a}$, представимым в ЭВМ точно.

Различают два вида погрешностей – абсолютную и относительную. *Абсолютная погрешность* некоторого числа равна разности между его истинным значением и приближенным значением, полученным в результате его округления. *Относительная погрешность* – это отношение абсолютной погрешности к значению числа.

Точность представления чисел в ЭВМ с плавающей запятой характеризуется *относительной погрешностью* $|a-\tilde{a}|/|a|$.

Величина относительной погрешности зависит от способа округления. Простейшим, но не самым точным способом округления является отбрасывание всех разрядов мантиссы числа a, которые выходят за пределы разрядной сетки.

Найдем границу относительной погрешности при таком способе округления. Пусть для записи мантиссы в ЭВМ отводится t двоичных разрядов. Предположим, что надо записать число, представленное в виде бесконечной десятичной дроби

$$a = \pm 2^p \left(\frac{a_1}{2} + \frac{a_2}{2^2} + \ldots + \frac{a_t}{2^t} + \frac{a_{t+1}}{2^{t+1}} + \ldots\right),$$

где каждое из a_j равно 0 или 1. Отбрасывая все лишние разряды, получим округленное число

$$\tilde{a} = \pm 2^p \left(\frac{a_1}{2} + \frac{a_2}{2^2} + \ldots + \frac{a_t}{2^t}\right),$$

таким образом, для погрешности округления

$$a - \tilde{a} = \pm 2^p \left(\frac{a_{t+1}}{2^{t+1}} + \frac{a_{t+2}}{2^{t+2}} + \ldots\right)$$

справедлива оценка

$$|a-\tilde{a}| \le 2^p \frac{1}{2^{t+1}}\left(1 + \frac{1}{2} + \frac{1}{2^2} + \ldots\right) = 2^{p-t}.$$

Далее заметим, что из условия нормировки $|M| \ge 0{,}5$ (см. (3)) следует, что всегда $a_1 = 1$. Поэтому $|a| \ge 2^p \cdot 2^{-1} = 2^{p-1}$, и для относительной погрешности округления получим оценку

$$\frac{|a-\tilde{a}|}{|a|} \le 2^{-t+1}.$$

При более точных способах округления можно уменьшить погрешность по крайней мере в два раза и добиться, чтобы выполнялась оценка

$$\frac{|a-\tilde{a}|}{|a|} \le 2^{-t}. \qquad (4)$$

Итак, относительная точно в ЭВМ с плавающей запятой определяется числом разрядов t, отводимых для записи мантиссы. Можно считать, что точное значение числа a и отвечающее ему округленное число $\tilde{a}$ связаны равенством

$$\tilde{a} = a(1+\varepsilon), \tag{5}$$

где $|\varepsilon| \le 2^{-t}$. Число 2^{-t} называют иногда *машинным эпсилоном*. Оно характеризует относительную точность представления чисел в ЭВМ. Для рассмотренной ЭВМ имеем $t = 40,\ 2^{-t} \approx 10^{-12}$, т.е. относительная точность представления чисел составляет 12 десятичных знаков.

Соотношение (5) справедливо лишь в случае $|a| \ge M_0$, где M_0 – машинный нуль. Если же число *a* мало, а именно $|a| < M_0$, то полагается $\tilde{a} = 0$, что соответствует $\varepsilon = -1$ в формуле (5).

6 Накопление погрешностей округления

В процессе проведения вычислений погрешности округлений могут накапливаться, так как выполнение каждой из четырех математических операций вносит некоторую погрешность.

Будем в дальнейшем обозначать округленное в системе с плавающей запятой число, соответствующее точному числу *x*, через *fl*(*x*) (от английского floating – плавающий). Считается, что выполнение каждой арифметической операции вносит относительную погрешность, не большую, чем 2^{-t}. Это предположение можно записать в виде

$$fl(a * b) = a * b(1+\varepsilon), \tag{6}$$

где звездочка означает любую из операций $+,-,\times,:$, и $|\varepsilon| \le 2^{-t}$. Если результат выполнения арифметической операции является машинным нулем, то в формуле (6) надо положить $\varepsilon = -1$.

Может показаться, что предположение (6) не обосновано, так как согласно (6) каждое из чисел *a* и *b* записывается с относительной погрешность 2^{-t}, следовательно, погрешность результата может достигнуть 2^{-t+1}. Однако ЭВМ обладает возможностью проводить промежуточные вычисления с двойной точностью, т.е. с мантиссой, содержащей $2t$ разрядов, причем округлению до t разрядов подвергается лишь окончательный результат. Это обстоятельство позволяет добиться выполнения соотношения (6).

Для оценки влияния погрешностей округления на результат того или иного вычислительного алгоритма очень часто используется предположение о том, что *результат вычислений, искаженный погрешностями округления, совпадает с результатом точного выполнения этого же алгоритма, но с иными входными данными.*

Рассмотрим, например, процесс вычисления суммы

$$z = y_1 + y_2 + y_3 \tag{7}$$

трех положительных чисел. Пусть сначала находится сумма $y_1 + y_2$. Тогда согласно (7) получим

$$z_1 = fl(y_1 + y_2) = (y_1 + y_2)(1 + \varepsilon_1), \quad |\varepsilon_1| \le 2^{-t}.$$

Затем в результате сложения z_1 и y_3 получим число

$$\tilde{z} = fl(z_1 + y_3) = (z_1 + y_3)(1 + \varepsilon_2),$$

где $|\varepsilon_2| \le 2^{-t}$. Таким образом, вместо точного значения суммы z получаем приближенное значение

$$\tilde{z} = (y_1 + y_2)(1 + \varepsilon_1)(1 + \varepsilon_2) + y_3(1 + \varepsilon_2).$$

Отсюда видно, что результат выполнения алгоритма (7), искаженный погрешностями округления, совпадает с результатом точного выполнения того же алгоритма (7), примененного к другим исходным данным

$$\tilde{y}_i = (1 + \varepsilon_1)(1 + \varepsilon_2) y_i, \quad i = 1,2, \quad \tilde{y}_3 = (1 + \varepsilon_2) y_3.$$

На этом же примере видно, что результирующая погрешность зависит от порядка выполнения операций, так что вычисление суммы (7) в обратном порядке $(y_3 + y_2) + y_1$ может привести к другому результату.

Приведенный пример имеет чисто иллюстративное значение, так как число слагаемых в сумме (7) невелико, а погрешности ε_i малы. Практический интерес представляют оценки результирующих действий n. Однако прежде чем перейти к получению таких оценок, необходимо познакомиться с методами решения разностных уравнений.

7 Действия над приближенными числами

Найдем оценки погрешностей при выполнении некоторых операций над приближенными числами и для вычисления функций, аргументами которых являются приближенные числа. Более полным оказывается общее

правило, основанное на вычислении приращения (погрешности) функции при заданных приращениях (погрешностях) аргументов.

Рассмотрим функцию одной переменной $y = f(x)$. Пусть a – приближенное значение аргумента x, Δa – его абсолютная погрешность. Абсолютную погрешность функции можно считать ее приращением, которое можно заменить дифференциалом: $\Delta y \approx dy$. Тогда для оценки абсолютной погрешности получим выражение $\Delta y = |f'(a)|\Delta a$.

Аналогичное выражение можно записать для функции нескольких аргументов. Например, оценка абсолютной погрешности функции $u = f(x, y, z)$, приближенные значения аргументов которой соответственно a, b, c, имеет вид

$$\Delta u = |f'_x(a,b,c)|\Delta a + |f'_y(a,b,c)|\Delta b + |f'_z(a,b,c)|\Delta c. \qquad (8)$$

Здесь $\Delta a, \Delta b, \Delta c$ – абсолютные погрешности аргументов. Относительная погрешность находится по формуле

$$\delta u = \frac{\Delta u}{|f(a,b,c)|}. \qquad (9)$$

Полученные соотношения можно использовать для вывода оценки погрешности произвольной функции. Например, $c = a - b$ по формуле (8) получаем $\Delta c = |c'_a|\Delta a + |c'_b|\Delta b = \Delta a + \Delta b.$

Сформулируем правила оценки предельных погрешностей при выполнении операций над приближенными числами.

При сложении или вычитании чисел их абсолютные погрешности складываются. Относительная погрешность суммы заключена между наибольшим и наименьшим значением относительных погрешностей слагаемых; на практике принимается наибольшее значение.

При умножении или делении чисел друг на друга их относительные погрешности складываются. При возведении в степень приближенного числа его относительная погрешность умножается на показатель степени.

Для случая двух приближенных чисел a и b эти правила можно записать в виде формул:

$$\Delta(a \pm b) = \Delta a + \Delta b, \quad \delta(a \cdot b) = \delta a + \delta b,$$
$$\delta(a / b) = \delta a + \delta b, \quad \delta(a^k) = k\delta a.$$

Запишем выражение для относительной погрешности разности двух чисел в виде

$$\delta(a-b)=\frac{\Delta(a-b)}{|a-b|}=\frac{\Delta a+\Delta b}{|a-b|}.$$

При $a \approx b$ эта погрешность может быть сколь угодно большой.

При выполнении арифметических операций следует стремиться к тому, чтобы все исходные данные были примерно одинаковой точности. Сильное уточнение одних исходных данных при наличии больших погрешностей в других, как правило, не приводит к повышению точности результатов.

8 Итерационные методы решения систем нелинейных уравнений

8.1 Постановка задачи

Рассмотрим систему нелинейных уравнений

$$\begin{aligned} f_1(x_1,x_2,\ldots,x_m)&=0,\\ f_2(x_1,x_2,\ldots,x_m)&=0,\\ f_m(x_1,x_2,\ldots,x_m)&=0, \end{aligned} \qquad (10)$$

где f_i , $i=1,2,\ldots,m$ – функции вещественных переменных $x_1, \ldots, x_m$. В дальнейшем систему (10) будем рассматривать как операторное уравнение в некотором линейном пространстве **H** размерности m. Обозначим

$$\mathbf{x}=(x_1, x_2,\ldots,x_m)^T \in \mathbf{H},$$

$$\mathbf{F}(\mathbf{x})=(f_1(\mathbf{x}), f_2(\mathbf{x}),\ldots, f_m(\mathbf{x}))^{\mathrm{T}}$$

и запишем (10) в виде операторного уравнения

$$\mathbf{F}(\mathbf{x})=0, \qquad (11)$$

где $\mathbf{F}: \mathbf{H} \to \mathbf{H}$– отображение, нелинейное, вообще говоря, из **H** в **H**.

Нелинейные уравнения можно разделить на два класса – алгебраические и трансцендентные. *Алгебраическими* уравнениями называются уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные). В частности, многочлен является целой алгебраической функцией. Уравнения, содержащие другие функции (тригонометрические, показательные, логарифмические и др.), называются *трансцендентными*. Уже на примере алгебраического многочлена известно, что его корни могут быть как действительными так и комплексными. Поэтому более точная постановка задачи состоит в

нахождении корней уравнений, расположенных в заданной области комплексной плоскости. Можно рассматривать также задачу нахождения действительных корней, расположенных на заданном отрезке. Иногда, пренебрегая точностью формулировок, будем говорить, что требуется решить уравнение (11).

Методы решения нелинейных уравнений делятся на прямые и итерационные. *Прямые* методы позволяют записать корни в виде некоторого конечного соотношения (формулы). Из школьного курса алгебры известны такие методы для решения отдельных тригонометрических, логарифмических, показательных, а также, простейших алгебраических уравнений.

Однако встречающиеся на практике уравнения нельзя решить такими простыми методами. Для их решения используются *итерационные* методы, т.е. методы последовательных приближений.

В общем случае, когда нет информации о поведении $\mathbf{F}(\mathbf{x})$ алгоритм нахождения корней уравнения с помощью итерационного метода можно представить состоящим из двух этапов: 1) отыскания приближенного значения корня или содержащего его отрезка; 2) уточнения приближенного значения до некоторой заданной степени точности.

Приближенное значения корня, используемое в качестве начального приближения для построения итерационного процесса, можно, например, получить решив соответствующую линейную задачу. Для частного случая $m=1$, соответствующему одному уравнению, ниже рассмотрим прием отделения корней.

На втором этапе, используя заданное начальное приближение, строится итерационный процесс, позволяющий уточнить значение отыскиваемого корня. Таким образом, находится один из корней (11) наиболее близкий к выбранному начальному приближению. Чтобы отыскать другие корни, надо менять начальные приближения. Может оказаться, что и при других начальных данных метод сходится к тому же корню. Тогда целесообразно использовать прием выделения корней: если корень $x = x_*$ кратности *m* найден, то рассматривается функция

$$\mathbf{G}(\mathbf{x}) = \frac{\mathbf{F}(\mathbf{x})}{(\mathbf{x} - \mathbf{x}_*)^m}$$

и для неё повторяется процесс нахождения корня.

8.2 Метод отделения корней

Рассмотрим частный случай одного нелинейного уравнения с одним неизвестным, когда в (10) $m=1$.

Как указывалось выше, необходимо реализовать первый этап, в ходе которого провести разделение корней, т.е. выделить области, содержащие только один корень.

Не существует каких-то общих регулярных приемов решения задачи о расположении корней произвольной функции $f(x)$. Наиболее полно изучен вопрос о расположении корней алгебраических многочленов.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$$

Например, известно, что если для такого многочлена с действительными коэффициентами выполнены неравенства

$$f(c) > 0, f'(c) > 0, \ldots, f^{(m)}(c) > 0$$

то положительные корни $f(x)$ не превосходят числа *c*. Действительно из формулы Тейлора

$$f(x) = f(c) + (x-c) f'(c) + \frac{(x-c)^2}{2!} f''(c) + \ldots + \frac{(x-c)^m}{m!} f^{(m)}(c)$$

получаем, что $f(x) > 0$, при $x \geq c$.

Можно предложить простой прием отделения действительных корней уравнения (11). Предположим, что $f(x)$ определена и непрерывна на [*a*,*b*]. Вычислим таблицу значений функции $f(x)$ в заданных точках $x_k \in [a,b]$, $k = 0, 1, \ldots, n$. Число разбиений *n* не следует брать ни очень малым, т. к. тогда увеличивается вероятность пропустить расположенную рядом пару корней, ни очень большим, чтобы не увеличивать количество вычислений. Если обнаружится, что при некотором *k* числа $f(x_k)$, $f(x_{k+1})$ имеют разные знаки, то это будет означать, что на интервале $[x_k, x_{k+1}]$ уравнение (11) имеет по крайней мере один действительный корень, точнее, имеет нечетное число корней на $[x_k, x_{k+1}]$. В качестве начального приближения для реализации следующего этапа уточнения корня можно брать любые, точки принадлежащие найденному интервалу.

8.3 Метод дихотомии

По-прежнему рассматриваем частный случай одного нелинейного уравнения с одним неизвестным.

Начальным приближением для метода дихотомии или, как его еще называют, метода деления отрезка пополам является отрезок, на концах которого $f(x)$ имеет разные знаки. Такой отрезок может быть получен рассмотренным в 8.2 методом отделения корней.

Предположим, что на $[a,b]$ расположен лишь один корень x_* уравнения (11). Пусть для определенности $f(a) > 0$, $f(b) < 0$. Положим $x_0 = (a+b)/2$ и вычислим $f(x_0)$. Если $f(x_0) < 0$, то искомый корень находится на интервале $[a,x_0]$, если же $f(x_0) > 0$, то $x_* \in [x_0,b]$. Другими словами, из двух интервалов $[a,x_0]$ и $[x_0,b]$ выбираем тот, на границах которого функция $f(x)$ имеет различные знаки. Далее находим точку x_1 – середину выбранного интервала, вычисляем $f(x_1)$ и повторяем указанный процесс. В результате получаем последовательность интервалов, содержащих искомый корень x_*, причем длина каждого последующего интервала вдвое меньше, чем предыдущего. Процесс заканчивается, когда длина вновь полученного интервала станет меньше заданного числа $\varepsilon > 0$, и в качестве ее корня x_* приближенно принимается середина этого интервала.

Заметим, что если на $[a,b]$ имеется несколько корней, то указанный процесс сойдется к одному из корней, но заранее неизвестно, к какому именно.

К недостаткам метода дихотомии относят его медленную сходимость, скорость которой можно оценить как 2^n, где n – количество итераций.

Для более быстрого получения решения нелинейного уравнения можно использовать методы Ньютона, секущих и релаксации, рассмотренные ниже для систем уравнений.

8.4 Одношаговые итерационные методы

Рассмотрев в 8.2 и 8.3 простые методы отделения и уточнения корней для случая одного нелинейного уравнения с одним неизвестным, вернемся к общему случаю системы m уравнений.

Многие одношаговые итерационные методы для решения системы (11) можно записать в виде

$$\mathbf{B}_{k+1}\frac{\mathbf{x}^{k+1}-\mathbf{x}^{k}}{\boldsymbol{\tau}_{k+1}}+\mathbf{F}(\mathbf{x}^{k})=0, \qquad (12)$$

где k=0, 1,…– номер итерации,

$$\mathbf{x}^k = (x_1^k, x_2^k, ..., x_m^k)^T,$$

$\mathbf{x}^0$ – заданное начальное приближение, $\boldsymbol{\tau}_{k+1}$ – числовые параметры, $\mathbf{B}_{k+1}$ – матрица $m \times m$, имеющая обратную.

Для нахождения $\mathbf{x}^{k+1}$ по известному $\mathbf{x}^k$ из уравнения (12) необходимо решить систему линейных алгебраических уравнений

$$\mathbf{B}_{k+1}\mathbf{x}^{k+1} = \mathbf{g}(\mathbf{x}^k), \qquad (13)$$

где $\mathbf{g}(\mathbf{x}^k) = \mathbf{B}_{k+1}\mathbf{x}^k - \boldsymbol{\tau}_{k+1}\mathbf{F}\left(\mathbf{x}^k\right)$. Метод (12) называется явным, если $\mathbf{B}_{k+1}$=$\mathbf{E}$ для всех k=0, 1,... и неявным – в противном случае. Метод (12) называется стационарным, если $\mathbf{B}$ и $\boldsymbol{\tau}$ не зависят от номера итерации k. Систему линейных уравнений (13) можно решать либо прямым, либо итерационным методом. В последнем случае итерации, приводящие к решению системы (13), называются *внутренними итерациями,* а итерации (12) – *внешними итерациями.*

8.4.1 Метод релаксации

Метод релаксации представляет собой частный случай метода (12), когда $\mathbf{B}_{K+1} = \mathbf{E}$, $\boldsymbol{\tau}_{k+1} = \boldsymbol{\tau}$. Это стационарный итерационный метод, который можно записать в виде

$$\mathbf{x}^{k+1} = \mathbf{S}(\mathbf{x}^k),$$

где $\mathbf{S(x)} = \mathbf{x} - \boldsymbol{\tau}\mathbf{F(x)}$.

Метод сходится, если $\left\|\mathbf{S}'(\mathbf{x})\right\| < 1$. В данном случае $\mathbf{S}'(\mathbf{x}) = \mathbf{E} - \boldsymbol{\tau}\mathbf{F}'\left(\mathbf{x}\right)$ и

$$\mathbf{F}'\left(\mathbf{x}\right) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_m} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m(x)}{\partial x_1} & \frac{\partial f_m(x)}{\partial x_2} & \cdots & \frac{\partial f_m(x)}{\partial x_m} \end{bmatrix}. \qquad (14)$$

Рассмотренный метод применяется и для решения отдельных уравнений, при этом он носит название метода простых итераций.

8.4.2 Метод Пикара

Пусть **F**(**x**) представляется в виде

$$\mathbf{F}(\mathbf{x})=\mathbf{A}\mathbf{x}+\mathbf{G}(\mathbf{x}),$$

где **A** – матрица $m\times m$. Тогда итерации можно определить следующим образом:

$$\mathbf{A}\mathbf{x}^{k+1}+\mathbf{G}(\mathbf{x}^k)=0.$$

Итерационный метод можно переписать в виде

$$\mathbf{A}(\mathbf{x}^{k+1}-\mathbf{x}^k)+\mathbf{F}(\mathbf{x}^k)=0,$$

т.е. в канонической форме (12) $\mathbf{B}_{k+1}=\mathbf{A},\boldsymbol{\tau}_{k+1}=1$. Можно и здесь ввести итерационный параметр и рассматривать более общий метод

$$\mathbf{A}\frac{\mathbf{x}^{k+1}-\mathbf{x}^k}{\boldsymbol{\tau}}+\mathbf{F}(\mathbf{x}^k)=0.$$

8.4.3 Метод Ньютона

Метод Ньютона для систем уравнений (10) строится следующим образом. Пусть приближение $\mathbf{x}^k=(x_1^k,x_2^k,...,x_m^k)^T$ уже известно. Выпишем разложение функции $f_i(x_1,x_2,...,x_m)$ по формуле Тейлора в точке x^k

$$f_i(x_1,x_2,...,x_m)=f_i(x_1^k,x_2^k,...,x_m^k)+(x_1-x_1^k)\frac{\partial f_i(x^k)}{\partial x_1}+$$

$$+(x_2-x_2{}^k)\frac{\partial f_i(x^k)}{\partial x_2}+...+(x_m-x_m^k)\frac{\partial f_i(x^k)}{\partial x_m}+O\left(\left|x-x^k\right|^2\right)$$

и отбросим величины второго порядка малости. Тогда система (10) заменится системой уравнений

$$\sum_{j=1}^{m}(x_j-x_j^k)\frac{\partial f_i(x^k)}{\partial x_j}+f_i(x^k)=0\,,\ i=1,2,3,...,m, \qquad (15)$$

линейной относительно приращений $x_j-x^k_j$, $j=1,2,...,m$. Решение $x=(x_1,x_2,...,x_m)^T$ системы (15) примем за следующее приближение и обозначим через

$$\mathbf{x}^{k+1}=(x_1^{k+1},...,x_m^{k+1})^T.$$

Таким образом, итерационный метод Ньютона для (10) определяется системой уравнений

$$\sum_{j=1}^{m}(x_j^{k+1}-x_j^k)\frac{\partial f_i(x^k)}{\partial x_j}+f_i(x^k)=0,\ i=1,2,3,...,m, \qquad (16)$$

из которой последовательно, начиная с заданного $\mathbf{x}^0=(x_1^0,...,x_m^0)$, находятся векторы $\mathbf{x}^k$, k=1,2,... Систему (16) можно записать в векторном виде

$$\mathbf{F}'(\mathbf{x}^k)(\mathbf{x}^{k+1}-\mathbf{x}^k)+\mathbf{F}(\mathbf{x}^k)=0, k=0,1,..,$$

где $\mathbf{x}^0$ задан, матрица $\mathbf{F}'(\mathbf{x})$ определена согласно (14). Таким образом, метод Ньютона имеет канонический вид (12), где

$$\mathbf{B}_{k+1}=\mathbf{F}'(\mathbf{x}^k),\ \boldsymbol{\tau}_{k+1}=1.$$

Метод Ньютона называют также методом касательных, так как в одномерном случае ($m=1$) новое приближение x^{k+1} является абсциссой точек пересечения касательных, проведенных в точках $\left(x^k, f\left(x^k\right)\right)$ к графику функции $f(x)$, с осью *Ox*.

Таким образом, метод Ньютона требует на каждой итерации вычисления производных, для его реализации необходимо существование матриц $(\mathbf{F}'(\mathbf{x}^k))^{-1}$, обратных $\mathbf{F}'(\mathbf{x}^k)$. Нетрудно убедиться, что приведенное в 8.4.1 условие сходимости имеет для метода Ньютона абсолютный минимум $\mathbf{S}'(\mathbf{x})=0$, следовательно, метод Ньютона не просто сходится, а имеет наивысшую скорость сходимости. Следствием малости $\mathbf{S}'(\mathbf{x})$ вблизи корня является существенно более быстрая, чем у метода релаксации, квадратичная сходимость, т.е. погрешность на каждой следующей итерации пропорциональна квадрату погрешности на предыдущей. Все эти утверждения справедливы, если начальное приближение выбрано достаточно хорошо. Если начальное приближение выбрано неудачно, то метод может сходиться медленно, либо не сойдется вообще. Формула метода Ньютона работает и в комплексной плоскости, причем можно не прибегать в (16) к разделению действительных и мнимых частей для работы только с действительными числами.

Модифицированный метод Ньютона имеет вид

$$\mathbf{F}'(\mathbf{x}^0)(\mathbf{x}^{k+1}-\mathbf{x}^k)+\mathbf{F}(\mathbf{x}^k)=0 \qquad (17)$$

и обладает линейной сходимостью. Упрощение в численной реализации по сравнению с обычным методом Ньютона состоит в том, что матрицу $\mathbf{F}'(\mathbf{x})$ надо обращать не на каждой итерации, а лишь один раз. Возможно

циклическое применение модифицированного метода Ньютона, когда $\mathbf{F}'(\mathbf{x})$ обращается через определенное число итераций.

Метод Ньютона с параметром имеет вид

$$\mathbf{F}'(\mathbf{x}^k)\frac{\mathbf{x}^{k+1}-\mathbf{x}^k}{\boldsymbol{\tau}_{k+1}}+\mathbf{F}(\mathbf{x}^k)=0. \qquad (18)$$

Рассмотренные до сих пор методы являлись линейными относительно новой итерации $\mathbf{x}^{k+1}$. Возможны и нелинейные методы, когда для вычисления $\mathbf{x}^{k+1}$ приходится решать нелинейные системы уравнений.

8.4.4 Метод секущих

Этот метод получается из метода Ньютона заменой производных $\frac{\partial f_i(x^k)}{\partial x_j}$ разделенными разностями $\frac{f_i(x^k)-f_i(x^{k-1})}{x_j^k-x_j^{k-1}}$, вычисленным по известным значениям на двух предыдущих итерациях. В результате получаем итерационный метод

$$\sum_{j=1}^{m}(x_j^{k+1}-x_j^k)\frac{f_i(x^k)-f_i(x^{k-1})}{x_j^k-x_j^{k-1}}+f_i(x^k)=0\,,\ i=1,2,3,...,m, \qquad (19)$$

который в отличие от рассмотренных ранее методов является двухшаговым, т.е. новое приближение определяется двумя предыдущими итерациями. Соответственно необходимо задавать два начальных приближения – вектора $\mathbf{x}^0$ и $\mathbf{x}^1$.

Геометрическая интерпретация метода секущих явствует в одномерном случае $(m=1)$. Через точки $\left(x^{k-1},f\left(x^{k-1}\right)\right)$, $\left(x^k,f\left(x^k\right)\right)$ проводится прямая, абсцисса точки пересечения этой прямой с осью *Ox* и является новым приближением x^{k+1}.

8.4.5 Нелинейный метод Якоби

Нелинейный метод Якоби для системы (10) имеет вид

$$f_i(x_1^k,x_2^k,...,x_{i-1}^k,x_i^{k+1},x_{i+1}^k,..,x_m^k)=0, \qquad i=1,2,...,m \qquad (20)$$

Здесь для отыскания $\mathbf{x}^{k+1}$ необходимо решить m независимых скалярных уравнений. Для решения скалярного уравнения (внутренние итерации) можно применить какой-либо из итерационных методов, например, 8.4.1 или 8.4.3 при m=1, причем не обязательно применять один и тот же метод

для всех уравнений. Окончание внешних итераций определяется либо заданием максимального числа итераций $k_{\max}$, либо условием

$$\max_{1\le i\le m}\left|x_i^{k+1}-x_i^k\right|<\varepsilon,$$

где $\varepsilon>0$ – заданное число.

Метод Якоби применяется не только для решения систем нелинейных уравнений, а также и для решения СЛАУ.

8.4.6 Нелинейный метод Зейделя

Нелинейный метод Зейделя отличается от только что рассмотренного метода Якоби тем, что для вычисления x_i^{k+1} используются все уже вычисленные на этой же $(k+1)$-ой итерации значения предыдущих неизвестных и состоит в последовательном решении уравнений

$$f_i(x_1^{k+1},x_2^{k+1},...,x_i^{k+1},x_{i+1}^k,...,x_m^k)=0 \qquad (21)$$

относительно переменной x_i^{k+1}, i=1,2,...,m.

Все сказанное выше о методе Якоби в равной степени относится и к методу Зейделя.

8.4.7 Гибридные методы

Большое распространение получили гибридные методы, когда внешние итерации осуществляются одним методом, а внутренние – другим. При этом число внутренних итераций может быть фиксированным и не очень большим, так что внутренние итерации не доводятся до сходимости. В результате получается некоторый новый метод, сочетающий свойства исходных методов. Приведем примеры таких методов.

Внешние итерации – по Зейделю и внутренние – по Ньютону

Здесь в качестве основной (внешней) итерации выбирается нелинейный метод Зейделя (21), а для нахождения x_i^{k+1} используется

метод Ньютона. Обозначим $y_i = x_i^{k+1}$. Тогда итерации определяются следующим образом:

$$\frac{\partial f_i}{\partial x_i}(x_1^{k+1}, x_2^{k+1}, \ldots, x_{i-1}^{k+1}, y_i^S, x_{i+1}^k, \ldots, x_m^k)(y_i^{S+1} - y_i^S) + \Big/$$

$$+ f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, y_i^S, x_{i+1}^k, \ldots, x_m^k) = 0, \quad s = 0,1$$

$$y_i^0 = x_i^k, \quad y_i^{l+1} = y_i^{k+1}, \qquad i = 1,2,\ldots,m. \tag{22}$$

Здесь индексом *s* обозначен номер внутренней итерации.

Иногда в (22) делают всего одну внутреннюю итерацию, *l*=0, $y_i^0 = x_i^k, y_i^1 = x_i^{k+1}$. Тогда приходят к следующему итерационному методу:

$$\frac{\partial f_i}{\partial x_i}(x_1^{k+1}, x_2^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^k, x_{i+1}^k, \ldots, x_m^k)(x_i^{k+1} - x_i^k) +$$

$$+ f_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^k, x_{i+1}^k, \ldots, x_m^k) = 0, \quad k = 0,1,\ldots \tag{23}$$

В частности, при *m*=2, метод (23) принимает вид

$$\frac{\partial f_1(x_1^k, x_2^k)}{\partial x_1}(x_1^{k+1} - x_1^k) + f_1(x_1^k, x_2^k) = 0,$$

$$\frac{\partial f_2(x_1^{k+1}, x_2^k)}{\partial x_2}(x_2^{k+1} - x_2^k) + f_2(x_1^{k+1}, x_2^k) = 0. \tag{24}$$

Внешние итерации – по Ньютону и внутренние – по Зейделю

Запишем метод Ньютона для системы (11) в виде

$$\mathbf{F}'(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) + \mathbf{F}(\mathbf{x}^k) = 0, \tag{25}$$

где $\mathbf{F}'(\mathbf{x}^k) = (a_{ij}), a_{ij} = \dfrac{\partial f_i(x^k)}{\partial x_j}, i, j = 1,2,\ldots,m$. Для решения системы линейных уравнений (25) воспользуемся методом Зейделя. Для линейной системы

$$\mathbf{A}\boldsymbol{\omega} + \mathbf{F} = 0$$

метод Зейделя строится следующим образом. Матрица **A** представляется в виде суммы

$$\mathbf{A} = \mathbf{A}_- + \mathbf{D} + \mathbf{A}_+ ,$$

где матрицы $\mathbf{A}_+$, $\mathbf{A}_-$, **D** соответственно нижняя треугольная, верхняя треугольная и диагональная. Итерации метода Зейделя строятся по правилу

$$\left(\mathbf{A}_{-}+\mathbf{D}\right)\boldsymbol{\omega}^{S+1}+\mathbf{A}_{+}\boldsymbol{\omega}^{S}+\mathbf{F}=0,\ \ s=0,1,...,l, \tag{26}$$

и система (26) решается путем обращения нижней треугольной матрицы $\mathbf{A}_{-}+\mathbf{D}$.

В случае системы (24) надо положить $\mathbf{A}=\mathbf{F}'(\mathbf{x}^{k})$, вычислить последовательно векторы $\boldsymbol{\omega}^{S}$ согласно (26), начиная с $\boldsymbol{\omega}^{0}=0$, и положить $\boldsymbol{\omega}^{l+1}=\mathbf{x}^{k+1}-\mathbf{x}^{k}$, так что $\mathbf{x}^{k+1}=\mathbf{x}^{k}+\boldsymbol{\omega}^{l+1}$.

Заметим, что итерации по Зейделю можно осуществлять и относительно вектора $\mathbf{x}^{k+1}$.

Пусть в (26) совершается только одна итерация, т.е. l=0. Тогда, учитывая, что $\boldsymbol{\omega}^{0}=0,\boldsymbol{\omega}^{1}=\mathbf{x}^{k+1}-\mathbf{x}^{k}$, получим метод

$$(\mathbf{A}_{-}+\mathbf{D})(\mathbf{x}^{k+1}-\mathbf{x}^{k})+\mathbf{F}(\mathbf{x}^{k})=0, \tag{27}$$

где $\mathbf{A}_{-}+\mathbf{D}-$ «нижняя треугольная» часть матрицы Якоби (11), вычисленной при $\mathbf{x}=\mathbf{x}^{k}$.

В частности, при m=2 метод (27) принимает вид

$$\begin{aligned}&\frac{\partial f_1}{\partial x_1}(x_1^k,x_2^k)(x_1^{k+1}-x_1^k)+f_1(x_1^k,x_2^k)=0,\\&\frac{\partial f_2}{\partial x_1}(x_1^k,x_2^k)(x_1^{k+1}-x_1^k)+\frac{\partial f_2}{\partial x_2}(x_1^k,x_2^k)(x_2^{k+1}-x_2^k)+f_2(x_1^k,x_2^k)=0.\end{aligned} \tag{28}$$

Сопоставление (23) и (28) показывает, что методы, рассмотренные в последних двух примерах, не совпадают.

8.4.8 О роли ошибок округления в итерационных методах

Многие утверждения о сходимости итерационных процессов говорят о том, что решение поставленной задачи при определенных условиях может быть найдено этим процессом сколь угодно точно, причем погрешность каждого приближения может быть эффективно проконтролирована. Нетрудно понять, что все это справедливо на самом деле лишь до тех пор, пока на погрешность метода (остаточную погрешность) не наложится вычислительная погрешность (погрешность округлений), неизбежная при любых реальных компьютерных расчетах. Особенно существенное и даже пагубное влияние на результат решения задачи итерационным методом могут оказать ошибки округления в тех

случаях, когда утверждения о сходимости метода не содержат эффективных оценок погрешности.

Рассмотрим различие между реальным и идеальным итерационными процессами на простейшем объекте на методе простой итерации. Пусть на *k-м* итерационном шаге вычислений по методу (12) ошибки округлений составляют вектор $\boldsymbol{\gamma}^{(k)}$. Тогда в отличие от идеального метода (12), генерирующего последовательность приближений $\mathbf{x}^{(k)}$ к решению $\mathbf{x}^*$ системы (10), такому, что

$$\mathbf{x}^* = \mathbf{B}\mathbf{x}^* + \mathbf{c}, \tag{29}$$

реальный метод будет иметь вид

$$\tilde{\mathbf{x}}^{(\mathbf{k+1})} = \mathbf{B}\tilde{\mathbf{x}}^{(\mathbf{k})} + \mathbf{c} + \boldsymbol{\gamma}^{(k)}. \tag{30}$$

Изучим поведение векторов

$\boldsymbol{\mu}_{\mathbf{k}} = \tilde{\mathbf{x}}^{(\mathbf{k})} - \mathbf{x}^*$

– ошибок приближений $\tilde{\mathbf{x}}^{(k)}$, получаемых реальным методом (30). Вычитая (29) из (30), имеем

$$\tilde{\mathbf{x}}^{(k+1)} - \mathbf{x}^* = \mathbf{B}(\tilde{\mathbf{x}}^{(k)} - \mathbf{x}^*) + \boldsymbol{\gamma}^{(k)},$$

т.е.

$$\begin{aligned}
&\boldsymbol{\mu}_{k+1} = \mathbf{B}\boldsymbol{\mu}_k + \boldsymbol{\gamma}^{(k)} = \mathbf{B}(\mathbf{B}\boldsymbol{\mu}_{k-1} + \boldsymbol{\gamma}^{(k-1)}) + \boldsymbol{\gamma}^{(k)} = \\
&= \mathbf{B}^2(\mathbf{B}\boldsymbol{\mu}_{k-2} + \boldsymbol{\gamma}^{(k-2)}) + \mathbf{B}\boldsymbol{\gamma}^{(k-1)} + \boldsymbol{\gamma}^{(k)} = \ldots = \\
&= \mathbf{B}^{k+1}\boldsymbol{\mu}_0 + (\mathbf{B}^k\boldsymbol{\gamma}^{(0)} + \mathbf{B}^{k-1}\boldsymbol{\gamma}^{(1)} + \ldots + \mathbf{B}\boldsymbol{\gamma}^{(k-1)} + \boldsymbol{\gamma}^{(k)}).
\end{aligned} \tag{31}$$

Первое слагаемое в последнем выражении отвечает за погрешность идеального метода и может быть сделано сколь угодно малым в процессе итерирования при условии его сходимости. Чтобы оценить второе слагаемое, предположим, что порог абсолютных погрешностей округлений, допускаемых на каждой итерации, есть γ, т.е. $\left\|\boldsymbol{\gamma}^{(k)}\right\| \le \gamma$. Тогда

$$\left\|\mathbf{B}^k\boldsymbol{\gamma}^{(0)} + \mathbf{B}^{k-1}\boldsymbol{\gamma}^{(1)} + \ldots + \mathbf{B}\boldsymbol{\gamma}^{(k-1)} + \boldsymbol{\gamma}^{(k)}\right\| \le \left\|\mathbf{E} + \mathbf{B} + \ldots + \mathbf{B}^k\right\|,$$

и если $\|\mathbf{B}\|<q<1$, то второе слагаемое в (31), хотя и не стремится к нулю, но ограничено по норме величиной

$$\gamma\frac{1-q^k}{1-q} < \frac{\gamma}{1-q}.$$

При условиях теоретически обеспечивающих сходимость идеального метода (12), малость этого второго слагаемого отнюдь не гарантируется, что означает допустимость ситуаций, когда в ходе реальных итераций

погрешность округлений будет накапливаться вплоть до переполнения множества чисел, представляемых используемым компьютером.

Отметим, что роль ошибок округлений в образовании общей погрешности тем сильнее, чем медленнее сходимость итерационного процесса, и, что необходимо с осторожностью применять процессы, когда для них нет эффективных оценок погрешности, и, по возможности, учитывать влияние ошибок округления, если такие оценки есть.

9 Методы решения систем линейных алгебраических уравнений (СЛАУ)

Пусть имеется система линейных уравнений

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{32}$$

где **A** – матрица размера (n x n) с постоянными коэффициентами; **b** – n-мерный вектор известных констант и **x** – n-мерный вектор неизвестных:

$$\begin{bmatrix} a_{11} & a_{12} & . & . & a_{1n} \\ a_{21} & a_{22} & . & . & a_{2n} \\ . & & & & . \\ . & & & & . \\ a_{n1} & a_{n2} & . & . & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{bmatrix}. \tag{33}$$

Формально эту систему уравнений можно решить, обратив матрицу **A**:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{34}$$

Очень часто, когда требуется найти только одну «выходную» переменную, используют метод, называемый *правилом Крамера*. Это правило гласит, что для системы (32) k-я компонента x_k вектора **x** равна отношению определителя матрицы **A**, в которой k-й столбец заменен вектором **b**, и определителя матриц **A**:

$$x_k = \frac{\det(\textit{матрицы } \mathbf{A} \textit{ с } k\textit{-м столбцом, замененным на } \mathbf{b})}{\det(\mathbf{A})}. \tag{35}$$

Правило Крамера используется для решения уравнений низкого порядка и при теоретических исследованиях. Однако этот метод требует больших затрат машинного времени (порядка $m!$ арифметических действий) и редко применяется в вычислительных программах.

Обсудим также вопрос о корректности задачи (33). Как следует из определения (см. 3.2), для этого необходима устойчивость по исходным данным, которыми в данном случае являются коэффициенты матрицы и правая часть. Задача (33) устойчива по правой части если $\det \mathbf{A} \neq 0$ и чем ближе к нулю $\det \mathbf{A}$, тем сильнее погрешность правой части может исказить решение. Число $M_A = \left\|\mathbf{A}^{-1}\right\| \cdot \left\|\mathbf{A}\right\|$ называется *числом обусловленности* матрицы **A** и характеризует степень зависимости $\delta\mathbf{x}$ от $\delta\mathbf{b}$, оно не связано с каким-либо численным алгоритмом, а характеризует только свойства задачи (33). Если еще дополнительно учесть возмущение коэффициентов матрицы **A**, то будет справедлива полная оценка относительной погрешности решения:

$$\left\|\delta\mathbf{x}\right\| \le \frac{M_A}{1 - M_A \dfrac{\left\|\Delta\mathbf{A}\right\|}{\left\|\mathbf{A}\right\|}} \left(\frac{\left\|\Delta\mathbf{A}\right\|}{\left\|\mathbf{A}\right\|} + \frac{\left\|\Delta\mathbf{b}\right\|}{\left\|\mathbf{b}\right\|} \right).$$

Методы численного решения СЛАУ делятся на две группы: прямые методы и итерационные. В прямых методах, рассматриваемых в данном разделе, решение системы (33) находится за конечное число арифметических действий. Вследствие погрешностей округления при решении задач на ЭВМ прямые методы на самом деле не приводят к точному решению. На практике они применяются для матриц умеренного порядка ($10^2 \div 10^3$). Для матриц более высокого порядка целесообразнее применять итерационные методы, например, рассмотренные в 8.4.5 и 8.4.6 методы Якоби и Зейделя.

9.1. Метод исключения Гаусса

Численное решение СЛАУ часто производится методом исключения Гаусса, который является весьма простым и наглядным. В нем используется свойство СЛАУ, что сложение одного уравнения системы с другим, возможно умноженным на константу, не изменяет решения системы.

Рассмотрим матричное уравнение (33) и перепишем его в координатной форме, обозначив элементы b_i вектора **b** через $a_{i,n+1}$. Это упростит дальнейшее обозначения. Тогда система примет вид

$$\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & = & a_{1,n+1} \\
a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & = & a_{2,n+1} \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
a_{n1}x_1 & + & a_{n2}x_2 & + & \ldots & + & a_{nn}x_n & = & an_{,n+1}
\end{array} \qquad (36)$$

Разделим первое уравнение на a_{11} и запишем его:

$$x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + ... = a_{1,n+1}^{(1)},$$

где введено обозначение $a_{12}^{(1)} = a_{12} / a_{11}$ и т.д. Умножим это уравнение на $-a_{21}$ и сложим его со вторым уравнением. Коэффициенты вновь полученного второго уравнения $a_{2j}^{(1)} = a_{2j} - a_{21} a_{1j}^{(1)}, \quad j = 1,2,...,n+1.$ Такой выбор множителя обеспечивает равенство нулю коэффициента $a_{21}^{(1)}$. Аналогично для других уравнений подстановка

$$a_{ij}^{(1)} = a_{ij} - a_{i1} a_{1j}^{(1)}, \qquad \begin{matrix} i = 2,3,...,n. \\ j = 1,2,...,n+1, \end{matrix}$$

обеспечивает равенство нулю всех коэффициентов в первом столбце матрицы **A**, за исключением $a_{11}^{(1)}$, который равен 1. Фактически не нужно вычислять элемент, который становится нулевым. Элементы a_{j1} теперь не занимают память ЭВМ, и вычисления выполняются, начиная с $j = 2$. В результате этих операций уравнения примут вид

$$\begin{array}{ccccccccccc} x_1 & + & a_{12}^{(1)} x_2 & + & a_{12}^{(1)} x_3 & + & ... & + & a_{1n}^{(1)} x_n & = & a_{1,n+1}^{(1)} \\ & & a_{22}^{(1)} x_2 & + & a_{23}^{(1)} x_3 & + & ... & + & a_{2x}^{(1)} x_n & = & a_{2,n+1}^{(1)} \\ & & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ & & a_{n2}^{(1)} x_2 & + & a_{n3}^{(1)} x_3 & + & ... & + & a_{nn}^{(1)} x_n & = & a_{n,n+1}^{(1)} \end{array}$$

Индекс в скобках показывает, что коэффициенты были один раз изменены.

На следующем шаге исключим из рассмотрения первые строку и столбец и применим аналогично процедуру к уравнениям от второго до *n*-го. Запишем формулы для вычисления новых значений коэффициентов:

$$a_{2j}^{(2)} = a_{2j}^{(1)} / a_{22}^{(1)}; a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)} a_{2j}^{(2)}; i = 3,,4,...,n; j = 3,4,...,n+1.$$

Повторим процедуру для всех строк матрицы **A**. Если обозначить $a_{ij}^{(0)} = a_{ij},$ то общую формулу метода исключения Гаусса можно представить следующим образом:

$$\begin{matrix} a_{kj}^{(k)} = a_{kj}^{(k-j)} / a_{kk}^{(k-1)}; a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} a_{kj}^{(k)}; k = 1,2,...,n; \\ i = k+1, k+2,...,n; j = k+1, k+2,...,n+1. \end{matrix} \qquad (37)$$

В результате система уравнений приводится к виду

$$\begin{aligned} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + \ldots + a_{1n}^{(1)} x_n &= a_{1,n+1}^{(1)} \\ x_2 + a_{23}^{(2)} x_3 + \ldots + a_{2n}^{(2)} x_n &= a_{2,n+1}^{(2)} \\ x_3 + \ldots + a_{3n}^{(3)} x_n &= a_{3,n+1}^{(3)} \\ \ldots\ldots\ldots\ldots & \\ x_n = \quad a_{n,n+1}^{(n)} & \end{aligned} \qquad (38)$$

Для определения вектора неизвестных необходимо произвести *обратную подстановку.* Последняя составляющая x_n используется в $(n-1)$-м уравнении для определения x_{n-1} и т.д. В общем виде обратную подстановку можно записать так:

$$x_i = a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij} x_j, \quad i = n,\ n-1,\ldots,\ 1. \qquad (39)$$

Здесь верхние индексы опущены.

Из-за накопления погрешностей округления решение определяется с относительной погрешностью $\delta x = O\left(M_A \cdot n \cdot 2^{-t}\right)$, где t – число разрядов мантиссы.

В вычислительной математике принято оценивать эффективность алгоритмов числом *операций*, причем каждая операция представляет собой комбинацию умножения и вычитания. Можно показать, что исключение по Гауссу требует $\sim n^3/3$ операций, где n – порядок матрицы, а обратная подстановка может быть выполнена приблизительно за $n^2/2$ операций.

9.2 Метод LU-разложения

Лучшим методом решения СЛАУ является метод *разложения на треугольные матрицы,* или метод *LU-факторизации.* Алгоритмы этого метода близки к методу исключения Гаусса, хотя вычисления могут проводиться в различной последовательности. Главным преимуществом метода LU-факторизации в сравнении с методом исключения Гаусса является возможность более простого получения решений для различных

векторов **b** в правой части системы (32), а также для *транспонированной системы* уравнений, что требуется при расчете чувствительностей.

Допустим, что матрицу системы уравнений (32) можно разложить на два сомножителя:

$$\mathbf{A} = \mathbf{LU} \tag{40}$$

$$\mathbf{L} = \begin{bmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & 0 & \\ l_{31} & l_{32} & l_{33} & & \\ . & . & . & . & . \\ l_{n1} & l_{n2} & l_{n3} & & l_{nn} \end{bmatrix} \tag{41}$$

и

$$\mathbf{U} = \begin{bmatrix} 1 & u_{12} & u_{13} & \ldots & u_{1n} \\ & 1 & u_{23} & \ldots & u_{2n} \\ & & 1 & \ldots & u_{3n} \\ . & . & . & . & . \\ & 0 & & & 1 \end{bmatrix} \tag{42}$$

Здесь матрица **L** является *нижней треугольной,* а матрица **U** – *верхней треугольной.* Заметим, что на главной диагонали матрицы **U** стоят единицы. Это означает, что определитель матрицы **A** равен произведению диагональных элементов матрицы **L**.

Рассмотрим алгоритм определения матриц **L** и **U** позже, а сейчас допустим, что такое разложение осуществлено. Перепишем систему уравнений (32) в следующем виде:

$$\mathbf{LUx} = \mathbf{b}. \tag{43}$$

Определим вспомогательный вектор **z** как

$$\mathbf{Ux} = \mathbf{z}. \tag{44}$$

Из этого уравнения вектор z найти невозможно, поскольку неизвестен вектор **x.** Однако, подставив **z** в (43), получим:

$$\mathbf{Lz} = \mathbf{b}. \tag{45}$$

Благодаря специальной форме матрицы **L** вектор **z** можно легко определить. Для этого запишем (45) в виде системы уравнений:

$$\begin{gathered} l_{11}z_1 = b_1 \\ l_{21}z_1 + l_{22}z_2 = b_2 \\ l_{31}z_1 + l_{32}z_2 + l_{33}z_3 = b_3 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ l_{n1}z_1 + l_{n2}z_2 + \quad + l_{nn}z_n = b_n \end{gathered}$$

откуда получаем:

$$z_1 = b_1/l_{11}$$

$$z_i = (b_i - \sum_{j=1}^{i-1} l_{ij}z_j)/l_{ii}, i = 2,3,....n \qquad (46)$$

Этот процесс называют *прямым исключением (прямой подстановкой* или прямым ходом). Чтобы уравнение (46) имело смысл, диагональные элементы матрицы **L** должны быть ненулевыми. Теперь вернемся к (44) и найдем вектор неизвестных **x**. Для этого запишем (44) в координатной форме

$$\begin{gathered} x_1 + u_{12}x_2 + u_{13}x_3 + \ldots\ldots\ldots + u_{1n}x_n = z_1 \\ x_2 + u_{23}x_3 + \ldots\ldots\ldots + u_{2n}x_n = z_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_{n-1} + u_{n-1,n}x_n = z_{n-1} \\ x_n = z_n \end{gathered}$$

Начиная с последнего уравнения, можно последовательно найти компоненты вектора **x**. В общем виде они определяются по формуле

$$x_i = z_i - \sum_{j=i+1}^{n} u_{ij}x_j, i = n-1, n-2, n-3, \ldots 1 \qquad (47)$$

$$x_n = z_n$$

Этот процесс называют *обратной подстановкой* или обратным ходом.

Число операций, требуемых для выполнения как прямой, так и обратной подстановок, равно примерно $n^2/2$, а в сумме для решения требуется n^2 операций. Изучение уравнений (46) показывает, что компоненты b_i используются только для определения величин z_i и позднее не требуются. Аналогично в (47) величина z_i. не нужна после вычисления переменных x_i. Следовательно, при такой системе расчетов векторы **b, z,** и **x** могут быть размещены в одних и тех же ячейках памяти ЭВМ. Обратите

также внимание на эквивалентность обратной подстановки в уравнениях (47) и (39).

Займемся выводом алгоритма LU-разложения. С этой целью рассмотрим матрицу размера 4 X 4. Предполагая, что разложение существует, запишем произведение матриц **L** и **U**:

$$\begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} & l_{11}u_{14} \\ l_{21} & l_{21}u_{12}+l_{22} & l_{21}u_{13}+l_{22}u_{23} & l_{21}u_{14}+l_{22}u_{24} \\ l_{31} & l_{31}u_{12}+l_{32} & l_{31}u_{13}+l_{32}u_{23}+l_{33} & l_{31}u_{14}+l_{32}u_{24}+l_{33}u_{34} \\ l_{41} & l_{41}u_{12}+l_{42} & l_{41}u_{13}+l_{42}u_{23}+l_{43} & l_{41}u_{14}+l_{42}u_{24}+l_{43}u_{34}+l_{44} \end{bmatrix}$$

Сравним это произведение с матрицей **A**. Видно, что первый столбец разложения остается неизменным и $l_{i1}=a_{i1}$, i = =1,2,3,4. Заметим также, что первая строка произведения может быть использована для определения элементов первой строки матрицы **U** из решения уравнений $l_{11}u_{1j} =a_{1j}$, j = 2,3,4. Поскольку во втором столбце элементы u_{12} и l_{i1} известны, $l_{i2} = a_{i2} – l_{i1}u_{12}$, i =2, 3, 4. Так как теперь известны l_{21}, l_{22} и u_{1j}, можно использовать вторую строку матриц для расчета u_{2j}, j = 3,4:

$$u_{2j} = (a_{2j} – l_{21}u_{1j})/l_{22} , \quad j= 3, 4.$$

Далее, чередуя строки и столбцы, можно аналогичным образом найти остальные элементы матриц **L** и **U**.

Чтобы получить общие соотношения, запишем произвольный элемент произведения матриц **L** и **U**:

$$a_{ij} = \sum_{m=1}^{n} l_{im}u_{mj} = \sum_{m=1}^{\min(i,j)} l_{im}u_{mj},$$

где верхний предел суммы учитывает наличие нулевых элементов в матрицах **L** и **U**. Рассмотрим произвольный элемент на или под главной диагональю матрицы **A**, для которого $i \geq j$ к заменим индекс j на индекс k. При этом, положив u_{kk} = 1, получим

$$a_{ij} = \sum_{m=1}^{k} l_{im}u_{mk} = l_{ik} + \sum_{m=1}^{k-1} l_{im}u_{mk}$$

или

$$a_{kj} = \sum_{m=1}^{k} l_{km} u_{mj} = l_{kk} u_{kj} + \sum_{m=1}^{k-1} l_{km} u_{mj}.$$

Аналогичным образом, рассматривая произвольный элемент над главной диагональю, для которого $i < j$, и используя индекс k вместо i, находим

$$l_{ik} = a_{ik} - \sum_{m=1}^{k-1} l_{im} u_{mk}, \; i \geq k \qquad (48)$$

После преобразования придем к следующему выражению для элементов матрицы **U**:

$$u_{kj} = (a_{kj} - \sum_{m=1}^{k-1} l_{km} \; u_{mj}) \cdot \frac{1}{l_{kk}} \qquad j > k. \qquad (49)$$

Уравнения (48) и (49) описывают алгоритм разложения на треугольные матрицы, называемый *алгоритмом Краута*. Его выполнение осуществляется при задании k= 1, 2,... ,n и использовании формул (48) и (49). Заметим, что требуемые в этих соотношениях значения элементов матриц **L** и **U** рассчитываются на предыдущих этапах процесса. Далее, каждый элемент a_{ij} матрицы **A** требуется для вычисления только соответствующих элементов матриц **L** и **U**. Так как нулевые элементы матриц **L** и **U**, а также единичную диагональ матрицы **U** запоминать не нужно, в процессе вычислений матрицы **L** и **U** могут быть записаны на месте матрицы **A**, причем **L** расположена в нижнем треугольнике ($i \geq j$), а **U** – соответственно в верхнем треугольнике ($i < j$) матрицы **A**. Обобщив все вышесказанное, опишем алгоритм **LU**-разложения следующим образом:

Шаг 1. Положим k = 1 и перейдем к шагу 3.

Шаг 2. Используя (48), рассчитываем *k-й* столбец матрицы **L**. Если $k = n$, закончим процедуру разложения.

Шаг 3. Используя (49), рассчитываем *k-ю* строку матрицы **U**.

Шаг 4. Положим $k = k + 1$ и перейдем к шагу 2.

Можно получить другую форму алгоритма разложения, если последовательно *строку за строкой* рассматривать произведение матриц **L** и **U**. Вначале определяем первую строку матрицы **U**. Затем находим l_{22}, но расчет остальных элементов второго столбца матрицы **L** пока откладываем. Потом определяем вторую строку матрицы **U**, а затем элементы l_{32} l_{33} и третью строку матрицы **U** и т.д. При таком алгоритме можно рассматривать матрицу **A** построчно. Преимущество такого

подхода особенно проявляется при матрицах большого размера. Аналогичный алгоритм можно построить при последовательном переборе столбцов матрицы.

Еще одну форму алгоритма можно получить, если рассмотреть элементы матрицы **A**, разложенной на **L**- и **U**-сомножители. Представим это разложение в виде одной матрицы, где на месте нижней треугольной матрицы записана матрица **L**, а на месте верхней треугольной матрицы записаны элементы матрицы **U**, отличные от нуля и единицы. Эту матрицу можно представить в виде

$$\begin{bmatrix} l_{11} & a_{12}/l_{11} & a_{13}/l_{11} & a_{14}/l_{11} \\ l_{21} & a_{22}-l_{21}u_{12} & (a_{23}-l_{21}u_{13})/l_{22} & (a_{24}-l_{21}u_{14})/l_{22} \\ l_{31} & a_{32}-l_{31}u_{12} & a_{33}-l_{31}u_{13}-l_{32}u_{23} & (a_{34}-l_{31}u_{14}-l_{32}u_{24})/l_{33} \\ l_{41} & a_{42}-l_{41}l_{12} & a_{43}-l_{41}u_{13}-l_{42}u_{23} & a_{44}-l_{41}u_{14}-l_{42}u_{24}-l_{43}u_{34} \end{bmatrix}.$$

Первый столбец и строка этой матрицы рассчитываются, как и раньше. Далее заметим, что первое вычитание можно произвести сразу во всей матрице, поскольку все элементы l_{i1} и u_{1i} известны. После этого находим все элементы второго столбца, т.е. l_{i2}, $i = 2, 3,..., n$. Теперь, чтобы найти оставшиеся элементы второй строки, необходимо произвести деление на рассчитанное значение l_{22}. Эта процедура аналогична процедуре, выполненной вначале, за исключением того, что индекс теперь начинается с цифры 2. После этого в оставшихся элементах матрицы можно выполнить второе вычитание и т.д. Алгоритм разложения можно, следовательно, представить в виде:

Шаг 1. Положим $k = 1$.

Шаг 2. $l_{ik}=a_{ik}$, $i \geq k$.

Шаг 3. $u_{ki}=a_{ki}/l_{kk}$, $i>k$.

Шаг 4. $a_{ij}=a_{ij}-1_{ik}u_{kj}$, $i,j>k$.

Шаг 5. Если $k=n$, закончим процедуру разложения, в противном случае положим $k=k+1$ и перейдем к шагу 2.

Описанный алгоритм LU-разложения *эквивалентен* приведению матрицы к верхней треугольной форме с помощью процедуры Гаусса. Отличие состоит в том, что элементы под главной диагональю в треугольной матрице замещаются не нулями, а элементами матрицы **L**.

Для машинных приложений метод Краута или разложение по строкам или столбцам являются более предпочтительными в сравнении с методом исключения Гаусса. Причина этого заключается в том, что эти методы позволяют реализовать подпрограммы, реже обращающиеся к массивам, что сокращает время обращения к памяти, программировать

внутренние циклы на языке ассемблера и вычислять ряд переменных с двойной точностью.

Важные черты метода LU-разложения состоят в следующем:

1. Легко вычисляется определитель матрицы A

$$\det \mathrm{A} = \prod_{i=1}^{n} l_{ii}. \qquad (50)$$

2. Элементы матриц **L** и **U** могут быть записаны на месте элементов матрицы **A** и занесены в те же ячейки памяти (запоминать единичные элементы на главной диагонали матрицы **U** нет необходимости).

3. Если требуется найти решение для другого вектора **b** в правой части, то не нужно повторно проводить разложение матриц на треугольные, а достаточно только произвести прямую и обратную подстановки.

4. Для уравнений с транспонированной матрицей $\mathbf{A}^{t}\mathbf{x}=\mathbf{c}$ решение находится при том же LU-разложении. Анализ таких систем уравнений необходим при расчете чувствительностей.

Число операций *M,* требуемых для проведения LU-разложения, определяется как

$$M = \sum_{j=1}^{n-1}[(n-j)+(n-j^2)] = \frac{n^3}{3} - \frac{n}{3}.$$

С точки зрения объема вычислений метод LU-разложения вместе с прямой и обратной подстановками *эквивалентен* методу исключения Гаусса. Его преимущества заключаются в свойствах, указанных в третьем и четвертом пунктах.

Пример 1. Вычислим разложение следующей матрицы на треугольные сомножители:

$$\begin{bmatrix} 2 & 4 & -4 & 6 \\ 1 & 4 & 2 & 1 \\ 3 & 8 & 1 & 1 \\ 2 & 5 & 0 & 5 \end{bmatrix}.$$

Матрицы, получаемые на каждом шаге разложения при использования метода Краута, построчного разложения и метода Гаусса, приведены ниже:

Метод Краута	Разложение по строкам	Метод Гаусса
$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 4 & 2 & 1 \\ 3 & 8 & 1 & 1 \\ 2 & 5 & 0 & 5 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 4 & 2 & 1 \\ 3 & 8 & 1 & 1 \\ 2 & 5 & 0 & 5 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 4 & -2 \\ 3 & 2 & 7 & -8 \\ 2 & 1 & 4 & -1 \end{bmatrix}$
$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 1 & 1 \\ 2 & 1 & 0 & 5 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 8 & 1 & 1 \\ 2 & 5 & 0 & 5 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 3 & -6 \\ 2 & 1 & 2 & 0 \end{bmatrix}$
$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 3 & -2 \\ 2 & 1 & 2 & 5 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 3 & -2 \\ 2 & 5 & 0 & 5 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 3 & -2 \\ 2 & 1 & 2 & 4 \end{bmatrix}$
$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 3 & -2 \\ 2 & 1 & 2 & 4 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 & -2 & 3 \\ 1 & 2 & 2 & -1 \\ 3 & 2 & 3 & -2 \\ 2 & 1 & 2 & 4 \end{bmatrix}$	

Хотя матрицы на промежуточных шагах разложения различны, во всех случаях сомножители имеют вид

$$L = \begin{bmatrix} 2 & & & \\ 1 & 2 & & \\ 3 & 2 & 3 & \\ 2 & 1 & 2 & 4 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 2 & -2 & 3 \\ & 1 & 2 & -1 \\ & & 1 & -2 \\ & & & 1 \end{bmatrix}$$

Если цепь состоит только из пассивных элементов и независимых источников, то она обычно описывается системой уравнений с симметричной матрицей. В таких случаях матрицу можно разложить на сомножители вида $\mathbf{A}=\mathbf{U}^t\mathbf{D}\mathbf{U}$, где **U** – верхняя треугольная матрица, а **D** – диагональная матрица. При этом требуемые объем памяти и машинное время можно сократить примерно в 2 раза по сравнению с обычным LU-разложением.

9.3 Метод прогонки для СЛАУ с трехдиагональной матрицей

СЛАУ с трехдиагональной матрицей, т.е. с матрицей, все элементы которой, не лежащие на главной и двух побочных диагоналях, равны нулю ($a_{ij}=0$ при $j>i+1$ и $j<i-1$), представляет собой частный случай (32). Актуальность рассмотрения такого частного случая объясняется широким классом краевых задач для дифференциальных уравнений (см. 16.3), решение которых разностным методом сводится именно к таким СЛАУ.

В общем случае СЛАУ с трехдиагональной матрицей имеют вид

$$a_j y_{j-1} - c_j y_j + b_j y_{j+1} = -f_j, j = 1,2,...,N-1, \qquad (51)$$

$$y_0 = \chi_1 y_1 + \mu_1, y_N = \chi_2 y_{N-1} + \mu_2. \qquad (52)$$

Для численного решения систем с трехдиагональными матрицами применяется *метод прогонки,* который представляет собой вариант метода последовательного исключения неизвестных.

Приведем вывод расчетных формул метода прогонки. Будем искать решение системы (51) в виде

$$y_j = \alpha_{j+1} y_{j+1} + \beta_{j+1}, j = 0,1,...,N-1, \qquad (53)$$

где $\alpha_{j+1}, \beta_{j+1}$ – неизвестные пока коэффициенты. Отсюда найдем

$$y_{j-1} = \alpha_j y_j + \beta_j = \alpha_j(\alpha_{j+1} y_{j+1} + \beta_{j+1}) + \beta_j =$$
$$= \alpha_j \alpha_{j+1} y_{j+1} + (\alpha_j \beta_{j+1} + \beta_j), j = 1,2,...,N-1.$$

Подставляя полученные выражения для y_j, y_{j-1} в уравнения (51), приходим при $j = 1,2,...,N-1$ к уравнению

$$\left[\alpha_{j+1}(a_j \alpha_j - c_j + b_j)\right] y_{j+1} + \left[\beta_{j+1}(a_j \alpha_j - c_j) + a_j \beta_j + f_j\right] = 0.$$

Последнее уравнение будет выполнено, если коэффициенты $\alpha_{j+1}, \beta_{j+1}$ выбрать такими, чтобы выражения в квадратных скобках обращались в нуль. А именно, достаточно положить

$$\alpha_{j+1} = \frac{b_j}{c_j - \alpha_j a_j}, \beta_{j+1} = \frac{a_j \beta_j + f_j}{c_j - \alpha_j a_j}, j = 1,2,...,N-1. \qquad (54)$$

Соотношения (54) представляют собой нелинейные разностные уравнения первого порядка. Для их решения необходимо задать начальные

значения α_1, β_1. Эти начальные значения находим из требования эквивалентности условия (53) при *j=0*, т.е. условия $y_0 = \alpha_1 y_1 + \beta_1$, первому из уравнений (52). Таким образом, получаем

$$\alpha_1 = \chi_1, \beta_1 = \mu_1. \tag{55}$$

Нахождение коэффициентов $\alpha_{j+1}, \beta_{j+1}$ по формулам (54), (55) называется *прямой прогонкой*. После того как прогоночные коэффициенты $\alpha_{j+1}, \beta_{j+1}, j = 0,1,..., N-1$ найдены, решение системы (51),(52) находится по рекуррентной формуле (53), начиная с *j=N–1*. Для начала счета по этой формуле требуется знать y_N, которое определяется из уравнений

$$y_N = \chi_2 y_{N-1} + \mu_2, \;\; y_{N-1} = \alpha_N y_N + \beta_N$$

и оказывается равно

$$\frac{\chi_2 \beta_N + \mu_2}{1 - \chi_2 \alpha_N}. \tag{56}$$

Нахождение y_j по формулам (53), начиная с (56), называется *обратной прогонкой*. Алгоритм решения системы (51) и (52), определяемый по формулам (54)–(56), называется *методом прогонки*.

Метод прогонки можно применять, если знаменатели выражений (54), (56) не обращаются в нуль. Покожем, что для возможности применения метода прогонки достаточно потребовать, чтобы коэффициенты системы (51), (52) удовлетворяли условиям

$$a_j \neq 0, \;\; b_j \neq 0, \;\; |c_j| \geq |a_j| + |b_j|, j = 1,2,..., N-1, \tag{57}$$

$$|\chi_1| \leq 1, \;\; |\chi_2| < 1. \tag{58}$$

Заметим, что числа $a_j, b_j, c_j, \chi_1, \chi_2$ могут быть комплексными.

Сначала докажем по индукции, что при условиях (57), (58) модули прогоночных коэффициентов $\alpha_j, j = 1,2,..., N-1$ не превосходят единицы. Согласно (55), (58) имеем $|\alpha_1| = |\chi_1| \leq 1$. Предположим, что $|\alpha_j| \leq 1$ для некоторого j и докажем, что $|\alpha_{j+1}| \leq 1$. Из оценок $|c_j - \alpha_j a_j| \geq \|c_j| - |a_j\|\alpha_j\| \geq \|c_j| - |a_j\|$ и условий (57) получаем

$\left|c_j - \alpha_j a_j\right| \geq \left|b_j\right| > 0$, т.е. знаменатели выражений (54) не обращаются в нуль. Более того,

$$\left|\alpha_{j+1}\right| = \frac{\left|b_j\right|}{\left|c_j - \alpha_j a_j\right|} \leq 1.$$

Следовательно, $\left|\alpha_j\right| \leq 1, j = 1,2,...,N.$ Далее, учитывая второе из условий (58) и только что доказанное неравенство $\left|\alpha_N\right| \leq 1,$ имеем

$$\left|1 - \chi_2 \alpha_N\right| \geq 1 - \left|\chi_2\right| \cdot \left|\alpha_N\right| \geq 1 - \left|\chi_2\right| > 0,$$

т.е. не обращается в нуль и знаменатель выражения для y_N.

К аналогичному выводу можно прийти и в том случае, когда условия (57), (58) заменяются условиями

$$a_j \neq 0, \quad b_j \neq 0, \quad \left(c_j\right(> \left(a_j\right(+ \left(b_j\right(, j = 1,2,...,N-1, \tag{59}$$

$$\left|\chi_1\right| \leq 1, \quad \left|\chi_2\right| \leq 1. \tag{60}$$

В этом случае из предположения $\left|\alpha_j\right| \leq 1$ следует

$$\left|c_j - \alpha_j a_j\right| \geq \left|\left|c_j\right| - \left|a_j\right|\right| > \left|b_j\right|, \quad \left|\alpha_{j+1}\right| < 1,$$

т.е. все прогоночные коэффициенты, начиная со второго, по модулю строго меньше единицы. При этом

$$\left|1 - \chi_2 \alpha_N\right| \geq 1 - \left|\chi_2\right| \cdot \left|\alpha_N\right| \geq 1 - \left|\alpha_N\right| > 0.$$

Таким образом, при выполнении условий (57), (58) (так же как и условий (59), (60)) система (51)–(52) эквивалентна системе (54)–(56). Поэтому условия (57), (58) (или условия (59), (60)) гарантируют существование и единственность решения системы (51), (52) и возможность нахождения этого решения методом прогонки. Кроме того, доказанные неравенства $\left|\alpha_j\right| \leq 1, j = 1,2,...,N$ обеспечивают устойчивость счета по рекуррентным формулам (56). Последнее означает, что погрешность, внесенная на каком-либо шаге вычислений, не будет возрастать при переходе к следующим шагам. Действительно, пусть в формуле (56) при $j = j_0 + 1$ вместо y_{j+1} вычислена величина

$\tilde{y}_{j_0+1} = y_{j_0+1} + \delta_{j_0+1}$. Тогда на следующем шаге вычислений, т.е. при $j = j_0$, вместо $y_{j_0} = \alpha_{j_0+1} y_{j_0+1} + \beta_{j_0+1}$ получим величину

$$\tilde{y}_{j_0} = \alpha_{j_0+1}(y_{j_0+1} + \delta_{j_0+1}) + \beta_{j_0+1}$$

и погрешность окажется равной

$$\delta_{j_0} = \tilde{y}_{j_0} - y_{j_0} = \alpha_{j_0+1}\delta_{j_0+1}.$$

Отсюда получим, что $\left|\delta_{j_0}\right| \le \left|\alpha_{j_0+1}\right| \cdot \left|\delta_{j_0+1}\right| \le \left|\delta_{j_0+1}\right|$, т.е. погрешность не возрастает.

10 Интерполяция функций

Интерполяция состоит в следующем: для данной функции $y = f(x)$ строим многочлен $\varphi(x)$ степени m, принимающий в заданных точках x_i те же значения y_i, что и функция $f(x)$, т.е.

$$\varphi(x_i) = y_i, \quad i = 0,1,...,n. \tag{61}$$

При этом предполагается, что среди значений x_i нет одинаковых, т.е. $x_i \neq x_k$ при $i \neq k$. Точки x_i называются узлами интерполяции, многочлен $\varphi(x)$ – интерполяционным многочленом, равенство (61) – основным условием интерполяции.

Таким образом, близость интерполяционного многочлена к заданной функции состоит в том, что их значения совпадают на заданной системе n точек $x_0 \ldots x_n$ (рисунок 3, сплошная линия).

Рис. 3 - Интерполяция и аппроксимация

Максимальная степень интерполяционного многочлена $m = n$; в этом случае говорят о глобальной интерполяции, поскольку один многочлен

$$\varphi(x) = a_0 + a_1 x + ... + a_n x^n \qquad (62)$$

используется для интерполяции функции $f(x)$ на всем рассматриваемом интервале изменения аргумента x. Коэффициенты a_j многочлена (62) находятся из системы уравнений (61). Можно показать, что при $x_i \neq x_k$ ($i \neq k$) эта система имеет единственное решение. При большом количестве узлов интерполяции получается высокая степень многочлена (62).

Интерполяционные многочлены могут строиться отдельно для разных частей рассматриваемого интервала изменения x. В этом случае имеем *кусочную* (или *локальную*) интерполяцию.

Как правило, интерполяционные многочлены используются для представления функции в промежуточных точках между крайними узлами интерполяции, т.е. при $x_0 < x < x_n$. Однако иногда они используются и для приближенного вычисления функции вне рассматриваемого отрезка ($x < x_0$, $x > x_n$). Это приближение называют *экстраполяцией.*

Рассмотрим сначала случаи глобальной интерполяции, т.е. построение интерполяционного многочлена, единого для всего отрезка $[x_0, x_n]$. Существует несколько алгоритмов построения такого интерполяционного многочлена.

10.1 Канонический многочлен

Искомый многочлен записывается в виде (62). Из основного условия интерполяции получим следующую систему уравнений для нахождения коэффициентов $a_0, a_1, ..., a_n$:

$$\begin{aligned} a_0 + a_1 x_0 + ... + a_n x_0^n &= y_0, \\ a_0 + a_1 x_1 + ... + a_n x_1^n &= y_1, \\ &\cdots\cdots\cdots\cdots \\ a_0 + a_1 x_n + ... + a_n x_n^n &= y_n. \end{aligned} \qquad (63)$$

Можно показать, что эта система имеет единственное решение, если среди узлов интерполяции нет совпадающих, т.е. если $x_i \neq x_k$, при $k \neq i$. Решив эту систему, найдем коэффициенты интерполяционного многочлена

(62). Такой путь построения интерполяционного многочлена требует значительного объема вычислений, особенно при большом числе узлов. Существуют более экономичные алгоритмы построения интерполяционных многочленов.

10.2 Многочлен Лагранжа

Будем искать многочлен в виде линейной комбинации многочленов степени n:

$$L(x) = y_0 l_0(x) + y_1 l_1(x) + ... + y_n l_n(x). \qquad (64)$$

При этом потребуем, чтобы каждый многочлен $l_i(x)$ обращался в нуль во всех узлах интерполяции, за исключением одного $(i-го)$, где он должен равняться единице. Легко проверить, что этим условиям отвечает многочлен вида

$$l_0(x) = \frac{(x-x_1)(x-x_2)...(x-x_n)}{(x_0-x_1)(x_0-x_2)...(x_0-x_n)}. \qquad (65)$$

Действительно, $l_0(x_0) = 1$ при $x = x_0$. При $x = x_1,\ x_2,...,x_n$ числитель выражения (65) обращается в нуль. По аналогии с (65) получим

$$\begin{gathered} l_1(x) = \frac{(x-x_0)(x-x_2)...(x-x_n)}{(x_1-x_0)(x_1-x_2)...(x_1-x_n)}, \\ l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)...(x-x_n)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)...(x_2-x_n)}, \qquad (66) \\ ..... \\ l_i(x) = \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}. \end{gathered}$$

Подставляя в (64) выражения (65) и (66), находим

$$L(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}. \qquad (67)$$

Эта формула называется интерполяционным многочленом Лагранжа.

Покажем, что этот многочлен является единственным. Допустим противоположное: пусть существует еще один многочлен $F(x)$ степени n, принимающий в узлах интерполяции заданные значения, т.е. $F(x_i) = y_i$.

Тогда разность $R(x) = L(x) - F(x)$, являющаяся многочленом степени n (или ниже), в узлах x_i равна

$$R(x_i) = L(x_i) - F(x_i) = 0, \quad i = 0,1,...,n.$$

Это означает, что многочлен $R(x)$ степени не больше n имеет $n+1$ корней. Отсюда следует, что $R(x) = 0$ и $F(x) = L(x)$.

Из формулы (67) можно получить выражения для линейной $(n = 1)$ и квадратичной $(n = 2)$ интерполяций:

$$L(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_{11}, \quad n = 1;$$

$$L(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 + \\ + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2, \quad n = 2.$$

Существует несколько обобщений интерполяционного многочлена Лагранжа. Например, довольно широко используются *интерполяционные многочлены Эрмита.* Здесь наряду со значениями функции y_i в узлах x_i задаются значения ее производной y_i'. Задача состоит в том, чтобы найти многочлен $\varphi(x)$ степени $2n+1$, значения которого и его производной в узлах x_i удовлетворяют соответственно соотношениям

$$\varphi(x) = y_i, \quad \varphi'(x_i) = y_i', \quad i = 0,1,...,n.$$

В этом случае также существует единственное решение, если все x_i различны.

10.3 Многочлен Ньютона

До сих пор не делалось никаких предположений о законе распределения узлов интерполяции. Теперь рассмотрим случай равноотстоящих значений аргумента, т.е. $x_i - x_{i-1} = h = const \quad (i = 1,2,...,n)$. Величина h называется *шагом.*

Введем также понятие *конечных разностей.* Пусть известны значения функции в узлах $x_i : y_i = f(x_i)$. Составим разности значений функции:

$$\Delta y_0 = y_1 - y_0 = f(x_0 + h) - f(x_0),$$
$$\Delta y_1 = y_2 - y_1 = f(x_0 + 2h) - f(x_0 + h),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\Delta y_{n-1} = y_n - y_{n-1} = f(x_0 + nh) - f(x_0 + (n-1)h).$$

Эти значения называются *первыми разностями* (или *разностями первого порядка)* функции. Можно составить *вторые разности* функции:

$\Delta^2 y_0 = \Delta y_1 - \Delta y_{0,} \quad \Delta^2 y_1 = \Delta y_2 - \Delta y_1, \ldots$

Аналогично составляются разности порядка k:

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i, \quad i = 0,1,\ldots,n-1. \tag{68}$$

Конечные разности можно выразить непосредственно через значения функции. Например,

$\Delta^2 y_0 = \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0,$

$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0 = \ldots = y_3 - 3y_2 + 3y_1 - y_0.$

Аналогично для любого k можно написать

$$\Delta^k y_0 = y_k - k y_{k-1} + \frac{k(k-1)}{2!} y_{k-2} + \ldots + (-1)^k y_0. \tag{69}$$

Эту формулу можно записать и для значения разности в узле x_i:

$$\Delta^k y_i = y_{k+i} - k y_{k+i-1} + \frac{k(k-1)}{2!} y_{k+i-2} + \ldots + (-1)^k y_i. \tag{70}$$

Используя конечные разности, можно определить y_k:

$$y_k = y_0 + k\Delta y_0 + \frac{k(k-1)}{2!}\Delta^2 y_0 + \ldots + \Delta^k y_0. \tag{71}$$

Перейдем к построению интерполяционного многочлена Ньютона. Этот многочлен будем искать в следующем виде:

$$N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + \\ + a_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}). \qquad (72)$$

График многочлена должен проходить через заданные узлы, т.е. $N(x_i) = y_i \quad (i = 0,1,\ldots,n)$. Эти условия используем для нахождения коэффициентов многочлена:

$$N(x_0) = a_0 = y_0,$$
$$N(x_1) = a_0 + a_1(x_1 - x_0) = a_0 + a_1 h = y_1,$$
$$N(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) =$$
$$= a_0 + 2a_1 h + 2a_2 h^2 = y_2.$$

Найдем отсюда коэффициенты a_0, a_1, a_2:

$$a_0 = y_0,$$
$$a_1 = \frac{y_1 - a_0}{h} = \frac{y_1 - y_0}{h} = \frac{y_0}{h},$$
$$a_2 = \frac{y_2 - a_0 - 2a_1 h}{2h^2} = \frac{y_2 - y_0 - 2\Delta y_0}{2h^2} = \frac{\Delta^2 y_0}{2h^2}.$$

Аналогично можно найти и другие коэффициенты. Общая формула имеет вид

$$a_k = \frac{\Delta^k y_0}{k! h^k}, \quad k = 0,1,\ldots,n. \qquad (73)$$

Подставляя эти выражения в формулу (72), получаем следующий вид *интерполяционного многочлена Ньютона:*

$$N(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2! h^2}(x - x_0)(x - x_1) + \ldots + \\ + \frac{\Delta^n y_0}{n! h^n}(x - x_0)(x - x_1)\ldots(x - x_{n-1}). \qquad (74)$$

Конечные разности $\Delta^k y_0$ могут быть вычислены по формуле (69).

Формулу (74) часто записывают в другом виде. Для этого вводится переменная $t = (x - x_0)/h$; тогда

$$x = x_0 + th, \quad \frac{x - x_1}{h} = \frac{x - x_0 - h}{h} = t - 1,$$
$$\frac{x - x_2}{h} = t - 2, \ldots, \frac{x - x_{n-1}}{h} = t - n + 1.$$

С учетом этих соотношений формулу (74) можно переписать в виде

$$N(x_0 + th) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \ldots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0. \qquad (75)$$

Полученное выражение интерполирует данную функцию $y = f(x)$ на всем отрезке изменения аргумента $[x_0, x_n]$. Однако более целесообразно (с точки зрения повышения точности расчетов и уменьшения числа членов в (75)) ограничиться случаем $t < 1$, т.е. использовать формулу (75) для $x_0 \le x \le x_i$. Для других значений аргумента, например для $x_1 \le x \le x_2$, вместо x_0 лучше взять значение x_1. Таким образом, интерполяционный многочлен Ньютона можно записать в виде

$$N(x_i + th) = y_i + t\Delta y_i + \frac{t(t-1)}{2!}\Delta_2 y_i + \ldots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_i, \quad i = 0,1,\ldots \qquad (76)$$

Полученное выражение называется первым интерполяционным многочленом Ньютона для интерполирования вперед.

Интерполяционную формулу (76) обычно используют для вычисления значений функции в точках левой половины рассматриваемого отрезка. Это объясняется следующим. Разности $\Delta^k y_i$ вычисляются через значения функции $y_i, y_{i+1}, \ldots, y_{i+k},$, причем $i + k \le n$; поэтому при больших значениях i мы не можем вычислить разности высших порядков $(k \le n - i)$. Например, при $i = n - 3$ в (76) можно учесть только Δy, $\Delta^2 y$ и $\Delta^3 y$.

Для правой половины рассматриваемого отрезка разности лучше вычислять справа налево. В этом случае

$$t = \frac{(x - x_n)}{h,}, \qquad (77)$$

т.е. $t < 0$, и интерполяционный многочлен Ньютона можно получить в виде

$$N(x_n + th) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta^2 y_{n-2} + \ldots + \frac{t(t+1)\ldots(t+n-1)}{n!}\Delta^n y_0. \qquad (78)$$

Формула (78) называется вторым интерполяционным многочленом Ньютона для интерполирования назад.

Рассмотрим пример применения интерполяционной формулы Ньютона при ручном счете.

Пример. Вычислить в точках $x = 0.1, 0.9$ значения функции $y = f(x)$, заданной табл. 1. Каждая последующая конечная разность получается путем вычитания в предыдущей колонке верхней строки из нижней.

Таблица 1

x	y	Δy	Δ^2y	Δ^3y	Δ^4y	Δ^5y
0	1.2715	1.1937	–0.0146	0.0007	–0.0001	0.0000
0.2	2.4652	1.1791	–0.0139	0.006	–0.0001	
0.4	3.6443	1.1652	–0.0133	0.005		
0.6	4.8095	1.1519	–0.0128			
0.8	5.9614	1.1391				
1.0	7.1005					

При $x = 0.1$ имеем $t = (x - x_0)/h = (0.1 - 0)/0.2 = 0.5$. По формуле (75) получим

$$f(0.1) \approx N(0.1) = 1.2715 + 0.5 \cdot 1.1937 + \frac{0.5(0.5-1)}{2!} \cdot (-0.0146) +$$

$$+ \frac{0.5(0.5-1)(0.5-2)}{3!} \cdot 0.0007 + \frac{0.5(0.5-1)(0.5-2)(0.5-3)}{4!} \cdot (-0.0001) =$$

$$= 1.2715 + 0.59685 + 0.00182 + 0.00004 + 0.000004 = 1.8702.$$

Для сравнения по формуле линейной интерполяции получаем $f(0.1) \approx 1.8684$.

Значение функции в точке $x = 0.9$ нужно вычислять по формуле (78). В этом случае имеем

$t = (x - x_n)/h = (0.9 - 1)/0.2 = -0.5$.

Тогда

$$f(0.9) \approx N(0.9) 7.1005 - 0.5 \cdot 1.1391 - \frac{0.5(-0.5+1)}{2!} \cdot (-0.0128) -$$

$$- \frac{0.5(-0.5+1)(-0.5+2)}{3!} \cdot 0.0005 -$$

$$- \frac{0.5(-0.5+1)(-0.5+2)(-0.5+3)}{4!} \cdot (-0.0001) =$$

$$= 7.1005 - 0.5696 + 0.0016 - 0.00003 + 0.000004 = 6.5325.$$

Мы рассмотрели построение интерполяционного многочлена Ньютона для равноотстоящих узлов. Можно построить многочлен

Ньютона и для произвольно расположенных узлов, как и в случае многочлена Лагранжа. Однако этот случай мы рассматривать не будем.

В заключение отметим, что разные способы построения канонического многочлена, Лагранжа и Ньютона дают тождественные интерполяционные формулы при заданной таблице значений функции. Это следует из единственности интерполяционного многочлена заданной степени (при отсутствии совпадающих узлов интерполяции).

Интерполяционную формулу Ньютона удобнее применять в том случае, когда интерполируется одна и та же функция f(x), но число узлов интерполяции постепенно увеличивается. Если узлы интерполяции фиксированы и интерполируется не одна, а несколько функций, то удобнее пользоваться формулой Лагранжа.

Рассмотренная интерполяция алгебраическими многочленами является лишь одним из способов замены заданной функции. Однако не всякую функцию целесообразно приближать алгебраическими многочленами. Отметим в виде примеров несколько других способов интерполирования.

10.4 Приближение рациональными функциями

Интерполирующая функция записывается в следующем виде:

$$\varphi_{kl}(x) = \frac{a_k x^k + a_{k-1} x^{k-1} + \ldots + a_0}{x^l + b_{l-1} x^{l-1} + \ldots + b_0} \qquad (79)$$

(k, l – заданы).

Основное условие интерполяции даст систему из n+1 уравнения относительно $k + l$+1 неизвестного $a_0, a_1, \ldots, a_k, b_0, b_1, \ldots, b_{l-1}$. Будем требовать, чтобы число уравнений равнялось числу неизвестных, т.е. $n=k+l$. Тогда придем к системе линейных уравнений

$$\sum_{i=0}^{k} a_i x_j^i - f_j \sum_{i=0}^{l-1} b_i x_j^i = f_j x_j^l, \qquad j = 0,1,\ldots,n, \qquad (80)$$

в которой неизвестными являются величины a_i, i=0, 1,..., k, и b_i, i=0, 1,…, l–1.

Частным случаем (79) является дробно-линейная интерполяция, когда $k=l=1$. Пусть значения функции $f(x)$ заданы в трех узлах, а именно в точках x_{i-1}, x_i, x_{i+1}, причем $x_{i-1}<x_i<x_{i+1}$. Построим функцию

$$\varphi(x) = \frac{a_1 x + a_0}{x + b_0}, \tag{81}$$

для которой $\varphi(x_j) = f(x_j)$, $j=i-1,\ i,\ i+1$. Для определения коэффициентов a_0, a_1, b_0 можно воспользоваться системой уравнений (80).

10.5 Тригонометрическая интерполяция

Если $f(x)$ – периодическая функция с периодом l, то естественно строить приближения с помощью функций

$$\varphi_k(x) = a_k \cos\frac{\pi k x}{l} + b_k \sin\frac{\pi k x}{l}, \qquad k = 0,1,...,n.$$

Таким образом, *тригонометрическая интерполяция* состоит в замене $f(x)$ тригонометрическим многочленом

$$T_n(x) = \sum_{k=0}^{n} \varphi_k(x) = a_0 + \sum_{k=0}^{n} \ (a_k \cos\frac{\pi k x}{l} + b_k \sin\frac{\pi k x}{l}), \tag{82}$$

коэффициенты которого отыскиваются из системы уравнений

$$T_n(x_j) = f(x_j), \qquad j = 1,2,...,n+1, \tag{83}$$

где $x_0 < x_1 < \ldots < x_{2n+1}$, $x_{2n+1} - x = l$.

10.6 Точность глобальной интерполяции

График интерполяционного многочлена $y = \varphi(x)$ проходит через заданные точки, т.е. значения многочлена и данной функции $y = f(x)$ совпадают в узлах $x = x_i \ \ (i = 0,1,...,n)$. Если функция $f(x)$ сама является многочленом степени n, то имеет место тождественное совпадение: $f(x) = F(x)$. В общем случае в точках, отличных от узлов интерполяции, $R(x) = f(x) - F(x) \neq 0$. Эта разность есть погрешность интерполяции и называется *остаточным членом интерполяционной формулы*. Оценим его значение.

Предположим, что заданные числа y_i являются значениями некоторой функции $y = f(x)$ в точках $x = x_i$. Пусть эта функция непрерывна и имеет непрерывные производные до $n+1$-го порядка включительно. Можно показать, что в этом случае остаточный член интерполяционного многочлена Лагранжа имеет вид

$$R_L(x) = \frac{(x-x_0)(x-x_1)...(x-x_n)}{(n+1)!} f^{(n+1)}(x_*). \qquad (84)$$

Здесь $f^{(n+1)}(x_*)$ – производная $n+1$-го порядка функции $f(x)$ в некоторой точке $x = x_*, x_* \in [x_0, x_n]$. Если максимальное значение этой производной равно

$$\max_{x_0 \le x \le x_n} \left| f^{n+1}(x) \right| = M_{n+1},$$

то можно записать формулу для оценки остаточного члена:

$$\left| R_L(x) \right| \le \frac{(x-x_0)(x-x_1)...(x-x_n)}{(n+1)!} M_{n+1}.$$

Остаточный член интерполяционного Ньютона можно записать в виде

$$R_N(x) = \frac{t(t-1)...(t-n)}{(n+1)!} f^{(n+1)}(x_*) h^{n+1}, t = \frac{x-x_0}{h}.$$

Если предположить, что разность $\Delta^{n+1} y_n$ постоянна, то можно записать следующую формулу остаточного члена первой интерполяционной формулы Ньютона:

$$R_N(x) = \frac{t(t-1)...(t-n)}{(n+1)!} \Delta^{n+1} y_0. \qquad (85)$$

10.7 О сходимости интерполяционного процесса

Возникает вопрос, будет ли стремиться к нулю погрешность интерполяции $f(x)–L_n(x)$, если число узлов n неограниченно увеличивать. Ответ, вообще говоря, отрицательный.

Более общее утверждение содержится в теореме Фабера: какова бы ни была последовательность сеток Ω_n , найдется непрерывная на $[a,b]$ функция $f(x)$, такая, что последовательность интерполяционных многочленов $L_n[f(x)]$ не сходится к $f(x)$ равномерно на отрезке $[a,b]$.

Такого рода ситуацию в 1901 г. обнаружил К. Рунге. Он строил на отрезке $-1 \le x \le 1$ интерполяционные многочлены с равномерным распределением узлов для функции $y = 1/(1+25x^2)$. Оказалось, что при увеличении степени интерполяционного многочлена последовательность

его значений расходится для любой фиксированной точки x при $0.7 < |x| < 1$.

Для заданной непрерывной функции $f(x)$ можно добиться сходимости за счет выбора расположения узлов интерполяции (если они не зафиксированы). Справедлива теорема Марцинкевича: если $f(x)$ непрерывна на $[a,b]$, то найдется такая последовательность сеток, для которой соответствующий интерполяционный процесс сходится равномерно на $[a, b]$. Заметим, что построить такие сетки сложно и, кроме того, для каждой функции требуется своя сетка.

Доказано, что если функция $f(x)$ имеет непрерывную производную на отрезке $[-1,1]$, то при выборе значений x_i, совпадающих с корнями многочленов Чебышева степени $n+1$, интерполяционные многочлены степени n сходятся к значениям функции в любой точке этого отрезка.

Таким образом, повышение точности интерполяции целесообразно производить за счет уменьшения шага и специального расположения точек x_i. Повышение степени интерполяционного многочлена при локальной интерполяции также уменьшает погрешность, однако здесь не всегда ясно поведение производной $f^{(n+1)}(x)$ при увеличении n. Поэтому на практике избегают пользоваться интерполяционными многочленами высокой степени, стараются использовать многочлены малой степени (линейную и квадратичную интерполяции, сплайны).

10.8 Многочлены Чебышева

В ряде вопросов численного анализа, связанных с проблемой минимизации погрешности вычислительного алгоритма, нашли применение многочлены, наименее уклоняющиеся от нуля.

Среди всех многочленов $T_n(x)$ степени n со старшим коэффициентом 1 тот многочлен, для которого величина $\max\limits_{x\in[-1,1]}|T_n(x)|$ является минимальной, называется *многочленом, наименее уклоняющимся от нуля на отрезке* $[-1, 1]$, или *многочленом Чебышева*. Можно доказать, что функция

$$T_n(x) = 2^{1-n}\cos(n\arccos x) \qquad (86)$$

является многочленом Чебышева.

Рассмотрим функцию

$$P_n(x) = \cos(n\arccos x), \qquad (87)$$

которая отличается от $T_n(x)$ только постоянным множителем. Проводя преобразование

$$\cos((n+1)\arccos x)+\cos((n-1)\arccos x)=$$

$$=2\cos(n\arccos x)\cos(\arccos x)=2xP_n(x),$$

убеждаемся в том, что справедливо рекуррентное соотношение

$$P_{n+1}(x)-2xP_n(x)+P_{n-1}(x)=0. \qquad (88)$$

Кроме того, согласно (87) имеем $P_0(x) = 1$, $P_1(x)=x$. Отсюда и из (88) по индукции легко доказать, что $P_n(x)$ – многочлен степени n со старшим коэффициентом 2^{n-1}, n=1,2,... Следовательно, $T_n(x)$ – многочлен степени n со старшим коэффициентом 1.

З а м е ч а н и е. Для вещественных x правая часть выражения (86) определена только при $|x|\le 1$. Если $|x|\ge 1$, то многочлен $T_n(x)$ доопределяется формулой

$$T_n(x)=2^{-n}((x+\sqrt{x^2-1})^n+(x-\sqrt{x^2-1})^n).$$

Возможность такого доопределения объясняется тем, что для любого комплексного числа ***z*** справедливо тождество

$$\cos(n\arccos z)=0,5((z+\sqrt{z^2-1})^n+(z-\sqrt{z^2-1})^n).$$

Корни многочлена $T_n(x)$ расположены в точках

$$x_k=\cos\frac{(2k+1)\pi}{2n},\quad k=0,1,...,n-1, \qquad (89)$$

а экстремумы – в точках

$$x'_k=\cos\frac{k\pi}{n},\qquad k=0,1,...,n \qquad (90)$$

причем

$$T_n(x'_k)=(-1)^k 2^{1-n},\quad k=0,1,...,n \qquad (91)$$

Следовательно,

$$\max_{x\in[-1,1]}|T_n(x)|=2^{1-n}. \qquad (92)$$

Иногда требуется найти многочлен, наименее уклоняющийся от нуля на заданном отрезке [*a*,*b*], среди всех многочленов степени *n* со старшим коэффициентом 1. Эта задача сводится к предыдущей с помощью замены

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

переводящей отрезок $a \le x \le b$ в отрезок $-1 \le t \le 1$. При такой замене многочлен Чебышева (86) преобразуется к виду

$$F_n(x) = 2^{1-n}\cos\left[n\arccos\left(2x - \frac{b+a}{b-a}\right)\right],$$

причем коэффициент при x^n оказывается равным $2^n/(b-a)^n$. Следовательно, многочленом, наименее уклоняющимся от нуля на [*a*,*b*], среди всех многочленов степени *n* со старшим коэффициентом 1 является многочлен

$$T_n(x) = \frac{(b-a)^n}{2^{2n-1}}\cos\left[n\arccos\left(\frac{2x-(b+a)}{b-a}\right)\right]. \qquad (93)$$

Корни этого многочлена расположены в точках

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{(2k+1)\pi}{2n}, \quad k = 0,1,...,n-1, \qquad (94)$$

а его максимальное отклонение от нуля равно

$$\max_{x\in[a,b]}|T_n(x)| = \frac{(b-a)^n}{2^{2n-1}}. \qquad (95)$$

В теории итерационных методов возникает следующая задача: найти многочлен $P_n(x)$ степени *n*, наименее уклоняющийся от нуля на [*a*,*b*], среди всех многочленов степени *n*, принимающих при $x=0$ значение 1. Ясно, что искомый многочлен отличается от многочлена (93) только нормировкой, т.е.

$$P_n(x) = \frac{T_n(x)}{T_n(0)}. \qquad (96)$$

Будем считать в дальнейшем, что $T_n(0) \neq 0$.

Многочлены Чебышева применяются при интерполяции. В 10.6 было показано, что остаточный член интерполяционного многочлена Лагранжа (84) зависит от расположения узлов интерполяции. Пусть требуется так подобрать числа x_k (среди которых нет совпадающих чисел), принадлежащие заданному отрезку $[a,b]$, чтобы минимизировать величину $\max\limits_{x\in[a,b]}\left|R_L(x).\right|$

Поскольку старший коэффициент многочлена $R_L(x)$ равен 1, для решения данной задачи достаточно потребовать, чтобы $R_L(x)$ совпал с многочленом Чебышева

$$T_{n+1}(x)=\frac{(b-a)^{n+1}}{2^{2n+1}}\cos\left[(n+1)\arccos\frac{2x-(b+a)}{b-a}\right]$$

(см. (93)). Условие $\left|R_L(x)\right|\equiv\left|T_{n+1}(x)\right|$ будет выполнено тогда и только тогда, когда совпадут все корни многочленов ω(x) и $T_{n+1}(x)$. Корнями многочлена $R_L(x)$ являются числа $x_0,x_1,\ldots,x_n$, а корни $T_{n+1}(x)$ определяются согласно (81) формулами

$$x_k=\frac{a+b}{2}+\frac{b-a}{2}\cos\frac{(2k+1)\pi}{2(n+1)},\ k=0,1,...,n, \qquad (97)$$

Точки x_k, расположенные по правилу (97), называются *чебышевскими узлами интерполяции*, величина отклонения многочлена $R_L(x)$ от нуля при этом оказывается минимальной и равной

$$\max_{x\in[a,b]}\left|f(x)-L_n(x)\right|\le\frac{M_{n+1}}{(n+1)!}\frac{(b-a)^{n+1}}{2^{2n+1}}. \qquad (98)$$

Оценка (98) называется наилучшей равномерной оценкой погрешности интерполяции.

Э*кономизация степенных рядов* – еще одно применение многочленов Чебышева. В математическом анализе хорошо изучено и широко применяется разложение функций в степенные ряды, в частности, в ряды Тейлора. Частичные суммы таких рядов – многочлены – используются в качестве локальных аппроксимаций для исходных функций. Степени используемых при этом многочленов зависят от требуемой точности аппроксимации, положения точки из области сходимости ряда, в окрестности которой производится аппроксимация, скорости сходимости ряда. В некоторых случаях такой подход мало приемлем из-за медленной

сходимости рядов и большой неравномерности, т.е. существенной разницы в необходимых для заданной точности степенях приближающих многочленов при разных значениях аргумента.

Для улучшения указанных параметров частичных сумм степенных рядов можно привлечь многочлены Чебышева. *Процедура такого преобразования и называется экономизацией степенного ряда.* Приведем формулы, по которым многочлены Чебышева $T_n(x)$ выражаются через степенные функции:

$$\begin{aligned} 1 &= T_0; \\ x &= T_1; \\ x^2 &= \frac{1}{2}(T_0 - T_2); \\ x^3 &= \frac{1}{4}(3T_1 + T_3); \\ x^4 &= \frac{1}{8}(3T_0 + 4T_2 + T_4); \\ x^5 &= \frac{1}{16}(10T_1 + 5T_3 + T_5) \end{aligned} \qquad (99)$$

и т. д. (аргумент x в этих выражениях для краткости опущен). В результате получается разложение $f(x)$ вида $f(x) = b_0 + b_1T_1 + b_2T_2 + ... + b_nT_n + ...$

10.9. Интерполяция сплайнами

Интерполяция многочленом Лагранжа или Ньютона на всем отрезке $[a,b]$ с использованием большого числа узлов интерполяции часто приводит к плохому приближению, что объясняется сильным накоплением погрешностей в процессе вычислений. Кроме того, из-за расходимости процесса интерполяции увеличение числа узлов не обязано приводить к повышению точности. Для того чтобы избежать больших погрешностей, весь отрезок $[a,b]$ разбивают на частичные отрезки и на каждом из частичных отрезков приближенно заменяют функцию *f(x)* многочленом невысокой степени (так называемая *кусочно-полиномиальная интерполяция).*

Одним из способов интерполирования на всем отрезке является интерполирование с помощью сплайн-функций. Сплайн-функцией или сплайном называют кусочно-полиномиальную функцию, определенную на

отрезке $[a,b]$ и имеющую на этом отрезке некоторое число непрерывных производных.

Слово «сплайн» (английское spline) означает гибкую линейку, используемую для проведения гладких кривых через заданные точки плоскости. Мы не будем придавать слову «сплайн» какого-либо определенного технического смысла.

Преимуществом сплайнов перед обычной интерполяцией является, во-первых, их сходимость и, во-вторых, устойчивость процесса вычислений.

Далее рассмотрим частный, но распространенный в вычислительной практике случай, когда сплайн определяется с помощью многочленов третьей степени (кубический сплайн).

Сеткой на отрезке $[a,b]$ называется любое конечное множество точек этого отрезка. Функция, определенная в точках сетки, называется *сеточной функцией.* Будем обозначать через ω_N сетку, удовлетворяющую условиям

$$a = x_0 < x_1 < x_2 < \ldots < x_{N-1} < x_N = b, \qquad (100)$$

и через f_i – значение сеточной функции f(x) в точке $x_i \in \omega_N$, т.е. $f_i = f(x_i)$. Точки $x_i \in \omega_N$ называются *узлами сетки* ω_N. Равномерной сеткой на $[a,b]$ называется множество точек

$$\omega_h = \{x_i = a + ih, i = 0,1,\ldots,N\}, \qquad (101)$$

где *h=(b–a)/N – шаг сетки*.

Пусть на $[a,b]$ задана непрерывная функция *f(x).* Введем согласно (100) сетку ω_N и обозначим $f_i = f(x_i),\ i = 0,1,...,N.$

Сплайном, соответствующим данной функции *f(x)* и данным узлам $\{x_i\}_{i=0}^{N}$, называется функция *s(x),* удовлетворяющая следующим условиям:

а) на каждом сегменте $[x_{i-1},x_i], i = 1,2,...,N,$ функция *s(x)* является многочленом третьей степени;

б) функция *s(x),* а также ее первая и вторая производные непрерывны на $[a,b]$;

в) $s(x_i) = f(x_i), i = 0,1,...,N.$

Сплайн, определяемый условиями а) – в), называется также *интерполяционным кубическим сплайном.*

Докажем существование и единственность сплайна, определяемого перечисленными условиями. Приведенное ниже доказательство содержит также способ построения сплайна.

На каждом из отрезков $[x_{i-1}, x_i], i = 1,2,...,N$, будем искать функцию $s(x) = s_i(x)$ в виде многочлена третьей степени

$$s_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3, \quad x_{i-1} \le x \le x_i, i = 1,2,...,N, \tag{102}$$

где a_i, b_i, c_i, d_i – коэффициенты, подлежащие определению. Поясним смысл введенных коэффициентов. Имеем

$$s_i' = b_i + c_i(x - x_i) + \frac{d_i}{2}(x - x_i)^2,$$
$$s_i''(x) = c_i + d_i(x - x_i),$$
$$s_i'''(x) = d_i,$$

поэтому

$$a_i = s_i(x_i), \quad b_i = s_i'(x_i), \quad c_i = s_i''(x_i), \quad d_i = s_i'''(x_i). \tag{103}$$

Из условий интерполирования $s(x_i) = f(x_i), \ i = 1,2,...,N,$ получаем, что

$$a_i = f(x_i), \quad i = 1,2,...,N. \tag{104}$$

Доопределим, кроме того, $a_0 = f(x_0)$.

Далее, требование непрерывности функции *s(x)* приводит к условиям $s_i(x_i) = s_{i+1}(x_i), \quad i = 1,2,...,N-1.$

Отсюда, учитывая выражения для функций $s_i(x)$, получаем при $i = 0,1,...,N-1$ уравнения

$$a_i = a_{i+1} + b_{i+1}(x_i - x_{i+1}) + \frac{c_{i+1}}{2}(x_i - x_{i+1})^2 + \frac{d_{i+1}}{6}(x_i - x_{i+1})^3.$$

Обозначая $h_i = x_i - x_{i-1,}$ перепишем эти уравнения в виде

$$h_i b_i - \frac{h_i^2}{2}c_i + \frac{h_i^3}{6}d_i = f_i - f_{i-1}, \quad i = 1,2,...,N. \tag{105}$$

Условия непрерывности первой производной

$$s_i'(x_i) = s_{i+1}'(x_i), \quad i = 1,2,...,N-1$$

приводят к уравнениям

$$c_i h_i - \frac{d_i}{2}h_i^2 = b_i - b_{i-1}, \quad i = 2,3,...,N. \tag{106}$$

Из условия непрерывности второй производной получаем уравнения

$$d_i h_i = c_i - c_{i-1}, \quad i = 2,3,...,N. \tag{107}$$

Объединяя (105) – (108), получим систему $3N-2$ уравнений относительно $3N$ неизвестных $b_i, c_i, d_i, \ i = 1,2,...,N$. Два недостающих уравнения получают, задавая те или иные граничные условия для *s(x)*. Предположим, например, что функция *f(x)* удовлетворяет условиям $f''(a) = f''(b) = 0$. Тогда естественно требовать, чтобы $s''(a) = s''(b) = 0$. Отсюда получаем $s_i''(x_0) = 0, \ \ s_N''(x_N) = 0,$ т.е. $c_1 - d_1 h_1 = 0, c_N = 0$. Заметим, что условие $c_1 - d_1 h_1 = 0$ совпадает с уравнением (107) при $i = 1$, если положить $c_0 = 0$:

$$h_i d_i = c_i - c_{i-1}, i = 1,2,...,N, \quad c_0 = c_N = 0, \tag{108}$$

Таким образом, получена замкнутая система уравнений (105), (106) и (108) для определения коэффициентов кубического сплайна.

Убедимся в том, что эта система имеет единственное решение. Исключим переменные $b_i, d_i, \ \ i = 1,2,...,N-1$ и получим систему, содержащую только c_i, $i = 1,2,...,N-1$. Для этого рассмотрим два соседних уравнения (105):

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

$$b_{i-1} = \frac{h_{i-1}}{2} c_{i-1} - \frac{h_{i-1}^2}{6} d_{i-1} + \frac{f_{i-1} - f_{i-2}}{h_{i-1}}$$

и вычтем второе уравнение из первого. Тогда получим

$$b_i - b_{i-1} = \frac{1}{2}(h_i c_i - h_{i-1} c_{i-1}) - \frac{1}{6}(h_i^2 d_i - h_{i-1}^2 d_{i-1}) + \frac{f_i - f_{i-1}}{h_i} - \frac{f_{i-1} - f_{i-2}}{h_{i-1}}.$$

Подставляя найденное выражение для $b_i - b_{i-1}$ в правую часть уравнения (106), получим

$$h_i c_i + h_{i-1} c_{i-1} - \frac{h_{i-1}^2}{3} d_{i-1} - \frac{2h_i^2}{3} d_i = 2\left(\frac{f_i - f_{i-1}}{h_i} - \frac{f_{i-1} - f_{i-2}}{h_{i-1}} \right). \tag{109}$$

Далее, из уравнения (108) получаем

$$h_i^2 d_i = h_i(c_i - c_{i-1}), \quad h_{i-1}^2 d_{i-1} = h_{i-1}(c_{i-1} - c_{i-2})$$

и, подставляя эти выражения в (109), приходим к уравнению

$$h_{i-1}c_{i-2} + 2(h_{i-1} + h_i)c_{i-1} + h_i c_i = 6\left(\frac{f_i - f_{i-1}}{h_i} - \frac{f_{i-1} - f_{i-2}}{h_{i-1}} \right).$$

Окончательно для определения коэффициентов c, получаем систему уравнений

$$h_i c_{i-1} + 2(h_i + h_{i+1})c_i + h_{i+1}c_{i+1} = 6\left(\frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \right), \qquad (110)$$

$$i = 1,2,...,N-1, \quad c_0 = c_N = 0.$$

В силу диагонального преобладания система (110) имеет единственное решение. Так как матрица системы трехдиагональная, решение легко найти методом прогонки, которая в данном случае устойчива. По найденным коэффициентам c_i коэффициенты b_i и d_i определяются с помощью явных формул

$$d_i = \frac{c_i - c_{i-1}}{h_i}, \quad b_i = \frac{h_i}{2}c_i - \frac{h_i^2}{6}d_i + \frac{f_i - f_{i-1}}{h_i}, \quad i = 1,2,...,N. \qquad (111)$$

Таким образом, доказано, что существует единственный кубический сплайн, определяемый условиями а) – в) и граничными условиями $s''(a) = s''(b) = 0$. Заметим, что можно рассматривать и другие граничные условия.

Итак, при интерполяции основным условием является прохождение графика интерполяционного многочлена через данные значения функции в узлах интерполяции. Однако в ряде случаев выполнение этого условия затруднительно или даже нецелесообразно, например, если исходная таблица значений функции получена в результате эксперимента и неизбежно уже содержит измерительную погрешность.

11 Аппроксимация функций

Если не требуется строгого выполнения условия прохождения функции через все узловые точки, а приоритет отдается простой и компактной записи функции, то для такого приближения используют аппроксимацию – наилучшее приближение функции, заданной таблично.

Пусть даны значения функции $f(x)$ в точках $x_k \in [a,b]$, $k = =0,1,....,n$. Аппроксимация состоит в выборе вида функции $\varphi(x)$ и определении ее коэффициентов исходя из критерия наилучшего приближения в точках x_k к $f(x)$. В качестве аппроксимирующей функции можно выбрать любую, качественное поведение которой напоминает $f(x)$. В качестве универсальных функций используются ортогональные полиномы степени $m<n$.

Введем обобщенный многочлен и будем рассматривать его значения только в узлах x_k, т.е.

$$\varphi(x_k) = c_0\varphi_0(x_k) + c_1\varphi_1(x_k) + ... + c_n\varphi_n(x_k), \quad k = 0,1,...,m. \qquad (112)$$

Функции φ_k называются базисными и должны быть ортогональны друг другу. В качестве базисных функций можно выбрать канонический многочлен (62), полиномы Чебышева (86) и др.

Образуем разности

$$r_k = \varphi(x_k) - f(x_k), \quad k = 0,1,...,m, \qquad (113)$$

характеризующие отклонение в узлах x_k точного значения функции $f(x)$ от ее приближенного значения, полученного с помощью обобщенного многочлена (112). Для вектора погрешностей

$$\mathbf{r} = (r_0, r_1, ..., r_m)^T \qquad (114)$$

можно ввести ту или иную норму, например,

$$\|\mathbf{r}\| = (\sum_{k=0}^{m} r^2{}_k)^{1/2} = (\sum_{k=0}^{m} (\varphi(x_k) - f(x_k))^2)^{1/2} \qquad (115)$$

или

$$\|\mathbf{r}\| = \max_{0\le k\le m} |r_k| = \max_{0\le k\le m} |\varphi_k - f(x_k)|. \qquad (116)$$

Задача о наилучшем приближении функции $f(x)$, заданной таблично, состоит в нахождении коэффициентов c_0, c_1,..., c_n, минимизирующих норму

вектора r. В зависимости от выбора нормы получим различные задачи. Так, норме (115) соответствует задача о наилучшем среднеквадратичном приближении, а норме (116) – задача о наилучшем равномерном приближении функции, заданной таблично.

Если $m=n$, то независимо от выбора нормы решение $\mathbf{c}=(c_0, c_1,..., c_n)^m$ задачи о наилучшем приближении совпадает с решением задачи интерполяции. Действительно, в этом случае требование $\|\mathbf{r}\|=0$ приводит к выполнению основного условия интерполяции

$$\varphi(x_k) = f(x_k),\ k = 0,1,...,n.$$

12 Сглаживание сеточных функций

Пусть имеется таблица значений $\{fi\}^N_{i=0}$ функции $f(x)$, полученная, например, путем измерения некоторой физической величины или с помощью численных расчетов. Может оказаться, что $f(x)$ сильно меняется на отдельных участках. В этом случае иногда целесообразно применить *процедуру сглаживания,* т.е. приближенно заменить $f(x)$ другой, более гладкой функцией $\varphi(x)$.

Для построения сглаженных функций можно воспользоваться среднеквадратичными приближениями. Многочлен $\varphi^{(i)}(x)$ наилучшего среднеквадратичного приближения, построенный по значениям f_{i-1}, f_i, f_{i+1}, имеет вид

$$\varphi^{(i)}(x) = \frac{f_{i-1} + f_i + f_{i+1}}{3} + \frac{f_{i+1} - f_{i-1}}{2h}(x - x_i), \qquad (117)$$

причем

$$\varphi^{(i)}(x_i) = \frac{f_{i-1} + f_i + f_{i+1}}{3},\ i = 1,2,...,N-1. \qquad (118)$$

Доопределим $\varphi^{(0)}(x_0)=f_0$, $\varphi^{(N)}(x_N)=f_N$ и обозначим $\varphi_i=\varphi^{(i)}(x_i)$ $i=0,1,\ldots,N$.

Процедура сглаживания по формулам (118) состоит в замене сеточной функции $\{f_i\}^N_i=0$ сеточной функцией $\{\varphi_i\}^N_i=0$. То, что такая замена действительно осуществляет сглаживание, можно иллюстрировать примером, приведенным в таблице 2.

Таблица 2

i	0	1	2	3	4	5	6	7	8	9	10	11	12
f_i	1	1	1	1	0	0	0	0	10	0	0	0	0
φ_i	1	1	1	2/3	1/3	0	0	10/3	10/3	10/3	0	0	0

Здесь функция f_i имеет две особенности: разрыв при i=3 и выброс при i=8. Сглаживание приводит к размазыванию разрыва, а также к размазыванию выброса и уменьшению его амплитуды. На участках гладкости $f(x)$ функция $\varphi(x)$ также остается гладкой. Для наглядности читателю предлагается построить графики функций $f(x)$ и $\varphi(x)$.

В рассмотренном случае сглаживание свелось к *осреднению* функции $f(x)$ по трем соседним точкам. Можно проводить осреднение и по большему числу точек, например по пяти точкам, когда

$$\varphi_i = \sum_{j=-2}^{2} \alpha_j f_{i+j}, \quad \sum_{j=-2}^{2} \alpha_j = 1.$$

Поясним, почему осреднение приводит к сглаживанию. Будем считать, что $f(x)$ задана на равномерной сетке

$$\omega_h = \{x_i = ih, i = 0,1,...,N, hN = l\},$$

причем f_0=f_N=0. Осреднение $f(x)$ по формулам (118) приводит к функции

$$\varphi_i = \frac{f_{i-1} + f_i + f_{i+1}}{3} = f_i + \frac{h^2}{3} f_{\overline{xx},i}, \qquad (119)$$

$i = 1,2,...,N-1, \varphi_0 = \varphi_N = 0$.

Таким образом, можно считать, что процедура осреднения представляет собой замену сеточной функции f сеточной функцией Tf, где $T=E+h^2\Lambda/3$, E – единичный оператор, Λ – оператор второй разностной производной. Будем называть *T оператором осреднения.*

Можно показать, что оператор T не подавляет низкочастотные гармоники и уменьшает амплитуду высокочастотных гармоник примерно в три раза, что и объясняет эффект сглаживания.

13. Разностная аппроксимация производных

Рассмотрим задачу о приближенном вычислении производных функции $u(x)$, определенной и непрерывной на отрезке $[a,b]$. Будем считать, что $u(x)$ обладает необходимой по ходу изложения гладкостью. Введем согласно (101) сетку ω_h и заменим предел приращения функции на конечную разность, используя значения сеточной функции слева и справа от рассматриваемой точки $x = x_i$:

$$u_i = u(x_i), \quad u_{\overline{x},i} = (u_i - u_{i-1})/h,$$
$$u_{x,i} = (u_{i+1} - u_i)/h, \quad u_{\overset{o}{x},i} = (u_{i+1} - u_{i-1})/2h. \tag{120}$$

Приведенные здесь разностные отношения называются соответственно *левой, правой и центральной разностными производными функции* $u(x)$ *в точке* $x = x_i$. Если точка x_i фиксирована, а шаг h стремится к нулю (при этом количество разбиений $N \to \infty$), то каждое из этих разностных соотношений стремится к значению производной функции $u(x)$ в точке x_i, поэтому в качестве приближенного значения $u'(x)$ можно взять любое из них.

Нетрудно получить выражение для погрешности, возникающей при замене дифференциального выражения разностным. Рассмотрим, например, левую разностную производную в точке $x = x_i$ и перепишем ее в виде

$$u_{\overline{x},i} = \frac{u(x) - u(x-h)}{h}.$$

Используя разложение функции $u(x)$ в ряд Тейлора

$$u(x-h) = u(x) - hu'(x) + \frac{h^2}{2}u''(\xi), \xi \in (x-h, x), \tag{121}$$

получим

$$u_{\overline{x},i} = u'(x_i) - \frac{h}{2}u''(\xi_i). \tag{122}$$

Погрешность $u_{\overline{x},i} - u'(x_i)$, возникающая при замене дифференциального выражения $u'(x)$ разностным выражением $u_{\overline{x},i}$, называется *погрешностью аппроксимации.* Из разложения (122) видно, что погрешность аппроксимации является величиной O(h) при $h \to 0$. В этом случае говорят, что имеет место *аппроксимация первого порядка.* Обратим внимание, что в этом разделе речь идет о разностной аппроксимации в отличие от аппроксимации функций, рассмотренной в разделе 11. Объединяет эти два раздела лишь английский термин (в обоих случаях речь идет о приближенной замене).

Приведем разложения, аналогичные (122), для других разностных отношений:

$$u_{x,i} = u'(x_i) + \frac{h}{2}u''(\xi_i^{(1)}), \quad \xi_i^{(1)} \in (x_i, x_{i+1}), \tag{123}$$

$$u_{\overset{o}{x},i} = u'(x_i) + \frac{h^2}{6}u'''(\xi_i^{(2)}), \quad \xi_i^{(2)} \in (x_{i-1}, x_{i+1}), \tag{124}$$

Из разложения (124) видно, что центральная разностная производная аппроксимирует $u'(x)$ со вторым порядком и, следовательно, является более точным приближением к $u'(x)$, чем левая или правая разностные производные. Наряду со (122) – (124) можно использовать менее детальную запись тех же разложений, а именно

$$u_{\overset{-}{x},i} = u'_i + o(h), u_{x,i} = u'_i + o(h), u_{\overset{o}{x},i} = u'_i + o(h^2).$$

Вторую производную $u''(x)$ можно приближенно заменить в точке $x_i \in \omega_h$ *второй разностной производной*

$$u_{\overset{-}{xx},i} = \frac{1}{h}(u_{x,i} - u_{\overset{-}{x},i}) = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}. \tag{125}$$

Разложение по формуле Тейлора приводит к следующему выражению для погрешности:

$$u_{\overset{-}{xx},i} - u''(x_i) = \frac{h^2}{12}u^{(4)}(\xi_i), \tag{126}$$

т.е. имеет место аппроксимация второго порядка.

Мы привели простейшие интуитивно понятные примеры аппроксимации дифференциальных выражений разностными, порядок аппроксимации которых относительно невысок.

Записывая интерполяционный многочлен Лагранжа $L(x)$ и его остаточный член $R_L(x)$ (см. (67) и (84)) для случая пяти узлов интерполяции (n=4) и дифференцируя их соответствующее количество раз, получим следующие выражения для производных функции в центральном узле $x = x_i$:

$$u'(x_i) = \frac{1}{12h}(u_{i-2} - 8u_{i-1} + 8u_{i+1} - u_{i+2}) + \frac{h^4}{30}u_*^V, \tag{127}$$

$$u''(x_i) = \frac{1}{12h^2}(-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}) + O(h^4). \tag{128}$$

Таким образом, получены формулы повышенного порядка аппроксимации. Отметим, что в случае трех узлов интерполяции (n=2)

получаются выражения, совпадающие с записанными ранее центральными разностными производными.

Для численного вычисления производной функции на концах отрезка [a,b] (где не определены, например, u_{i-2} или u_{i+1}) можно воспользоваться левой и правой разностными производными соответственно, а также формулами повышенного порядка аппроксимации, которые можно получить, записав продифференцированный многочлен Лагранжа не в центральном, а в левом или правом узле интерполяции.

В общем случае погрешность, возникающая в результате замены дифференциального выражения разностным, зависит как от распределения узлов сетки, так и от гладкости функции.

Для случая произвольного расположения узлов использование многочленов Лагранжа приводит к вычислению громоздких выражений, поэтому удобнее применять метод неопределенных коэффициентов.

14 Численное интегрирование

14.1 Квадратурная формула. Частичные отрезки

Численное вычисление определенных интегралов

$$I = \int_a^b f(x)dx \tag{129}$$

основано на замене интеграла конечной суммой

$$I_n = \sum_{k=0}^{n} c_k f(x_k), \tag{130}$$

где c_k – числовые коэффициенты и x_k – точки отрезка [a, b], k=0,1,…,n. Приближенное равенство

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

называется квадратурной формулой, а сумма вида (130) – квадратурной суммой. Точки x_k называются узлами квадратурной формулы, а числа c_k – коэффициентами квадратурной формулы. Разность

$$\Psi = \int_a^b f(x)dx - \sum_{k=0}^{n} c_k f(x_k) \qquad (131)$$

называется *погрешностью квадратурной формулы*. Погрешность зависит как от расположения узлов, так и от выбора коэффициентов. При оценке погрешности в приводимых ниже примерах функция $f(x)$ предполагается достаточно гладкой.

Введем на $[a, b]$ равномерную сетку ω_h с шагом h (101) и представим интеграл (129) в виде суммы интегралов по частичным отрезкам:

$$\int_a^b f(x)dx = \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} f(x)dx. \qquad (132)$$

Для построения формулы численного интегрирования на всем отрезке $[a, b]$ достаточно построить квадратурную формулу для интеграла

$$\int_{x_{i-1}}^{x_i} f(x)dx \qquad (133)$$

на частичном отрезке $[x_{i-1}, x_i]$ и воспользоваться свойством (132).

14.2 Формула прямоугольников

Заменим интеграл (133) выражением $f(x_{i-1/2})h$, где $x_{i-1/2}=x_i-0{,}5h$. Геометрически такая замена означает, что площадь криволинейной трапеции *ABCD* заменяется площадью прямоугольника ABC'D' (см. рис. 4). Тогда получим формулу

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx f(x_{i-1/2})h, \qquad (134)$$

которая называется формулой прямоугольников на частичном отрезке$[x_{i-1}, x_i]$.

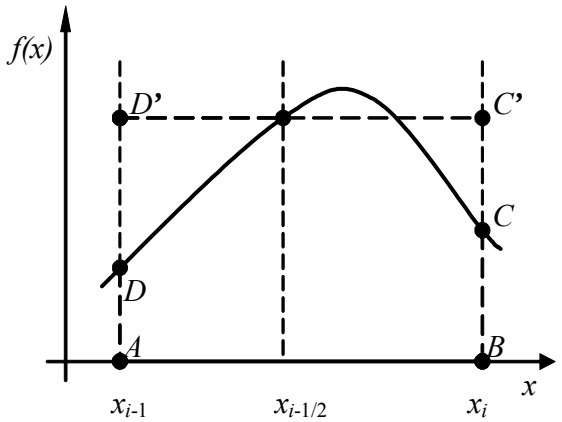

Рис. 4 – Геометрический смысл
формулы прямоугольников

Погрешность метода (134) определяется величиной

$$\Psi_i = \int_{x_{i-1}}^{x_i} f(x)dx - f(x_{i-1/2})h,$$

которую легко оценить с помощью формулы Тейлора. Действительно, запишем Ψ_i в виде

$$\Psi_i = \int_{x_{i-1}}^{x_i} (f(x)dx - f(x_{i-1/2}))h \qquad (135)$$

и воспользуемся разложением

$$f(x) = f(x_{i-1/2}) + (x - x_{i-1/2})f'(x_{i-1/2}) + \frac{(x - x_{i-1/2})^2}{2} f''(\zeta_i),$$

где $\zeta_i = \zeta_i(x) \in [x_{i-1/2}, x_i]$. Тогда из (135) получим

$$\Psi_i = \int_{x_{i-1}}^{x_i} \frac{(x_{i-1/2})^2}{2} f''(\zeta_i)dx.$$

Обозначая $M_{2,i} = \max\limits_{x\in[x_{i-1},x_i]} |f''(x)|$, оценим Ψ_i следующим образом:

$$|\Psi_i| \le M_{2,i} \int_{x_{i-1}}^{x_i} \frac{(x_{i-1/2})^2}{2} dx = \frac{h^3}{24} M_{2,i}.$$

Таким образом, для погрешности формулы прямоугольников на частичном отрезке справедлива оценка

$$|\Psi_i| \le \frac{h^3}{24} M_{2,i}, \qquad (136)$$

т.е. формула имеет погрешность $O(h^3)$ при $h \to 0$.

Заметим, что оценка (136) является неулучшаемой, т.е. существует функция $f(x)$, для которой (136) выполняется со знаком равенства. Действительно, для $f(x) = (x - x_{i-1/2})^2$ имеем $M_{2.i} = 2$, $f(x_{i-1/2}) = 0$ и

$$\int_{x_{i-1}}^{x_i} f(x)dx - f(x_{i-1/2})h = \frac{h^3}{12} = \frac{h^3}{24} M_{2,i}.$$

Суммируя равенства (134) по i от 1 до N, получим *составную формулу прямоугольников*

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} f(x_{i-1/2})h. \qquad (137)$$

Погрешность этой формулы

$$\Psi = \int_a^b f(x)dx - \sum_{i=1}^{N} f(x_{i-1/2})h$$

равна сумме погрешностей по всем частичным отрезкам,

$$\Psi = \sum_{i=1}^{N} \Psi_i = \sum_{i=1}^{N} \int_{x_{i-1}}^{x_i} \frac{(x - x_{i-1/2})^2}{2} f''(\zeta_i)dx.$$

Отсюда, обозначая $M_2 = \max\limits_{x \in [a,b]} |f''(x)|$, получим

$$|\Psi| \le \frac{M_2 N h^3}{24} = \frac{h^2 (b-a) M_2}{24}, \qquad (138)$$

т.е. погрешность формулы прямоугольников на всем отрезке есть величина $O(h^2)$.

В этом случае говорят, что квадратурная формула имеет *второй порядок точности.*

З а м е ч а н и е. Возможны формулы прямоугольников и при ином расположении узлов, например, такие

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} hf(x_{i-1}), \quad \int_a^b f(x)dx \approx \sum_{i=1}^{N} hf(x_i),$$

называемые формулами левых и правых прямоугольников.

Однако из-за нарушения симметрии погрешность таких формул является величиной O(h).

14.3 Формула трапеций

На частичном отрезке эта формула имеет вид

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{f(x_{i-1})+f(x_i)}{2}h \qquad (139)$$

и получается путем замены подинтегральной функции *f(x)* интерполяционным многочленом первой степени, построенным по узлам x_{i-1}, x_i, т.е. функцией

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

Для оценки погрешности воспользуемся известным свойством

$$f(x) - L_{1,i}(x) = \frac{(x-x_{i-1})(x-x_i)}{2} f''(\zeta_i(x)).$$

Отсюда получим

$$\Psi_i = \int_{x_{i-1}}^{x_i} f(x)dx - \frac{f(x_{i-1})+f(x_i)}{2}h = \int_{x_{i-1}}^{x_i} (f(x) - L_{1,i}(x))dx =$$

$$= \int_{x_{i-1}}^{x_i} \frac{(x-x_{i-1})(x-x_i)}{2} f''(\zeta_i(x))dx$$

и, следовательно,

$$|\Psi_i| \le \frac{M_{2,i}h^3}{12}. \qquad (140)$$

Оценка (140) неулучшаемая, так как в ней достигается равенство, например, для $f(x) = (x-x_i)^2$.

Составная формула трапеций имеет вид

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} \frac{f(x_i)+f(x_{i-1})}{2}h = h(0.5f_0 + f_1 + ... + f_{n-1} + 0.5f_N), \qquad (141)$$

где $f_i = f(x_i),\ i = 0,1,...,N,\ hN = b-a.$

Погрешность этой формулы оценивается следующим образом:

$$|\Psi| \le \frac{h^2(b-a)}{12} M_2, M_2 = \max_{x\in[a,b]} |f''(x)|. \qquad (142)$$

Таким образом, формула трапеций имеет, так же как и формула прямоугольников, второй порядок точности $\Psi = O(h^2)$, но ее погрешность оценивается величиной в два раза большей (см. (138)).

14.4 Формула Симпсона

При апроксимации интеграла (133) заменим функцию *f*(*x*) параболой, проходящей через точки $(x_j, f(x_j))$, *j=i–1, i–0.5,i,* т.е. представим приближенно *f*(*x*) в виде

$$f(x) \approx L_{2,i}(x), \quad x \in [x_{i-1}, x_i],$$

где $L_{2,i}(x)$ – интерполяционный многочлен Лагранжа второй степени,

$$L_{2,i}(x) = \frac{2}{h^2}\{(x - x_{i-1/2})(x - x_i) f_{i-1} -$$
$$-2(x - x_{i-1})(x - x_i) f_{i-1/2} + (x - x_i)(x - x_{i-1/2}) f_i\}. \qquad (143)$$

Проводя интегрирование, получим

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

Таким образом, приходим к приближенному равенству

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{h}{6}(f_{i-1} + 4f_{i-1/2} + f_i), \qquad (144)$$

которое называется формулой Симпсона или формулой парабол.

На всем отрезке [*a*, *b*] формула Симпсона имеет вид

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} \frac{h}{6}(f_{i-1} + 4f_{i-1/2} + f_i) =$$
$$= \frac{h}{6}[f_0 + f_N + 2(f_1 + f_2 + \ldots + f_{n-1}) + 4(f_{1/2} + f_{3/2} + \ldots + f_{N-1/2}].$$

Чтобы не использовать дробных индексов, можно обозначить

$$x_i = a + 0.5hi, \quad f_i = f(x_i), \quad i = 0,1,...,2N, \quad hN = b - a$$

и записать формулу Симпсона в виде

$$\int_a^b f(x)dx \approx b - a[f_0 + f_{2N} + 2(f_2 + f_4 + ... + f_{2N-2}) + 4(f_1 + f_3 + ... + f_{2N-1})]. \qquad (145)$$

Прежде чем переходить к оценке погрешности формулы (144), заметим, что она является точной для любого многочлена третьей степени, т.е. имеет место точное равенство

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

если $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$. Это утверждение нетрудно проверить непосредственно, что и предоставляется сделать читателю.

Для погрешности формулы Симпсона справедлива оценка:

$$|\Psi_i| \le \frac{h^5}{2880} M_{4,i}. \qquad (146)$$

Погрешность составной формулы Симпсона (145) вычисляется так:

$$|\Psi| \le \frac{h^4(b-a)}{2880} M_4, \quad hN = b - a, \; M_4 = \sup_{x \in [a,b]} |f^{IV}(x)|. \qquad (147)$$

Отсюда видно, что формула Симпсона существенно точнее, чем формулы прямоугольников и трапеций. На частичном отрезке она имеет точность $O(h^5)$, а на всем отрезке – $O(h^4)$.

15 Численные методы решения задачи Коши для обыкновенных дифференциальных уравнений

15.1 Постановка задачи Коши

Будем рассматривать задачу Коши для систем обыкновенных дифференциальных уравнений

$$\frac{d\mathbf{u}(t)}{dt} = f(t, \mathbf{u}), \quad t > 0, \quad \mathbf{u}(0) = \mathbf{u}^{(0)} \qquad (148)$$

или, подробнее,

$$\frac{du_i(t)}{dt} = f_i(t,u_1,u_2,...,u_m), \quad t>0, \quad i=1,2,...,m. \tag{149}$$

$$u_i(0) = u_i^{(0)}, \quad i=1,2,...,m. \tag{150}$$

Хорошо известны условия, гарантирующие существование и единственность решения задачи Коши. Предположим, что функции f_i, $i=1,2,...,m$ непрерывны по всем аргументам в замкнутой области

$$D = \left\{ \ |t| \le a, \left|u_i - u_i^{(0)}\right| \le b, i=1,2,...,m \ \right\}.$$

Из непрерывности функций f_i следует их ограниченность, т.е. существование константы $M>0$, такой, что всюду в D выполняются неравенства $|f_i| \le M, i=1,2,...,m.$

Предположим, кроме того, что в D функции f_i удовлетворяют условию Липшица по аргументам $u_1,u_2,...,u_m$, т.е.

$$f_i(t,u_1',u_2',...,\ u_m') - f_i(t,u_1'',u_2'',...,\ u_m'') \le L\left\{|u_1'-u_2''| + |u_2'-u_2''| + ... + |u_m'-u_m''|\right\}$$

для любых точек $(t,u_1',...,u_m')$ и $(t,u_1'',u_2'',...,u_m'')$ области D.

Если выполнены сформулированные выше предположения, то существует единственное решение

$$u_1 = u_1(t), u_2(t), ..., u_m = u_m(t)$$

системы (149), определенное при $\left(t\right(\le t_0 = \min(a, b/M)$ и принимающее при $t=0$ заданные начальные значения (150).

При исследовании численных методов для задачи Коши будем заранее предполагать, что ее решение существует, единственно и обладает необходимыми свойствами гладкости.

Для простоты изложения будем рассматривать далее одно уравнение

$$\frac{du}{dt} = f(t,u), \qquad t>0, \qquad u(0)=u_0. \tag{151}$$

Введем по переменному t равномерную сетку с шагом $\tau>0$, т.е. рассмотрим множество точек

$$\omega_\tau = \{t_n = n\tau, n = 0,\ 1,\ 2,...\}.$$

Будем обозначать через $u(t)$ точное решение задачи (151), а через $y_n = y(t_n) -$ приближенное решение. Заметим, что приближенное решение является сеточной функцией, т.е. определено только в точках сетки ω_τ.

15.2 Метод Эйлера

Уравнение (151) заменяется разностным уравнением

$$\frac{y_{n+1} - y_n}{\tau} - f(t_n, y_n) = 0, \quad n = 0,\ 1,\ 2,..., \quad y_0 = u_0. \qquad (152)$$

Решение этого уравнения находится явным образом по рекуррентной формуле

$$y_{n+1} = y_n + \tau f(t_n, y_n), \quad n = 0,\ 1,\ 2,..., y_0 = u_0. \qquad (153)$$

При использовании приближенных методов основным является вопрос о сходимости. Понятие о сходимости приближенного метода можно сформулировать по-разному. Применительно к разностным методам, к которым относится метод Эйлера (153), наибольшее распространение получило понятие сходимости при $\tau \to 0$. Оно означает следующее. Фиксируем точку t и построим последовательность сеток ω_τ таких, что $\tau \to 0$ и $t_n = n\tau = t$ (тогда необходимо $n \to \infty$). Говорят, что метод (153) сходится в точке t, если $|y_n - u(t_n)| \to 0$ при $\tau \to 0, t_n = t$. Метод *сходится* на отрезке $(0,T]$, если он сходится в каждой точке $t \in (0,T]$. Говорят, что метод имеет p-й порядок точности, если существует число $p>0$ такое, что $|y_n - u(t_n)| = O(\tau^P)$ при $\tau \to 0$.

Получим уравнение, которому удовлетворяет *погрешность метода* $z_n = y_n - u(t_n)$. Подставляя $y_n = z_n + u_n$ в (152), получим

$$\frac{z_{n+1} - z_n}{\tau} = f(t_n, u_n + z_n) - \frac{u_{n+1} - u_n}{\tau}. \qquad (154)$$

Правую часть уравнения (154) можно представить в виде суммы $\psi_n^{(1)} + \psi_n^{(2)}$,

где

$$\psi_n^{(1)} = -\frac{u_{n+1} - u_n}{\tau} + f(t_n, u_n),$$

$$\psi_n^{(2)} = f(t_n, u_n + z_n) - f(t_n, u_n).$$

Функция $\psi_n^{(1)}$ называется *невязкой* или *погрешностью аппроксимации разностного уравнения (152) на решении исходного уравнения (151)*. Видно, что невязка представляет собой результат подстановки точного решения $u = u(t)$ в левую часть разностного уравнения (152). Если бы приближенное решение y_n совпадало с точным $u(t_n)$, то невязка равнялась бы нулю. Говорят, что разностный метод *аппроксимирует исходное дифференциальное уравнение*, если $\psi_n^{(1)} \to 0$ при $\tau \to 0$. Разностный метод имеет *p*-й порядок аппроксимации, если $\psi_n^{(1)} = O(\tau^p)$. Можно показать, что при очень общих предположениях порядок точности разностного метода совпадает с порядком аппроксимации.

Функция

$$\psi_n^{(2)} = f(t_n, u_n + z_n) - f(t_n, u_n)$$

обращается в нуль, если правая часть f не зависит от решения $u(t)$. В общем случае $\psi_n^{(2)}$ пропорциональна погрешности z_n, так как по формуле конечных приращений имеем

$$\psi_n^{(2)} = \frac{\partial f}{\partial u}(t_n, u_n + \theta z_n) z_n, \qquad |\theta| \le 1.$$

Порядок аппроксимации метода Эйлера (153) нетрудно найти, используя разложение по формуле Тейлора. Поскольку

$$\frac{u_{n+1} - u_n}{\tau} = u'(t_n) + O(\tau),$$

то в силу уравнения (151)

$$\psi_n^{(1)} = -u'(t_n) + f(t_n, u_n) + O(\tau) = O(\tau),$$

т.е. метод Эйлера имеет первый порядок аппроксимации. При выводе предполагалась ограниченность $u''(t)$.

15.3 Симметричная схема

Уравнение (151) заменяется разностным уравнением

$$\frac{y_{n+1}-y_n}{\tau}-\frac{1}{2}(f(t_n,y_n)+f(t_{n+1},y_{n+1}))=0,\ n=0,1,...,\quad y_0=u_0. \quad (155)$$

Данный метод более сложен в реализации, чем метод Эйлера (153), так как новое значение y_{n+1} определяется по найденному раннее y_n путем решения уравнения

$$y_{n+1}-0.5\tau f(t_{n+1},y_{n+1})=F_n,$$

где

$$F_n=y_n+0.5\tau f(t_n,y_n).$$

По этой причине метод называется *неявным*. Преимуществом метода (155) по сравнению с (153) является более высокий порядок точности.

Для невязки

$$\psi_n^{(1)}=-\frac{u_{n+1}-u_n}{\tau}+\frac{1}{2}(f(t_n,y_n)+f(t_{n+1},y_{n+1}))$$

справедливо разложение

$$\psi_n^{(1)}=-u_n'-\frac{\tau}{2}u_n''+O(\tau^2)+\frac{1}{2}(u_n'+u_{n+1}')=-u_n'-\frac{\tau}{2}u_n''+$$

$$+\frac{1}{2}(u_n'+u_n'+\tau u_n''+O(\tau^2)).$$

т.е. $\psi_n^{(1)}=O(\tau^2)$. Таким образом, метод (155) имеет второй порядок аппроксимации и, как можно показать, второй порядок точности.

Приведенные примеры представляют собой простейшие случаи *разностных методов*, или, как их еще называют, *разностных схем*. Методы Рунге–Кутта отличаются от разностных методов тем, что в них допускается вычисление правых частей $f(t,u)$ не только в точках сетки, но и в промежуточных точках.

15.4 Методы Рунге–Кутта

Явный m-этапный метод Рунге–Кутта состоит в следующем. Пусть решение $y_n=y(t_n)$ уже известно. Задаются числовые коэффициенты

$$a_i,b_{ij},i=2,\ 3,...,m,\ j=1,\ 2,...,m-1,\sigma_i,i=1,\ 2,...,m, \quad (156)$$

и последовательно вычисляются функции

$$k_1 = f(t_n, y_n), k_2 = f(t_n + a_2\tau, y_n + b_{21}\tau k_1),$$
$$k_3 = f(t_n + a_3\tau, y_n + b_{31}\tau k_1 + b_{32}\tau k_2), \ldots, \quad (157)$$
$$k_m = f(t_n + a_m\tau, y_n + b_{m1}\tau k_1 + b_{m2}\tau k_2 + \ldots + b_{m,m-1}\tau k_{m-1}).$$

Затем из формулы

$$\frac{y_{n+1} - y_n}{\tau} = \sum_{i=1} \sigma_i k_i \quad (158)$$

находится новое значение $y_{n+1} = y(t_{n+1})$.

Коэффициенты a_i, b_{ij}, σ_i выбираются из соображений точности. Например, для того чтобы уравнение (158) аппроксимировало исходное уравнение (151), необходимо потребовать $\sum_{i=1}^{m} \sigma_i = 1$. Отметим, что методы Рунге–Кутта при *m>5* не используются.

Остановимся более подробно на отдельных методах. При $m = 1$ получаем рассмотренный ранее метод Эйлера. При $m = 2$ получаем семейство методов

$$k_1 = f(t_n, y_n), k_2 = f(t_n + a_2\tau, y_n + b_{21}\tau k_1),$$
$$y_{n+1} = y_n + \tau(\sigma_1 k_1 + \sigma_2 k_2). \quad (159)$$

Исследуем погрешность аппроксимации методов (159) в зависимости от выбора параметров. Исключая из последнего уравнения функции k_1 и k_2, получаем

$$\frac{y_{n+1} - y_n}{\tau} = \sigma_1 f(t_n, y_n) + \sigma_2 f(t_n + a_2\tau, y_n + b_{21}\tau f(t_n, y_n)). \quad (160)$$

По определению погрешностью аппроксимации или невязкой метода (159) является выражение

$$\psi_n^{(1)} = -\frac{u_{n+1} - u_n}{\tau} + \sigma_1 f(t_n, u_n) + \sigma_2 f(t_n + a_2\tau, u_n + b_{21}\tau f(t_n, u_n)), \quad (161)$$

полученное заменой в (160) приближенного решения y_n точным решением $u_n = u(t_n)$.

Найдем порядок погрешности аппроксимации в предположении достаточной гладкости решения $u(t)$ и функции $f(t,u)$. Для этого

разложим все величины, входящие в выражение (161), по формуле Тейлора в точке t_n. Имеем

$$\frac{u_{n+1}-u_n}{\tau}=u'(t_n)+\frac{\tau}{2}u''(t_n)+O(\tau^2),$$

$$f(t_n+a_2\tau,u_n+b_{21}\tau f_n)=f_n+a_2\tau\frac{\partial f_n}{\partial t}+b_{21}\tau f_n\ \frac{\partial f_n}{\partial u}+O(\tau^2),$$

где $f_n=f(t_n,u_n),\frac{\partial f_n}{\partial u}=\frac{\partial f}{\partial u}(t_n,u_n).$ Используя согласно уравнению (151) тождество

$$u''=\frac{\partial f}{\partial t}+\frac{\partial f}{\partial u}u'=\frac{\partial f}{\partial t}+\frac{\partial f}{\partial u}f.$$

получим

$$\psi_n^{(1)}=-u_n'+(\sigma_1+\sigma_2)f_n+\tau\left[(\sigma_2b_{21}-0.5)f_n\frac{\partial f_n}{\partial u}+(\sigma_2a_2-0.5)\frac{\partial f_n}{\partial t}\right]+O(\tau^2). \qquad (162)$$

Отсюда видно, что методы (159) имеют первый порядок аппроксимации, если $\sigma_1+\sigma_2=1.$

Если же дополнительно потребовать $\sigma_2a_2=\sigma_2b_{21}=0.5,$ то получим методы второго порядка аппроксимации. Таким образом, имеется однопараметрическое семейство двухэтапных методов Рунге–Кутта второго порядка аппроксимации. Это семейство методов можно записать в виде

$$\frac{y_{n+1}-y_n}{\tau}=(1-\sigma)f(t_n,y_n)+\sigma f(t_n a\tau,y_n+a\tau f(t_n,y_n)), \qquad (163)$$

где $\sigma a=0.5.$

В частности, при $\sigma=0.5,\ a=1$ получаем метод Рунге– Кутта второго порядка со средней производной:

$$\begin{aligned}k_1&=f(t_n,y_n),k_2=f(t_n+\tau,y_n+\tau k_1),\\ y_{n+1}&=y_n+0.5\tau(k_1+k_2).\end{aligned} \qquad (164)$$

При $\sigma=1,\ a=0.5$ получаем метод Рунге–Кутта второго порядка с производной в средней точке.

$$k_1=f(t_n,y_n),\qquad k_2=f(t_n+0.5\tau,y_n+0.5\tau k_1),$$
$$y_{n+1}=y_n+\tau k_2. \qquad (165)$$

Поскольку требуется вычислить две промежуточные функции k_1 и k_2, данные методы относятся к *двухэтапным методам*.

Для исследования невязки перепишем (165) в одно уравнение:

$$\frac{y_{n+1}-y_n}{\tau}-f(t_n+0.5\tau,y_n+0.5\tau f_n)=0. \qquad (166)$$

Его невязка равна

$$\psi_n^{(1)}=-\frac{u_{n+1}-u_n}{\tau}+f(t_n+0.5\tau,u_n+0.5\tau f(t_n,u_n)). \qquad (167)$$

Имеем

$$\frac{u_{n+1}-u_n}{\tau}=u_n'+0.5\tau u_n''+O(\tau^2),$$

$$f(t_n+0.5\tau,u_n+0.5\tau f(t_n,u_n))=f(t_n,u_n)+$$

$$0.5\tau\frac{\partial f(t_n,u_n)}{\partial t}+0.5\tau f(t_n,u_n)\frac{\partial f(t_n,u_n)}{\partial u}=f(t_n,u_n)+0.5\tau u_n'',$$

Таким образом, метод (165) имеет второй порядок погрешности аппроксимации $\psi_n^{(1)}=O(\tau^2)$ и в отличие от симметричной схемы (155) является явным.

Реализация двухэтапных методов Рунге–Кутта (164) и (165) соответствует методу *предиктор – корректор* (предсказывающе – исправляющему), поскольку на первом этапе приближенное значение предсказывается с невысокой точностью $O(\tau)$, а на втором этапе это предсказанное значение исправляется, так что результирующая погрешность имеет второй порядок по τ.

Двухэтапных методов третьего порядка аппроксимации не существует. Чтобы убедиться в этом, достаточно рассмотреть уравнение $u'=u.$ Для него двухэтапный метод Рунге–Кутта (163) принимает вид

$$\frac{y_{n+1}-y_n}{\tau}=(1+\tau\sigma a)y_n$$

и погрешность аппроксимации равна

$$\psi_n^{(1)}=-\frac{u_{n+1}-u_n}{\tau}+(1+\tau\sigma a)u_n.$$

Разлагая $\psi_1^{(n)}$ по формуле Тейлора и учитывая, что $u'''=u''=u'=u,$ получим

$$\psi_n^{(1)}=\tau(\sigma a-0.5)u-\frac{\tau^2}{6}u(t_n+\theta\tau),\ 0\le\theta\le 1.$$

Отсюда видно, что наивысший достижимый порядок аппроксимации равен двум.

Приведем частные случаи методов Рунге–Кутта более высокого порядка аппроксимации.

Метод третьего порядка:

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau}{2}k_1),$$

$$k_3 = f(t_n + \tau, y_n - \tau k_1 + 2\tau k_2), \quad \frac{y_{n+1} - y_n}{\tau} = \frac{1}{6}(k_1 + 4k_2 + k_3).$$

Метод третьего порядка:

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + \frac{\tau}{3}, y_n + \frac{\tau k_1}{3}),$$

$$k_3 = f(t_n + \frac{2\tau}{3}, y_n + \frac{2\tau k_2}{3}), \quad \frac{y_{n+1} - y_n}{\tau} = \frac{1}{4}(k_1 + 3k_3).$$

Метод четвертого порядка:

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_1}{2}),$$

$$k_3 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_2}{2}), \quad k_4 = f(t_n + \tau, y_n + \tau k_3),$$

$$\frac{y_{n+1} - y_n}{\tau} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Метод четвертого порядка:

$$k_1 = f(t_n, y_n), \quad k_2 = f(t_n + \frac{\tau}{4}, y_n + \frac{\tau k_1}{4}),$$

$$k_3 = f(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_2}{2}),$$

$$k_4 = f(t_n + \tau, y_n + \tau k_1 - 2\tau k_2 + 2\tau k_3),$$

$$\frac{y_{n+1} - y_n}{\tau} = \frac{1}{6}(k_1 + 4k_3 + k_4).$$

16 Численные методы решения граничных задач для обыкновенных дифференциальных уравнений

16.1. Постановка граничной задачи

Граничная задача для систем ОДУ (149)

$$\frac{du_i(t)}{dt} = f_i(t,u_1,u_2,...,u_m), \quad t>0, \quad i=1,2,...,m$$

отличается от рассмотренной ранее задачи Коши тем, что ни в одной из точек t не известны условия на все сразу функции u_i, $(i=1,2,...,m)$, и состоит в отыскании функций $u_i(t)$, непрерывно дифференцируемых на интервале (a,b), непрерывных на отрезке $[a,b]$, удовлетворяющих уравнениям (149) при $t \in (a,b)$ дополненных граничными условиями

$$u_i(t_i) = u_i^{(0)}, \quad t_i \in [a,b], \; i=1,2,...,m. \tag{168}$$

Очевидно, что граничная задача определена минимум для двух уравнений. Как правило, n условий известно в начальной точке a интервала и m-n условий – в конечной точке b интервала, т.е. на его границах. Ни один из методов решения задачи Коши не удается применить к решению граничной задачи в силу недостатка условий на m-n функций в начальной точке.

16.2 Метод стрельбы

Метод стрельбы основывается на рассмотренных ранее методах решения задачи Коши и состоит в построении предположения о недостающих значениях неизвестных функций в начальной точке интервала и их пошаговом уточнении в дальнейшем.

Рассмотрим для простоты систему двух ОДУ

$$\frac{du_1(t)}{dt} = f_1(t,u_1,u_2),$$

$$\frac{du_2(t)}{dt} = f_2(t,u_1,u_2), \quad a \le t \le b, \tag{169}$$

дополненную граничными условиями

$$u_1(a) = \mu_1, u_2(b) = \mu_2, \tag{170}$$

где μ_1, μ_2 – заданные числа.

Сделаем предположение, что $u_2(a) = C_0$ и решим одним из известных методов сформированную таким образом задачу Коши. В результате получим некоторое значение $u_2(b) = D_0$, отличающееся от заданного второго граничного условия (170). Введем функцию $g(C_i) = D_i - \mu_2$, характеризующую неточность проведенного решения.

Если бы мы сразу точно угадали значение $u_2(a)$, соответствующее данной граничной задаче, то функция g обратилась бы в ноль. Однако, благодаря произвольному выбору C_0, она в этой точке C_0 отлична от нуля и нам как раз необходимо найти ноль функции $g(C)$, который даст истинное значение C. Это можно сделать одним из методов нахождения корней, например, методом секущих, но для этого необходима вторая точка функции $g(C)$.

Изменим выбранное значение C_0 на небольшую величину Δ и решим такую задачу Коши. В результате получим новое значение $u_2(b) = D_i^{\Delta}$ и функции $g(C_0+\Delta) = D_0^{\Delta} - \mu_2$. Теперь можно применить метод секущих:

$$C_{i+1} = C_i - \frac{g(C_i)\cdot\Delta}{g(C_i+\Delta)-g(C_i)} = C_i - \frac{(D_i-\mu_2)\cdot\Delta}{D_i^{\Delta}-D_i}. \qquad (171)$$

Вычисления по этой рекуррентной формуле проводятся до тех пор, пока полученное приращение $C_{i+1}-C_i$ не станет меньше требуемой точности ε.

Обычно при уточнении значения C используют малозатратный метод невысокого порядка точности, а когда значение C найдено с требуемой точностью, задачу Коши решают методом Рунге–Кутта высокого порядка точности.

Описанный метод получил название метода стрельбы потому, что задание C_0 и построение соответствующей траектории напоминает пробный выстрел, а уточнение начального условия по формуле (171), соответствующее сведению к нулю функции $g(C)$, – серию выстрелов, постепенно приближающихся к цели.

Метод стрельбы может быть применим только к задачам, устойчивым по начальным условиям.

16.3 Разностный метод

Разностный метод основан на введении на отрезке [a,b] сетки (100), а в большинстве случаев – равномерной сетки (101), и замене всех входящих в уравнения производных $u''(x_i)$ разностными производными $u_{x,i}$ (120), $u_{\bar{x}x,i}$ (125) и т.д. Он приводит к замене системы m дифференциальных уравнений первого порядка (149) системой $m\cdot N$ алгебраических

уравнений относительно $m\cdot(N+1)$ неизвестных значений функций в N+1 точках сетки. Недостающие m уравнений дают граничные условия (168).

Нелинейные дефференциальные уравнения приводят к системе нелинейных алгебраических уравнений, а линейные – к СЛАУ.

Рассмотрим в качестве примера дифференциальное уравнение второго порядка (эквивалентное системе двух уравнений первого порядка)

$$u''(x) = -f(x) \tag{172}$$

при $x \in (a,b)$, дополненное условиями

$$u(a) = \mu_1, \quad u(b) = \mu_2. \tag{173}$$

Заменим $u''(x_i)$ второй разностной производной $u_{\bar{x}x,i}$ (125) и получим разностное уравнение

$$\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = -f_i. \tag{174}$$

Это уравнение можно записать для i = 1, 2, …, N–1, т.е. во всех внутренних точках сетки ω_h; общее количество неизвестных $N+1$.

В граничных точках в соответствии со (173) следует положить

$$u_0 = \mu_1, u_N = \mu_2. \tag{175}$$

Таким образом, (175) как раз и представляют два недостающих уравнения для определения $N+1$ неизвестных. Система уравнений (174), (175) называется *разностной схемой* или *разностной краевой задачей*, соответствующей исходной дифференциальной задаче (172)–(173). В дальнейшем, чтобы не было путаницы в обозначениях, будем через $u(t)$ обозначать решение дифференциальной задачи и через $y_i = y(t_i)$ – решение разностной задачи.

Итак, мы получили разностную схему

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} = -f_i, i = 1,2,\ldots,N-1, \tag{176}$$

$$y_0 = \mu_1, y_N = \mu_2,$$

решать далее которую можно методом прогонки (см. 9.3).

17 Уменьшение погрешностей вычисления

Как упоминалось в начале пособия и как было показано в ходе рассмотрения отдельных численных методов, все вычисления с их помощью производятся с погрешностью. Для ряда методов погрешность задается заранее пользователем, для других, в частности разностных и использующих сетки, приведены оценки погрешности. Если в последнем случае величина погрешности оказывается велика, то ее можно уменьшить путем измельчения сетки (уменьшения шага), но при этом пропорционально возрастет объем вычислений. Другой путь – это использование методов более высоких порядков, использующих на каждом шаге большее число узлов, что делает соотношения более громоздкими и также приводит к возрастанию объема вычислений. Усложняется также оценка точности получаемых результатов. Вместе с тем существует простой и эффективный способ уточнения решения при фиксированном числе узлов, используемых в аппроксимирующих конечно-разностных соотношениях.

17.1 Апостериорные оценки погрешности по Рунге

Рассматриваемый ниже метод можно применить ко всем рассмотренным ранее методам, априорную оценку погрешности которых можно записать через главный член погрешности в виде

$$R_0 = Ah^p, \tag{177}$$

где A – коэффициент, зависящий от метода и вида функции; h – шаг сетки; p – порядок метода. Зависимости (177) подчиняется главный член погрешности большинства методов численного интегрирования; численного дифференцирования; методов решения задачи Коши и разностных методов для граничной задачи для ОДУ.

Пусть вычисляется значение некоторой переменной w с шагом h, тогда

$$w = w_h + R,$$

где $R = R_0 + O\left(h^{p+1}\right)$ или

$$w = w_h + Ah^p + O(h^{p+1}), \tag{178}$$

где w_h – приближенное значение w; Ah^p – главный член погрешности; $O(h^{p+1})$ – величина порядка h^{p+1}, т.е. не содержащая членов порядка h^p, как главный член.

Вычислим ту же самую переменную w с шагом kh

$$w = w_{kh} + A(kh)^p + O((kh)^{p+1}), \qquad (179)$$

где коэффициент пропорциональности k может быть как больше, так и меньше единицы, например 0,5. Коэффициент A в выражениях (178) и (179) будет одинаковым, так как вычисляется одна и та же переменная, одним и тем же методом, а от величины шага h значение A не зависит.

Пренебрегая малыми по сравнению с h^p величинами, приравняем первые части соотношений (178) и (179) с учетом формулы (177) и получим

$$w_h + R_0 = w_{kh} + k^p R_0 .$$

Откуда найдем главный член погрешности

$$R_0 = \frac{w_h - w_{kh}}{k^p - 1}. \qquad (180)$$

Формула (180) называется первой формулой Рунге и позволяет путем двойного просчета величины w с шагами h и kh оценить погрешность. Так как оценка осуществляется после вычисления, то она является апостериорной. Формула (180) имеет большое практическое значение, так как позволяет провести оценку погрешности без изменения алгоритма используемого вычислительного процесса. При уменьшении шага h главный член погрешности R_0 будет стремиться к полной погрешности R.

После определения R_0 можно вычислить уточненное значение искомой величины

$$w_{уточн} = w_h + R_0 + O(h^{p+1}), \qquad (181)$$

с погрешностью на один порядок меньше, чем R_0. Последнее соотношение называют второй формулой Рунге.

Таким образом, формула Рунге дает более точное значение искомой величины. В общем случае порядок точности аппроксимации увеличивается на единицу.

Мы рассмотрели уточненные решения, полученные при двух значениях шага. Предположим теперь, что расчеты могут быть проведены с шагами $h_1, h_2, ..., h_q$. Тогда можно получить уточненное решение для искомой величины $w(x)$ по формуле Ромберга, которая имеет вид

$$w(x)=\begin{vmatrix} w(x,h_1) & h_1^p & h_1^{p+1} & .... & h_1^{p+q-2} \\ w(x,h_2) & h_2^p & h^{p+1}{}_2 & .... & h_2^{p+q-2} \\ ..... & ..... & ..... & ..... & ..... \\ w(x,h_q) & h_q^p & h_q^{p+1} & .... & h_q^{p+q-2} \end{vmatrix} \times \begin{vmatrix} 1 & h_1^p & h_1^{p+1} & .... & h_1^{p+q-2} \\ 1 & h_2^p & h_2^{p+1} & .... & h_2^{p+q-2} \\ ..... & ..... & ..... & ..... & ..... \\ 1 & h_q^p & h_q^{p+1} & .... & h_q^{p+q-2} \end{vmatrix} + \\ + O(h^{p+q-1}). \tag{182}$$

Таким образом, порядок точности взрастает на $q-1$. Заметим, что для успешного применения этой формулы исходная функция должна иметь непрерывные производные достаточно высокого порядка.

Формулы Рунге справедливы для всех вычислительных процессов, для которых выполняется степенной закон (177). Для определения порядка метода p необходимо проведение априорной оценки погрешности, что не всегда легко осуществить.

17.2 Апостериорное определение порядка метода по Эйткену

Английский математик Эйткен предложил способ оценки погрешности для случая, когда порядок p метода не известен. Более того, алгоритм Эйткена позволяет опытным путем определить и порядок метода. Для этого необходимо в третий раз вычислить значение величины w с шагом k^2h, т.е.

$$w = w_{k^2h} + A(k^2h)^p$$

или

$$w = w_{k^2h} + k^{2p}R_0. \tag{183}$$

Приравнивая правые части выражений (179) и (183), получим соотношение

$$w_{kh} - w_{k^2h} = R_0k^p(k^p - 1),$$

подставляя в которое значение R_0 из первой формулы Рунге (180) найдем

$$k^p = \frac{w_{kh} - w_{k^2h}}{w_h - w_{kh}}. \tag{184}$$

Полученное соотношение (184) совместно с первой формулой Рунге (180) позволяет оценить погрешность при использовании вычислительного

метода с неизвестным порядком p. Более того, порядок p можно определить, логарифмируя левую и правую части формулы (184)

$$p = \ln\left(\frac{w_{kh} - w_{k^2h}}{w_h - w_{kh}}\right) / \ln k. \qquad (185)$$

Для выбранного вычислительного процесса алгоритм Эйткена достаточно применить только один раз для определения порядка метода, а затем использовать формулу Рунге, требующую только двукратного вычисления искомой величины. Формулу (185) можно использовать для тестирования программ, реализующих вычислительные методы с известной априорной погрешностью. Априорный и апостериорный порядки должны получаться совпадающими для правильных программ. Конечно, это совпадение будет приближенным, так как при получении алгоритмов Рунге и Эйткена учитывались только главные члены погрешности.

17.3. Применение формулы Рунге для уменьшения объема вычислений

Возможность апостериорно оценивать погрешность позволяет вычислять интеграл (129) с заданной точностью $\varepsilon > 0$ путем автоматического выбора шага интегрирования h_i. Пусть используется составная квадратурная формула (132), причем на каждом частичном отрезке используется одна и та же квадратурная формула (например, формула трапеции, Симпсона и др.). Проведем на каждом частичном отрезке $[x_{i-1}, x_i]$ все вычисления дважды, один раз с шагом h_i и второй раз – с шагом $0,5h_i$, и оценим погрешность по первой формуле Рунге (180).

Если для заданного $\varepsilon > 0$ будут выполняться неравенства

$$\left|I_i - I_{h/2,i}\right| \approx \frac{\left|I_{h/2,i} - I_{h,i}\right|}{2^m - 1} \le \frac{\varepsilon h_i}{b-a}, \quad i = 1,2,...,N, \qquad (186)$$

то получим

$$\left|I - I_{h/2}\right| \le \frac{\varepsilon}{b-a}\sum_{i=1}^{N} h_i = \varepsilon,$$

т.е. будет достигнута заданная точность ε.

Если же на каком-то из частичных отрезков оценка (186) не будет выполняться, то шаг на этом отрезке надо измельчить еще в два раза и снова оценить погрешность. Измельчение сетки на данном отрезке следует

проводить до тех пор, пока не будет достигнута оценка вида (186). Заметим, что для некоторых подинтегральных функций $f(x)$ такое измельчение может продолжаться слишком долго. Поэтому в соответствующей программе следует предусмотреть ограничение сверху на число измельчений, а также возможность увеличения ε.

Таким образом, автоматический выбор шага интегрирования приводит к тому, что интегрирование ведется с крупным шагом на участках плавного изменения функции $f(x)$ и с мелким шагом – на участках быстрого изменения $f(x)$. Это позволяет при заданной точности ε уменьшить количество вычислений значений $f(x)$ по сравнению с расчетом на сетке с постоянным шагом. Подчеркнем, что для нахождения сумм $I_{h/2,i}$ не надо пересчитывать значения $f(x)$ во всех узлах, достаточно вычислять $f(x)$ только в новых узлах.

17.4 Метод Эйткена ускорения сходимости

Предположим, что какой-либо итерационный метод имеет линейную сходимость, т.е.

$x_k - x_* \approx aq^k, \quad q \in (0,1), \quad k = 1,2,...$

Числа a, q, x_* заранее неизвестны, но их можно найти, используя три последовательных итерации x_k, x_{k+1}, x_{k+2}. Составим уравнения

$x_k - x_* = aq^k, \quad x_{k+1} - x_* = aq^{k+1}, \quad x_{k+2} - x_* = aq^{k+2}$

(здесь равенства надо понимать как приближенные), из которых найдем

$\Delta x_k = x_{k+1} - x_k = aq^k(q-1),$

$\Delta^2 = \Delta x_{k+1} - \Delta x_k = aq^k(q-1)^2 = x_{k+2} - 2x_{k+1} + x_k,$

$$x_* = x_{k+2} - \frac{(\Delta x_{k+1})^2}{\Delta^2 x_k}. \qquad (187)$$

Метод Эйткена ускорения сходимости состоит в том, что после вычисления x_k, x_{k+1}, x_{k+2} производится пересчет по формуле

$$y_{k+1} = x_{k+2} - \frac{(\Delta x_{k+1})^2}{\Delta^2 x_k} \qquad (188)$$

и значение y_{k+1} принимается за новое приближение. Если бы равенство (187) выполнялось точно, то y_{k+1} совпало бы с точным решением x_*. В общем случае y_{k+1} дает лучшее приближение к x_*, чем очередная итерация

x_{k+2}. Подчеркнем, что главным предположением здесь является требование линейной сходимости основного итерационного метода. В случае методов, имеющих более высокую скорость сходимости (например, метод Ньютона), ускорение по Эйткену в форме (188) неэффективно.

На практике необязательно проводить пересчет по формуле (188) на каждой итерации k. Употребительны методы, в которых такой пересчет осуществляется циклически, т.е. через определенное число итераций.

С помощью метода Эйткена на основе известных итерационных методов можно получить иногда новые итерационные методы, обладающие высокой сходимостью. Рассмотрим, например, метод релаксации

$$\frac{x_{k+1}-x_k}{\tau}+f(x_k)=0, \tag{189}$$

который имеет линейную сходимость, если

$M_1 > f'(x) > 0, \quad 0 < \tau < 2/M_1.$

Предположим, что при некотором k получены значения x_k, x_{k+1}, x_{k+2}. Вычислим согласно (188) величину

$$y_{k+1}=x_{k+2}-\frac{(x_{k+2}-x_{k+1})^2}{x_{k+2}-2x_{k+1}+x_k} \tag{190}$$

и исключим из (190) с помощью (189) величины x_{k+1}, x_{k+2}. Имеем

$x_{k+1}=x_k-\tau f(x_k),$

$x_{k+2}=x_{k+1}-\tau f(x_{k+1})=x_k-\tau f(x_k)-\tau f(x_k-\tau f(x_k)),$

следовательно,

$$y_{k+1}=x_k-\tau\frac{f^2(x_k)}{f(x_k)-f(x_k-\tau f(x_k))}.$$

Проведенные построения позволяют предположить, что одношаговый итерационный метод

$$y_{k+1}=y_k-\tau\frac{f^2(y_k)}{f(y_k)-f(y_k-\tau f(y_k))} \tag{191}$$

обладает более быстрой сходимостью, чем исходный метод релаксации (12). Действительно, метод (191) при $\tau=1$ (метод Стеффенсена) имеет квадратичную сходимость.

Список литературы

1. Основы компьютерного моделирования наносистем / Ибрагимов И.М. , Ковшов А.Н., Назаров Ю.Ф. – М.: Изд-во «Лань» , 2010.- 384 с. ISBN *978-5-8114-1032-3*:

2. Поршнев С.В. Компьютерное моделирование физических процессов в пакете MATLAB. + CD. - М.: Изд-во «Лань», 2011.- 736 с. ISBN 978-5-8114-1063-7

3. Моделирование компонентов и элементов интегральных схем / Петров М.Н., Гудков Г.В. - М.: Изд-во «Лань», 2011.- 464 с. ISBN 978-5-8114-1075-0

4. Основы автоматизированного проектирования [Текст] : учебник для вузов / Е. М. Кудрявцев. - М. : Академия, 2011. - 304 с. - ISBN 978-5-7695-6004-0

5. Компьютерное моделирование и проектирование: учебное пособие / Ю. Р. Саликаев.- Томск: ТУСУР, 2012. - 100 с. Препринт. Режим доступа:: http://edu.tusur.ru/ training/ publications/

6. Численные методы в примерах и задачах : Учебное пособие для втузов / В. И. Киреев, А. В. Пантелеев. - 2-е изд., стереотип. - М. : Высшая школа, 2006. - 479 с. - (Прикладная математика для втузов). - ISBN 5-06-004763-6 :

7. Численные методы : Учебное пособие для втузов / В. Ф. Формалев, Д. Л. Ревизников ; ред. : А. И. Кибзун. - 2-е изд., испр. и доп. - М. : Физматлит, 2006. - 398 с. ISBN 5-9221-0737-2

8. Основы численных методов : Учебник для вузов / Валентин Михайлович Вержбицкий. - М. : Высшая школа, 2002. - 848 с. - ISBN 5-06-004020-8

9. Машинные методы анализа и проектирования электронных схем : / И. Влах, К. Сингхал ; пер.: А. Ф. Объедков, Н. Н. Удалов, В. М. Демидов ; ред. пер. А. А. Туркина. - М. : Радио и связь, 1988. - 560 с. - ISBN 5-256-00054-3

10. Очков В. Ф. Mathcad PLUS 6.0 для студентов и инженеров. - М.: Компьюетр-Пресс, 1996. – 238 с.

11. Mathcad для студента / А. М. Половко , И. В. Ганичев. - СПб. : БХВ-Петербург, 2006. - 336 с. - ISBN 5-94157-596-3

12 Mathcad 12 для студентов и инженеров / В. Ф. Очков. - СПб. : БХВ-Петербург, 2005. - 457 с. - ISBN 5-94157-289-1

Саликаев Ю.Р.

Компьютерное моделирование и проектирование

Учебное пособие

Усл. печ. л. Препринт
Томский государственный университет
систем управления и радиоэлектроники
634050, г.Томск, пр.Ленина, 40