

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего
образования «Крымский федеральный университет имени В.И. Вернадского»
Физико-технический институт (структурное подразделение)

Кафедра компьютерной инженерии и моделирования

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ
«АЛГОРИТМЫ И МЕТОДЫ ВЫЧИСЛЕНИЙ»**

для направлений подготовки
09.03.01 "Информатика и вычислительная техника",
09.03.04 «Программная инженерия»

квалификация бакалавр

Симферополь 2020

Лабораторный практикум по учебной дисциплине «Алгоритмы и методы вычислений» . – Симферополь: ФГАОУ ВО «Крымский федеральный университет имени В. И. Вернадского», 2020. – 128стр.

Разработчики - Милюков В.В., к.т.н., заведующий кафедрой компьютерной инженерии и моделирования, (л/р 1-5, 8), Горская И.Ю., старший преподаватель кафедры компьютерной инженерии и моделирования (л/р 6,7)

Утверждено на заседании кафедры компьютерной инженерии и моделирования, протокол от 15.01.2020г. № 5

Заведующий кафедрой компьютерной инженерии и моделирования _____ (Милюков В.В.)

Издается по решению Методического совета Физико-технического института ФГАОУ ВО «КФУ им. В.И. Вернадского»
(протокол №5 от 24.01.2020 г.).

Председатель методической комиссии _____ (А.Ф. Рыбась)

© Милюков В.В., Горская И.Ю., 2020

© ФГАОУ ВО «Крымский федеральный университет имени В.И. Вернадского», 2020

ТЕМЫ ЛАБОРАТОРНЫХ РАБОТ

(содержат обязательные задания 7 или 8 работ и дополнительные задания, выполнение которых заменяет соответствующую обязательную работу или добавляет дополнительные баллы к баллам обязательных работ)

№	Тема лабораторной работы	Обязательные задания компьютерная реализация (сумма до 90 баллов)	Дополнительные бонусные задания (компьютерная реализация с описанием)
1	Численное интегрирование	1) Метод Симпсона с контролем погрешности по формуле Рунге 2) Метод Гаусса - Кронрода, или Чебышева, или Монте-Карло	Адаптивный метод с использованием метода Симпсона, или Гаусса (10 баллов)
2	Решение СЛАУ	1) Метод Гаусса-Жордана с выбором ведущего элемента; 2) Итерационный метод (или Зейделя, или наискорейший спуск, или градиентный спуск, или случайных направлений)	Метод сопряженных направлений (например Флетчера-Ривса, +10 баллов), Метод регуляризации Тихонова для решения плохо-обусловленных СЛАУ(+10 баллов)
3	Решение нелинейных уравнений	1) Методы дихотомии + Ньютона, или золотое сечение + Ньютона, или секущих; 2) Решение СНАУ методом Ньютона-Рафсона	Решение нелинейной системы из 6 уравнений с 6 неизвестными (10 баллов) Обратная квадратичная интерполяция для поиска минимума функции (10 баллов)
4	Аппроксимация и интерполяция	1) Компьютерная реализация полинома Лагранжа 2) Компьютерная реализация - кубический сплайн	1) Сплайн Эрмита (10 баллов) 2) Кривые Безье (+5 баллов) 3) Устойчивый алгоритм численного дифференцирования (+5 баллов)
5	Аппроксимация тригонометрическими функциями. Ряды Фурье	1) Компьютерная реализация - разложение в ряд Фурье (коэффициенты разложения вычислять с помощью численного интегрирования)	Быстрое преобразование Фурье (+10 баллов)
6	Многомерная оптимизация	1) Компьютерная реализация - задача линейного программирования	Метод минимизации второго порядка (+10 баллов)

	непрерывных функций	2) Компьютерная реализация - минимизация нелинейных функционалов без ограничений (один из методов нулевого или первого порядка)	
7	Многомерная дискретная оптимизация	1) Компьютерная реализация - задача коммивояжера, или задача о рюкзаке, или задача о расписании. 2) Компьютерная реализация - методы ветвей и границ, или муравьиный алгоритм, или жадный алгоритм, или метод перебора	Генетический алгоритм (+10 баллов)
8	Дифференциальные уравнения в частных производных. Интегральные уравнения		Решение уравнения Лапласа на прямоугольнике, или одномерное уравнение теплопроводности (остывание шара, распространение тепла в стержне), или волновое уравнение (колебание струны) (+10 баллов) Емкость квадратной пластины, или синтез магнитного поля на оси соленоида (+10 баллов)

Лабораторная работа №1 ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

Ключевые понятия: метод прямоугольников, трапеций, Симпсона, Гаусса, Монте-Карло, правило Рунге, адаптивное интегрирование.

Перед выполнением лабораторной работы рекомендуем:

1. Изучить презентацию лектора курса: «Введение в вычислительную математику и физику», «Численное интегрирование (Численные квадратуры)», материалы в электронном виде доступны в "облаке" Mail.ru.
2. Прочитать соответствующие разделы в книгах:
Каханер Д., Моулдер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ.- М.: Мир, 1998.- 575 с.,
Бахвалов, Н.С. Численные методы : учебник / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков — 9-е изд. — Москва : Лаборатория знаний, 2020. — 636 с.
Мухамадиев. Конспект лекций по вычислительной математике, 2007.,

Х.Гулд, Я.Тобочник. Компьютерное моделирование в физике, в 2-ух томах.- М.:Мир.-1990г. (метод Моне-Карло).

Далее здесь использован термин численное интегрирование вместо термина «Приближенное вычисление определенных интегралов» с целью расширительного использования термина интегрирование, включающего вычисление площади методом Монте-Карло (классические квадратуры), и интегрирование функций, заданных табличными значениями.

Цель работы:

1. Изучить и научиться использовать на практике наиболее эффективные алгоритмы численного интегрирования.
2. Изучить приемы контроля точности вычисляемых интегралов и оптимизации быстродействия используемых алгоритмов.
3. Написать программу, реализующую два метода численного интегрирования в следующих комбинациях:
 - 1) Метод Симпсона с контролем погрешности по формуле Рунге;
 - 2) Метод Гаусса - Кронрода, или Чебышева, или Монте-Карло.Примеры 3 функций придумать самостоятельно, при этом одна функция обязательно должна быть периодической, а одна иметь выраженный пикообразный характер.
4. Указание: метод Симпсона должен быть реализован по схеме удвоения числа разбиений с контролируемой точностью, должно быть задано минимальное и максимальное число разбиений, требуемая абсолютная погрешность. Должны выводиться – фактическое число разбиений, фактическая погрешность по методу Рунге.

Дополнительные задания (бонусы):

- 1) Реализовать программу адаптивного интегрирования с плавающим числом разбиений в зависимости от скорости изменения значений функции.
- 2) Реализовать вычисление площади замкнутой фигуры методом Монте-Карло. Вычислить этим методом площадь Черного моря, взяв его изображение на Google maps. Оценить погрешность метода.

Содержание отчета:

1. Постановка задачи. Формульное описание алгоритма.
2. Описание компьютерной программы. Объяснение значений всех параметров.
3. Нахождение приближённого значения определённого интеграла по выбранным квадратурным формулам и оценка погрешности.
4. Оценка скорости работы алгоритма.
5. Сравнение эффективности различных алгоритмов вычисления определённых интегралов.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. Формула центральных прямоугольников (со средней точкой)

Пусть требуется вычислить определенный интеграл $J = \int_a^b f(x) dx$.

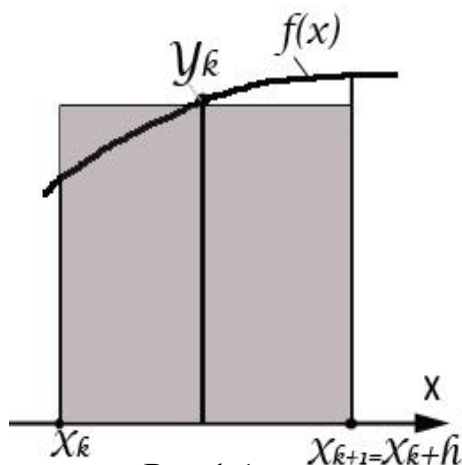


Рис.1.1

Разбиваем отрезок $[a, b]$ на n равных частей с границами: $x_0 = a$, $x_1 = a + h$, ..., $x_n = b$, где $h = (b - a) / n$.

На каждом частичном отрезке $[x_k, x_{k+1}]$, $(k = 1, \dots, n)$ подынтегральную функцию $f(x)$ заменяем ее значением в середине элементарного отрезка $y_k = f(x_k + \frac{h}{2})$,

$(k = 1, \dots, n)$, рис.1.1.

Таким образом, получаем приближённое равенство

$$\int_a^b f(x) dx \approx \sum_{i=1}^n y_k \cdot h = \sum_{i=1}^n f(a + i \cdot h + \frac{h}{2}) \cdot h, \text{ или более удобную}$$

окончательную формулу

$$J_n \approx \frac{b-a}{n} (y_1 + y_2 + \dots + y_n) \quad (1.1)$$

Отметим, что формула центральных прямоугольников обеспечивает такую же погрешность, как и метод трапеций, $R_n = -\frac{(b-a)^3}{12n^2} f''(\xi)$, где $a \leq \xi \leq b$, (точно интегрируется полином первой степени). Эта погрешность значительно меньше, чем в методах левых и правых прямоугольников.

2. Формула трапеций

Разбиваем отрезок $[a, b]$ на n равных частей с границами: $x_0 = a$, $x_1 = a + h$, ..., $x_n = b$, где $h = (b - a) / n$.

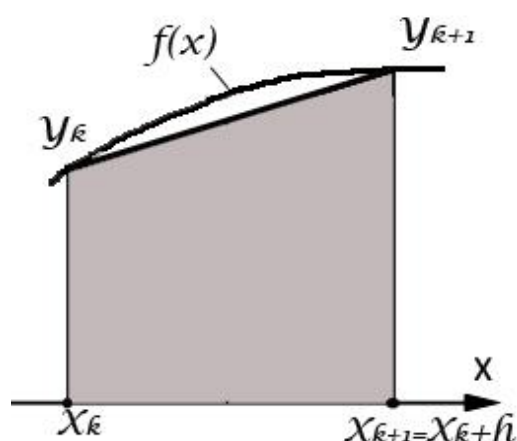


Рис.1.2

Обозначим $y_k = f(x_k)$ ($k = 0, 1, \dots, n$).

На каждом частичном отрезке $[x_k, x_{k+1}]$ ($k = 0, 1, \dots, n-1$) подынтегральную функцию $f(x)$ заменяем интерполяционным многочленом первой степени, построенным по двум узлам интерполяции (x_k и x_{k+1}), т.е. дугу графика подынтегральной функции

$y = f(x)$ заменяем стягивающей её хордой. Отсюда получаем приближённое

равенство
$$\int_{x_k}^{x_{k+1}} f(x) dx \approx \frac{b-a}{2n} (y_k + y_{k+1}).$$

Сумма интегралов по всем отрезкам дает общую (составную) квадратурную формулу трапеций

$$J_n \approx \frac{b-a}{2n} (y_0 + 2y_1 + 2y_2 + \dots + 2y_{n-1} + y_n), \quad (1.2)$$

здесь коэффициенты меняются по схеме : 1 - 2 - 2 - 2 ... 2 - 1

Погрешность этой формулы такая же, как и формулы центральных прямоугольников: $R_n = -\frac{(b-a)^3}{12n^2} f''(\xi)$, где $a \leq \xi \leq b$.

3. Формула Симпсона.

Правило Симпсона впервые было получено Кавальери в 1639 г., позже Джеймсом Грегори в 1668 г. и Томасом Симпсоном в 1743 г. Его же называют правилом парабол.

В методе Симпсона приближённое значение определённого интеграла J вычисляется с использованием квадратичной интерполяции.

Разбиваем отрезок $[a, b]$ на n равных частей (n - чётное).

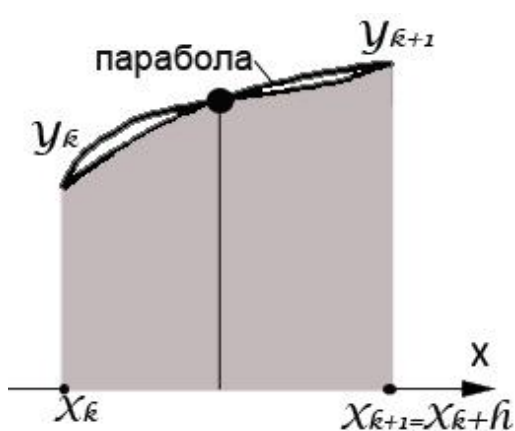


Рис.1.3

Рассмотрим отрезки $[x_0, x_2]$, $[x_2, x_4]$, ..., $[x_{n-2}, x_n]$. На каждом сдвоенном отрезке $[x_k, x_{k+2}]$ функцию $f(x)$ заменяем интерполяционным многочленом второй степени, построенным по трём узлам интерполяции (x_k, x_{k+1}, x_{k+2}) , т.е. дугу графика подынтегральной функции $y = f(x)$ на каждом сдвоенном отрезке заменяем параболой, проходящей через точки (x_k, y_k) ,

(x_{k+1}, y_{k+1}) , (x_{k+2}, y_{k+2}) , рис1.3.

Интерполирующий полином имеет вид:

$$f(x) = f(x_k) \cdot \frac{(x-x_{k+1})(x-x_{k+2})}{(x_k-x_{k+1})(x_k-x_{k+2})} + f(x_{k+1}) \cdot \frac{(x-x_k)(x-x_{k+2})}{(x_{k+1}-x_k)(x_{k+1}-x_{k+2})} + f(x_{k+2}) \cdot \frac{(x-x_k)(x-x_{k+1})}{(x_{k+2}-x_k)(x_{k+2}-x_{k+1})}.$$

Отсюда получаем приближённое значение интеграла:

$$\int_{x_k}^{x_{k+2}} f(x) \cdot dx = f(x_1) \cdot w_1 + f(x_2) \cdot w_2 + f(x_3) \cdot w_3,$$

где: $w_1 = h/6$, $w_2 = h \cdot 4/6$, $w_3 = h/6$.

Или окончательно:
$$\int_{x_k}^{x_{k+2}} f(x) dx \approx \frac{b-a}{3n} (y_k + 4y_{k+1} + y_{k+2}).$$

Сумма интегралов по всем двоянным частичным отрезкам дает общую (составную) формулу Симпсона:

$$J \approx \frac{b-a}{3n} [y_0 + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2}) + y_n], \quad (1.3)$$

здесь коэффициенты меняются по схеме : 1 - 4 - 2 - 4 ... 4 - 1, число разбиений отрезка (a,b) должно быть четным, а число точек нечетным.

Погрешность формулы Симпсона выражается через четвертую производную интегрируемой функции $R_n = -\frac{(b-a)^5}{180n^4} f^{(4)}(\xi)$, где $a \leq \xi \leq b$, то есть формула Симпсона точно интегрирует полиномы до третьей степени включительно.

4. Оценка погрешности квадратурных формул

Таблица 1. Сравнение методов

№	Название метода	Квадратурная формула	k
1	Левых прямоугольников	$J \approx \frac{b-a}{n} \sum_{i=0}^{n-1} y_i$	1
2	Правых прямоугольников	$J \approx \frac{b-a}{n} \sum_{i=1}^n y_i$	1
3	Центральных прямоугольников	$J \approx \frac{b-a}{n} \sum_{i=1}^n \tilde{y}_i$	2
4	Трапеций	$J \approx \frac{b-a}{2n} (y_0 + 2y_1 + 2y_2 + \dots + 2y_{n-1} + y_n)$	2
5	Симпсона (парабол)	$J \approx \frac{b-a}{3n} [y_0 + 4(y_1 + y_3 + \dots + y_{n-1}) + 2(y_2 + y_4 + \dots + y_{n-2}) + y_n]$	4

Одним из способов практической оценки погрешности дискретизации, которая возникает при применении численного интегрирования, является правило Рунге, заключающееся в последовательном увеличении (например, удвоении) числа узловых точек n и соответствующем уменьшении шага дискретизации h . Оценка по правилу Рунге имеет вид:

$$J - J_{h/2} \approx \frac{J_{h/2} - J_h}{2^k - 1}, \quad (1.4)$$

где J - точное, а $J_h, J_{h/2}$ - приближенные значения интегралов, полученные, соответственно, с шагом h и $h/2$, k - порядок точности метода.

Тогда при заданной точности ε величина h должна выбираться так, чтобы выполнялось условие $\Delta_{Рунге} = \frac{J_{h/2} - J_h}{2^k - 1} \leq \varepsilon$.

Например, за приближённое значение J интеграла, вычисленного по формуле Симпсона с поправкой по Рунге, можно принять

$$J = J_{2n} + (J_{2n} - J_n)/15. \quad (1.5)$$

Погрешность этого результата приближённо оценивают величиной

$\Delta = |J_{2n} - J_n|/15$. Для формул центральных прямоугольников и трапеций имеет место $J = J_{2n} + (J_{2n} - J_n)/3$, $\Delta = |J_{2n} - J_n|/3$.

Пример. Вычислим погрешность интегрирования функции $y=\sin(x)$ методом Симпсона¹

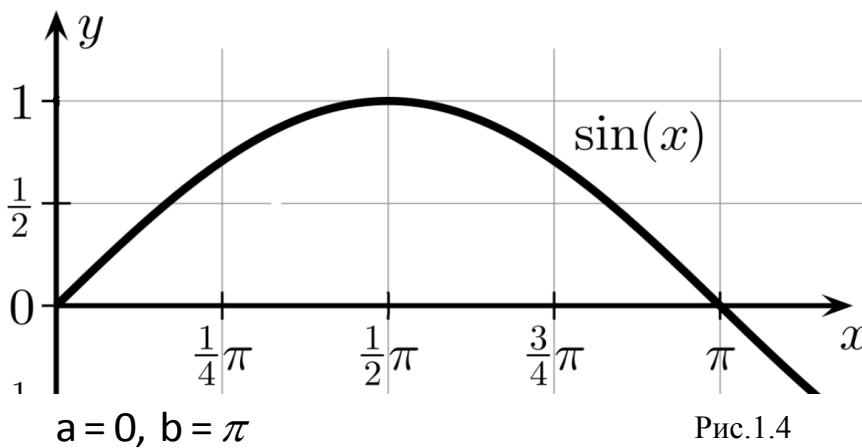


Рис.1.4

$$J_2 \approx \frac{\pi}{3 \cdot 2} [0 + 4 + 0] = 2.0944$$

Точное значение интеграла равно 2.0. Разобьем интервал $[0, \pi]$ по схеме удвоения на 4 отрезка (5 точек).

По формуле Симпсона имеем:

$$J_4 \approx \frac{\pi}{3 \cdot 4} \left[0 + 4 \cdot \frac{\sqrt{2}}{2} + 2 \cdot 1 + 4 \cdot \frac{\sqrt{2}}{2} + 0 \right] = 2.0045.$$

Вычисляем погрешность по формуле Рунге

$$\varepsilon_{2-4} = (2.0045 - 2.0944) / 15 = -0.006.$$

Видим, что истинная погрешность 0.0045 находится в пределах погрешности Рунге. Предсказываем результат интегрирования $J_{np} \approx 2.0045 - 0.006 = 1.9985$.

Продолжая, получим: $J_8 \approx 2.000269$, $\varepsilon_{4-8} = -0.0004$

и.т.д.

¹ Советуем подробно изучить этот простой и наглядный пример!

5. Квадратурная формула Гаусса.

До сих пор мы рассматривали квадратурные формулы на равномерной сетке. При использовании неравномерной сетки при специальном расположении узлов точность интегрирования может быть существенно повышена. Рассмотрим функцию $f(t)$, заданную на интервале $[-1, 1]$. Квадратурная формула Гаусса в этом случае имеет вид:

$$\int_{-1}^1 f(t) dt \approx \sum_{k=1}^n A_k \cdot f(t_k). \quad (1.6)$$

Коэффициенты A_k и точки t_k выбираем так, чтобы формула была точной для всех многочленов наивысшей возможной степени N . Доказано, что такие числа A_k , t_k определяются однозначно при $N = 2n - 1$. Точки t_k , используемые в квадратурной формуле, являются корнями полинома Лежандра степени n :

$$P_n(t) = \frac{1}{2^n n!} \frac{d^n (t^2 - 1)^n}{dt^n}.$$

Доказано, что $P_n(t)$ имеет на интервале $[-1, 1]$ n действительных различных корней. Значения A_k и t_k ($k = 1, 2, \dots, n$) для различных n табулированы (ниже приведены коэффициенты для 16 точечной схемы).

При вычислении интеграла с произвольными пределами $[a, b]$ делаем замену переменной $x = \frac{b+a}{2} + \frac{b-a}{2}t$.

В этом случае квадратурная формула Гаусса принимает вид:

$$\int_a^b f(x) \cdot dx \approx \frac{b-a}{2} \sum_{k=1}^n A_k \cdot f(x_k) \quad (1.7)$$

где $x_k = \frac{b+a}{2} + \frac{b-a}{2}t_k$, ($k = 1, 2, \dots, n$); A_k и t_k ($k = 1, 2, \dots, n$) - табулированные значения. Приводим эти значения для $n = 6$:

$$t_{1,6} = \pm 0.932470; t_{2,5} = \pm 0.661209; t_{3,4} = \pm 0.238619$$

$$A_1 = A_6 = 0.171324; A_2 = A_5 = 0.360762; A_3 = A_4 = 0.467914.$$

В приложении 2 приведен текст программы, реализующий 16 точечную формулу Гаусса, позволяющую точно вычислять интегралы от полиномов до 31 степени. Близок по идеологии к методу Гаусса и метод Чебышева, проведенные нами численные эксперименты показывают, что он не имеет преимуществ перед методом Гаусса.

Приводим формулы для узлов метода Чебышева для 2, ..., 7 точечной

схемы. Коэффициенты записаны в виде двумерного массива данных:

Двухточечная схема квадратурной формулы Чебышева	-0.577350; 0.577350;
Трехточечная схема квадратурной формулы Чебышева	-0.707107; 0; 0.707107;
Четырехточечная схема квадратурной формулы Чебышева	-0.794654; -0.187592; 0.187592; 0.794654;
Пятиточечная схема квадратурной формулы Чебышева	-0.832498; -0.374541; 0; 0.374541; 0.832498;
Шеститочечная схема квадратурной формулы Чебышева	-0.866247; -0.422519; -0.266635; 0.266635; 0.422519; 0.866247;
Семиточечная схема квадратурной формулы Чебышева	-0.883862; -0.529657; -0.323912; 0; 0.323912; 0.529657; 0.883862;

В настоящее время одним из наиболее эффективных методов численного интегрирования считается комбинированный метод Гаусса-Кронрода, например (G7, K15), семиточечная формула Гаусса и пятнадцатиточечная формула Кронрода . Проведенные нами численные эксперименты не показали преимуществ (G7, K15) перед парой (G8, G16).

Проведенные нами многочисленные эксперименты для различных функций (в том числе для "пикообразных") показывают, что самые надежные и эффективные программы численного интегрирования получаются при использовании пары (G8, G16) в сочетании с адаптивным квадратурным алгоритмом. Вопрос интегрирования функции, заданной своими табличными значениями рассмотрен в разделе интерполяция.

6. Квадратурный метод Монте-Карло

Каким образом можно с помощью кучи камней измерить площадь пруда? Предположим, что пруд расположен в центре поля известной площади S . Бросайте камни произвольным образом так, чтобы они падали в случайных точках в пределах поля, и считайте количество всплесков при попадании камней в пруд. Площадь пруда приблизительно равна площади поля, умноженной на долю камней, попавших в пруд. Эта простая процедура является примером метода Монте-Карло. Термин «метод Монте-Карло» был введен во время второй мировой войны фон Нейманом и Уламом в Лос-Аламосе в связи с моделированием нейтронной диффузии в расщепляемом материале. Окончательно название утвердилось благодаря Нику Метрополису.

В начале 30-х годов в Римском университете Энрико Ферми, воспользовавшись этой идеей и проведя расчеты вручную, предсказал результаты экспериментов, чем изумил своих коллег.

Метод «Монте-Карло» - общее название группы методов, основанных на получении большого числа реализаций стохастического процесса, так, что вероятностные характеристики совпадают со свойствами модели.

На рис. 1.4 представлен пример вычисления площади Симферопольского



Рис.1.4

водохранилища методом Монте-Карло.

При числе бросков $n=1000000$ в двух сериях испытаний получен результат $S=1473,5$ тыс.кв.м. ± 600 кв.м.

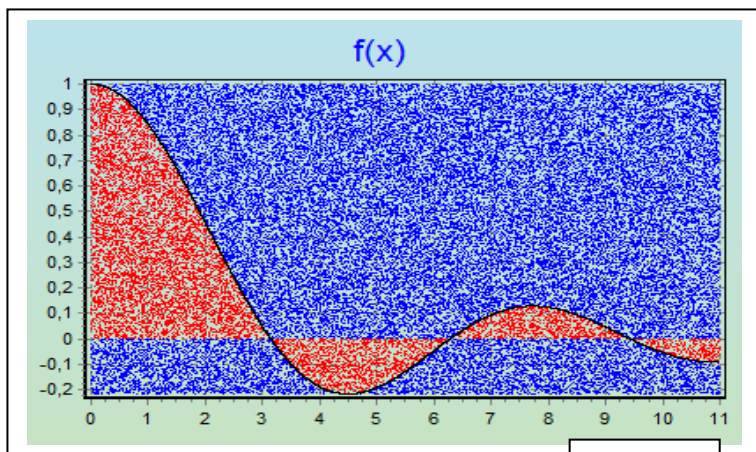


Рис.1.5

Погрешность метода
вычислялась в виде
стандартного отклонения
средних:

$$\sigma = \sqrt{\langle S_k^2 \rangle - \langle S_k \rangle^2},$$
$$\langle S_k^2 \rangle = \frac{1}{k} \sum_1^k S_m^2, \quad \langle S_k \rangle = \frac{1}{k} \sum_1^k S_m$$

Замечание 1. При программировании метода Монте-Карло для вычисления одномерного интеграла следует быть аккуратным в случае, когда минимальное и максимальное значения функции имеют разные знаки. При случайных бросках и попаданиях под график функции следует учитывать знаки (при отрицательных значениях функции нужно не суммировать число попаданий, а вычитать, общее число попаданий может быть отрицательным числом).

Замечание 2. Метод Монте-Карло обладает медленной сходимостью в зависимости от числа бросков, погрешность обратно пропорциональна корню из N , однако этот метод незаменим при вычислении многомерных интегралов, часто возникающих при компьютерном моделировании в теоретической физике. Дело в том, что обычные квадратурные формулы в этом случае не работают из-за катастрофически быстрого нарастания числа разбиений.

7. Алгоритм адаптивного интегрирования

Предлагаем вашему вниманию алгоритм адаптивного интегрирования, основанный на применении рекурсии и делении отрезков интегрирования пополам, (или удвоении, в зависимости от ситуации). Алгоритм может быть основан на паре Симпсона, например (S2, S4), или на комбинированном методе Гаусса-Кронрода, например (G7, K15), или на паре Гаусса, например (G8, G16), (в скобках указано число разбиений отрезка интегрирования для каждой пары).

Описание программы адаптивного интегрирования на примере метода Симпсона (S2, S4):

Входные параметры:

a, b-пределы интегрирования,

x_1, x_2 – плавающие пределы, при первом запуске $x_1=a, x_2=b$, далее, при рекурсии это уже другие значения,

eps – требуемая абсолютная погрешность (можно задавать относительную погрешность, но могут возникнуть проблемы при интегрировании знакопеременных функций),

выходной параметр S-значение интеграла, должно быть глобальным параметром и занулено перед первым запуском.

1 шаг: вычисляем два значения интеграла, например пару Симпсона (S2, S4).

2 шаг: проверяем значение погрешности (для метода Симпсона если

$\frac{|s_4-s_2|}{15} < eps$, если при этом достигли правого края, $x_2=b$, то считаем, что интеграл вычислен, выходим из программы,

3 шаг: если правый край не достигнут, но точность достигнута, накапливаем результат и переносим отрезок интегрирования вправо, удваиваем шаг и проверяем, чтобы не выйти за пределы (a, b), после этого рекурсия к программе,

4 шаг: если квадратурная пара не достигает точности на отрезке, то делим

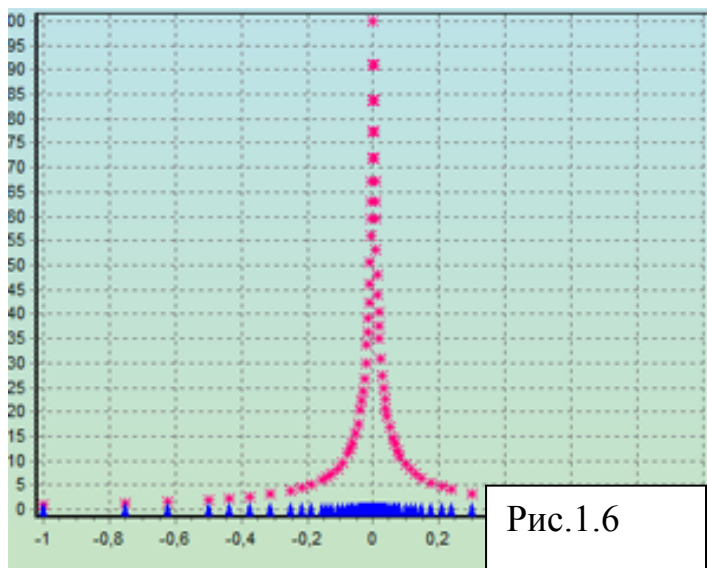


Рис.1.6

отрезок пополам и исследуем левую половину отрезка, затем снова рекурсия!

На рис. 1.6 приведен пример интегрирования функции $1/(|x| + 0.01)$.

Использован классический метод Симпсона, который для достижения реальной погрешности $3 \cdot 10^{-7}$ использовал 2048 разбиений и уточняющую формулу.

Время вычисления интеграла 0,12 мс. Адаптивный метод Симпсона достиг погрешности $1 \cdot 10^{-7}$, используя 77 отрезков (учитывая 4-точечную схему, 308 отрезков). Время вычисления интеграла 0,06 мс. Выигрыш по времени не такой внушительный из-за «ложных» ходов, которые есть в адаптивном методе.

Адаптивный метод Гаусса, пара (G8, G16) для достижения меньшей погрешности $8 \cdot 10^{-9}$ использовал всего лишь 12 разбиений, рис.1.7, (учитывая 16-точечную схему, 192 точки). Время вычисления интеграла 0,02 мс.

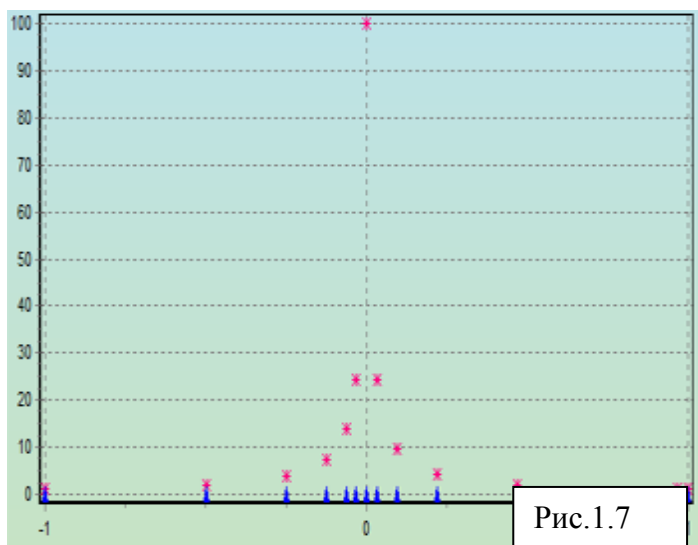


Рис.1.7

Численные эксперименты показывают, что адаптивный метод Гаусса (G8, G16) почти всегда имеет заметное преимущество перед методом Симпсона, а адаптивный метод Симпсона дает выигрыш перед классическим методом только при интегрировании функций с острыми пиками.

Кроме того, выявлено, что адаптивный метод Симпсона

плохо пригоден для интегрирования периодических функций. Иногда он останавливается, преждевременно решив, что точность достигнута. С адаптивным методом Гаусса этого не происходит из-за сложного расположения узлов.

8. Заключение

Для практического использования алгоритмов численного интегрирования рекомендуем использовать метод Симпсона с удвоением шага, контролем

погрешности по правилу Рунге и предсказанием результата с учетом величины и знака погрешности.

В сложных случаях, когда функция быстро изменяется или имеет пикообразный характер, рекомендуем использовать адаптивный метод Гаусса, например в сочетании (G8, G16).

Наконец, если значения функции заданы таблично на неравномерной сетке, то необходимо сначала осуществить интерполяцию значений функции, например с помощью кубического сплайна. После вычисления коэффициентов сплайна интеграл может быть легко вычислен аналитически.

Приложение 1.1. Фрагмент программного кода на языке Pascal, реализующего метод Симпсона:

```
Function FixSimpson(a,b:real;N:integer):real;  
// a,b-пределы интегрирования, N- число разбиений интервала  
var i:integer; xi,h,h2,s,p,v:real;  
begin  
N:=N div 2 *2;    // N - должно быть четным, число точек N+1  
  h:=(b-a)/N;      //размер отрезка  
  h2:=h*2;          // двойной интервал  
  v:=0;   p:=0;     //обнуляем сумматоры  
  s:=f(a)+f(b);    // отдельно суммируем для крайних точек  
  xi:=a+h;  
while xi<b do begin    //первый цикл для точек с коэфф. 4  
  p:=p+f(xi);  
  xi:=xi+h2;  
  end;  
xi:=a+h2;  
while xi<b-h do begin  //второй цикл для точек с коэфф. 2  
  v:=v+f(xi);  
  xi:=xi+h2;  
  end;  
FixSimpson:=h/3*(s+4*p+2*v); end;
```

Замечание. В приведенном фрагменте программы погрешность метода никак не контролируется, поэтому программу следует доработать. Если стремиться к созданию эффективной и быстро работающей программы для практических вычислений, то необходимо сделать алгоритм гибким, чтобы он анализировал и контролировал погрешность, а также наращивал число разбиений только в случае, если мы ему явно укажем на необходимость этой процедуры.

В общем случае список параметров функции вычисления интеграла может выглядеть так: (a, b , accuracy , N0 , Nmax , flagR , errRez , N)

где: входные параметры:

a , b – пределы интеграла,

accuracy – заданный порог абсолютной погрешности вычислений интеграла,
 N0- начальное число отрезков в методе Симпсона (2, 4, 8 и т.д.)
 Nmax – максимально разрешенное число отрезков, Nmax >=N0, если ,
 Nmax=Nmin, то программа считает один раз при N=Nmin.

выходные параметры:

accuracyRez – достигнутое значение абсолютной погрешности,
 N – реальное использованное число разбиений.

***Приложение 1.2. Фрагмент программного кода на языке Pascal,
 реализующего квадратурную формулу Гаусса (16 точечную)***

```
function GS_int ( a,b,err_rel:real;var ierr:integer) :real;
{ вычисление опр. интеграла с помощью квадратур Гаусса,(16
точек) a,b - пределы интегрирования (a>b),
  err_rel - допустимая относительная погрешность
    (верных знаков примерно 1/err_rel),
  ierr - критерий ошибки на выходе,
    ( 0 - точность достигнута, 1 - нет ),
    f(x) - функция одной переменной }
var N,n1,N2,k                                     :integer;
var h,x,x0,dx,ds,ba,Fab,Sold,S,gs                :real;
const
t:array[1..30] of real = (
-0.577350269189626,0.577350269189626,  { 2 }
-0.861136311594053,-0.339981043584856,
  0.339981043584856, 0.861136311594053, { 4 }
-0.960289856497536,-0.796666477413627,
-0.525532409916329,-0.183434642495650,
  0.183434642495650, 0.525532409916329,
  0.796666477413627, 0.960289856497536, { 8 }
-0.989400934991650,-0.944575023073233,
-0.865631202387832,-0.755404408355003,
-0.617876244402644,-0.458016777657227,
-0.281603550779259,-0.095012509837637,
  0.095012509837637, 0.281603550779259,
  0.458016777657227, 0.617876244402644,
  0.755404408355003, 0.865631202387832,
  0.944575023073233, 0.989400934991650); { 16}
w:array[1..30] of real = (
1.0                                     ,1.0,                                { 2 }
0.347854845137454,0.652145154862546,
0.652145154862546,0.347854845137454,  { 4 }
0.101228536290376,0.222381034453374,
0.313706645877887,0.362683783378362,
0.362683783378362,0.313706645877887,
```



```

0.222381034453374,0.101228536290376,    { 8 }
0.027152459411754,0.062253523938648,
0.095158511682493,0.124628971255534,
0.149595988816577,0.169156519395003,
0.182603415044924,0.189450610455069,
0.189450610455069,0.182603415044924,
0.169156519395003,0.149595988816577,
0.124628971255534,0.095158511682493,
0.062253523938648,0.027152459411754);    { 16 }
begin;
  ba:=b-a;
  x0:=(a+b)/2;
  dx:=(b-a)/2;
  Sold:=f(x0)*ba;
  N:=1;
  N1:=1;
  repeat
    N:=N*2;
    N2:=N1+N-1;
    S:=0;{ занулить сумматор }
  for k:=N1 to N2 do begin
    x:=x0+dx*t[k];
    s:=s+f(x)*w[k];
  end;    { цикл по к }
  gs:=s*dx; { врем. знач. интеграла }
  ds:=abs(1-Sold/gs); { отн. измен. результата }
  if ds < err_rel/2 then begin
    ierr:=0;{ точность достигнута }
    GS_int:=gs;
    exit;
  end; { if - проверки на погрешность }
  Sold:=gs;
  N1:=N1*2+1;
  until N>=16;{ делать, пока N<16}
  ierr:=1;
  GS_int:=gs;
  end;{GS_int}

```

Лабораторная работа №2

РЕШЕНИЕ СИСТЕМ

ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ)

Ключевые понятия: матрица СЛАУ, определитель, обусловленность, совместность, определенность СЛАУ, псевдорешение, факторизация, методы Гаусса, Гаусса-Жордана, Зейделя, наискорейшего спуска, градиентного спуска, овражный метод, сопряженных градиентов, регуляризации.

Перед выполнением лабораторной работы рекомендуем:

3. Изучить презентацию лектора курса: «Решение систем линейных алгебраических уравнений (СЛАУ)», материалы доступны "облаке" Mail.ru.
1. Прочитать соответствующие разделы книги:
Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ. - М.: Мир, 1998. - 575 с.,
Фаддеев Д.К., Фаддеева В.Н. Вычислительные методы линейной алгебры. - М., Л.: «Наука» .- 1963г.,
А.Н.Тихонов и др. Регуляризирующие алгоритмы и априорная информация .- М.: «Наука» .- 1983.

Цель работы:

1. Изучить и научиться использовать на практике наиболее эффективные прямые и итерационные алгоритмы решения СЛАУ.
2. Написать программу, реализующую два метода численного решения СЛАУ в следующих комбинациях: первый метод – метод исключения Гаусса с выборкой ведущего элемента, (или Гаусса-Жордана), второй метод - Гаусса-Зейделя, или метод группы градиентного (наискорейшего) спуска.
3. Пример выбрать самостоятельно, матрицу можно заполнить случайными числами. Для предотвращения расходимости метода Зейделя необходимо усилить диагональ матрицы.
4. Дополнительные задания (бонусы к оценке):
 - 1) Реализовать метод сопряженных градиентов или переменных направлений.
 - 3) Реализовать метод регуляризации Тихонова для решения плохо-обусловленных СЛАУ.

Содержание отчета:

1. Постановка задачи.
2. Краткое описание метода решения СЛАУ.
3. Листинг программы с подробными комментариями.
4. Результаты решения и проверки решения СЛАУ.

1. Основные положения теории СЛАУ

Предположим, что задана система m линейных уравнений относительно n неизвестных x_1, x_2, \dots, x_n . В общем случае такую систему можно записать в следующем виде:

[illegible]

Сведем коэффициенты при неизвестных в системе (2.1) в матрицу

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Эта матрица содержит, очевидно, m строк и n столбцов и называется *матрицей системы*. Если число уравнений в системе (2.1) равно числу неизвестных ($m = n$), то матрица системы называется *квадратной*, в противном случае – *прямоугольной*.

Используя понятие произведения матриц, систему (2.1) можно кратко записать в матричной форме:

$$A \cdot X = B, \quad (2.2)$$

где A – матрица системы,

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \text{ — вектор-столбец из неизвестных;}$$

$$B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \text{ — вектор-столбец свободных членов.}$$

Система (2.1) называется *совместной*, если она имеет, по крайней мере, одно решение. В противном случае система называется *несовместной*. Совместная система может обладать единственным решением, но может иметь

и более одного решения (в том числе и бесконечное множество решений). В том случае, когда СЛАУ имеет единственное решение, она называется *определённой*, в противном случае – *неопределённой*. Так, например, совместная СЛАУ, у которой число уравнений меньше числа неизвестных ($m < n$), является неопределённой. СЛАУ, у которой число уравнений больше числа неизвестных ($m > n$), является *переопределённой*.

Переопределённая СЛАУ чаще всего несовместна, в таком случае отыскивают её так называемое *псевдорешение* – приближённое решение, которое представляет собой такой вектор переменных, который наилучшим образом удовлетворяет всем уравнениям системы (минимизирует функционал невязки решения, или более сложный функционал, обеспечивающий не только уменьшение невязки, но и гладкость решения).

На практике встречаются СЛАУ с квадратной матрицей при $n > 1000$, положительно определенные, переопределённые, ленточные при $n > 10000$, разреженные, плохо обусловленные. При этом традиционный подход, основанный на вычислении определителя СЛАУ непригоден даже при $n = 20$. Современному компьютеру с производительностью 3 Гигафлопс для решения СЛАУ для $n = 20$ по формуле Крамера требуется 1000 лет.

На практике вместо определителя используют понятие обусловленности матрицы СЛАУ.

$$cond(A) = \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|}, \text{ здесь чаще всего используются квадратичные нормы векторов.}$$

Влияние обусловленности на погрешность решения можно выразить следующим неравенством,

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A) \cdot \frac{\|\Delta b\|}{\|b\|} \quad (2.3)$$

которое показывает, что погрешность задания правой части уравнения (также как и погрешность вычисления коэффициентов матрицы), может усиливаться при вычислении вектора решения в cond раз.

Рассмотрим пример:

$$A = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix} \quad b = \begin{pmatrix} 4.1 \\ 9.7 \end{pmatrix} \quad x = \begin{pmatrix} 1.0 \\ 0 \end{pmatrix}$$

Обусловленность матрицы с использованием квадратичной нормы векторов равна $\text{cond}=2200$.

При внесении незначительной погрешности в правую часть ($b_1 = 4,11$ вместо 4,10), получаем совершенно другое решение:

$$A = \begin{pmatrix} 4.1 & 2.8 \\ 9.7 & 6.6 \end{pmatrix} \quad b = \begin{pmatrix} 4.11 \\ 9.70 \end{pmatrix} \quad x = \begin{pmatrix} 0.34 \\ 0.97 \end{pmatrix}$$

Точные (прямые) методы решения СЛАУ

Метод решения СЛАУ называют точным (прямым), если он позволяет получить решение после выполнения конечного числа элементарных операций. К прямым методам относят метод Крамера, метод Гаусса, метод Холецкого и другие. Основным недостатком прямых методов является то, что для нахождения решения необходимо выполнить большое число операций. Этот недостаток особенно проявляется при необходимости решать большие системы с заполненными матрицами. Кроме того, прямые методы, как правило, неприменимы для решения плохо обусловленных систем и не являются точными из-за ошибок округлений даже для обычных систем. Решение, полученное прямым методом может быть уточнено итерационным методом (например градиентным методом, который является не ухудшающим).

Метод исключения (Гаусса)

Один из старейших численных методов - это метод последовательного исключения неизвестных, называемый обычно именем Гаусса. В конце 40-х годов 20 века, вскоре после изобретения электронных вычислительных машин, этот метод одним из первых был проанализирован на предмет его поведения в арифметике конечной разрядности. Результаты, полученные Джоном фон

Нейманом и другими авторами, оценивались как пессимистические, отчасти вследствие технической сложности, но также потому, что их подход подчеркивал некоторые худшие аспекты метода. Гауссово исключение утратило популярность. Но примерно около 1960-го года, главным образом благодаря деятельности Дж. Уилкинсона, обнаружилось, что метод представляет собой почти идеальный алгоритм. Он дает решения системы, не худшие, чем мог бы дать любой другой мыслимый алгоритм. С этих пор гауссово исключение заняло центральное положение в численном анализе.

Современные исследования, относящиеся к гауссову исключению, вскрыли важность двух идей: необходимости выбора главного элемента и правильной интерпретации ошибок округления.

Идея метода состоит в последовательном исключении неизвестных.

Пусть дана система уравнений

[illegible]

Процесс решения по методу Гаусса состоит из двух этапов. На первом этапе (прямой ход) система приводится к треугольному виду:

[illegible]

где верхний индекс в круглых скобках означает номер шага исключения.

На втором этапе (обратный ход) идет последовательное определение неизвестных из этой ступенчатой системы.

Прямой ход.

Положим $a_{11} \neq 0$, если $a_{11} = 0$, то первым в системе запишем уравнение, в котором $a_{11} \neq 0$.

Расставим уравнения системы таким образом, чтобы коэффициент при x_1 имел наибольшее значение (другими словами отсортируем систему по убыванию).

Преобразуем систему, исключив неизвестное x_1 во всех уравнениях, кроме первого (используя элементарные преобразования системы). Для этого умножим обе части первого уравнения на $-\frac{a_{21}}{a_{11}}$ и сложим почленно со вторым уравнением системы. Затем умножим обе части первого уравнения на $-\frac{a_{31}}{a_{11}}$ и сложим с третьим уравнением системы. Продолжая этот процесс, получаем систему

[illegible]

Здесь $a_{ij}^{(1)}, b_{ij}^{(1)}$ ($i, j = \overline{2, m}$) - новые значения коэффициентов и правых частей, которые получаются после первого шага.

Аналогичным образом, считая главным элементом $a_{22}^{(1)} \neq 0$, исключим неизвестное x_2 из всех уравнений системы, кроме первого и второго, и т.д. Продолжаем этот процесс пока это возможно.

Если в процессе приведения системы к треугольному виду появятся нулевые решения (равенства вида $0=0$) их отбрасывают. Если же появится уравнение вида $0=b_i$, а $b_i \neq 0$, то это говорит о несовместимости системы.

Весь алгоритм можно компактно записать в матричных обозначениях. Он соответствует разложению матрицы A в произведение более простых матриц: $A = PLU$, где L и U -треугольные матрицы, а P - матрица перестановки, хранящая информацию о переупорядочении строк.

Второй этап (обратный ход) заключается в решении треугольной системы. В последнем уравнении этой системы выражаем первое неизвестное

x_k через остальные неизвестные (x_{k+1}, \dots, x_n). Затем подставляем значение x_k в предпоследнее уравнение системы и выражаем x_{k-1} через (x_{k+1}, \dots, x_n), затем находим x_{k-2}, \dots, x_1 .

Пример решения СЛАУ методом Гаусса:

$$\left. \begin{array}{rcl} x_1 + 5 \cdot x_2 - x_3 & = & 2 \\ x_1 & + & 2 \cdot x_3 = -1 \\ 2 \cdot x_1 - x_2 - 3 \cdot x_3 & = & 5 \end{array} \right\}$$

После вычитания первой строки, умноженной на соответствующие коэффициенты (1 и 2), из второй и третьей строк, получим:

$$\left. \begin{array}{rcl} x_1 + 5 \cdot x_2 - x_3 & = & 2 \\ - 5 \cdot x_2 + 3 \cdot x_3 & = & -3 \\ - 11 \cdot x_2 - x_3 & = & 1 \end{array} \right\}$$

После вычитания из третьей строки второй строки, умноженной на -11/5, получим:

$$\left. \begin{array}{rcl} x_1 + 5 \cdot x_2 - x_3 & = & 2 \\ - 5 \cdot x_2 + 3 \cdot x_3 & = & -3 \\ - 38/5 \cdot x_3 & = & 38/5 \end{array} \right\}$$

Получили треугольную матрицу. Выполняем обратный ход:

$$x_3 = -1, -5 \cdot x_2 + 3 \cdot x_3 = -3 \Rightarrow x_2 = (3 + 3 \cdot x_3) = (3 + 3 \cdot (-1)) = 0,$$

$$x_1 + 5 \cdot x_2 - x_3 = 2 \Rightarrow x_1 = 2 + 5 \cdot x_2 + x_3 = 2 + 5 \cdot 0 + (-1) = 1.$$

Программная реализация приведена в приложении.

Итерационные методы решения СЛАУ

Достоинства итерационных методов:

1. Погрешность округления не накапливается от итерации к итерации. Например, градиентные методы при оптимальном выборе шага могут только уменьшать невязку решения, но не увеличивать.
2. Число требуемых итераций обычно меньше n , поэтому общее число действий меньше n^3 , т.е. меньше, чем в методе исключений Гаусса.
4. Итерационные методы особенно выгодны для систем с большим количеством нулевых коэффициентов (систем с разреженной матрицей).

тоды исключения наоборот: чем больше нулей, тем чаще требуется
 бирать новую рабочую строку.

Некоторые итерационные методы обладают саморегуляризирующими свойствами (не позволяют возникать осцилляциям и обеспечивают гладкость решения).

Метод простой итерации

Рассмотрим систему линейных алгебраических уравнений

[illegible]

Приведём систему к виду

$$\left. \begin{aligned} x_1 &= c_{11}x_1 + c_{12}x_2 + \dots + c_{1n}x_n + f_1, \\ x_2 &= c_{21}x_1 + c_{22}x_2 + \dots + c_{2n}x_n + f_2, \\ &\dots\dots\dots, \\ x_n &= c_{n1}x_1 + c_{n2}x_2 + \dots + c_{nn}x_n + f_n, \end{aligned} \right\}$$

В матричном виде полученную систему можно записать следующим образом:

$$X = CX + F, \quad (2.4)$$

$$\text{где } X = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}, C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix}, F = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{pmatrix}.$$

За нулевое приближение решения системы возьмём произвольный вектор

$$X^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \dots \\ x_n^{(0)} \end{pmatrix}$$

Строим векторы $X^{(1)} = CX^{(0)} + F$, $X^{(2)} = CX^{(1)} + F$ и т.д., т.е $X^{(k+1)} = CX^{(k)} + F$, $k = 0, 1, 2, \dots$.

Таким образом, методом простой итерации приближение $X^{(k+1)}$ вычисляется по предыдущему приближению $X^{(k)}$ путём подстановки компонент $X^{(k)}$ в правую часть уравнений системы:

$$x_i^{(k+1)} = \sum_{j=1}^n c_{ij} x_j + f_i; \quad i = 1, \dots, n; \quad k = 0, 1, 2, \dots \quad (2.5)$$

Если последовательность векторов $X^{(k)}$ имеет предел при $k \rightarrow \infty$, то этот предел является решением СЛАУ.

Доказано, что для сходимости итерационного процесса достаточно, чтобы какая-либо норма матрицы C была меньше единицы. Отсюда с учётом наиболее применимых норм матрицы вытекает следствие.

Следствие. Для того, чтобы метод простой итерации для системы уравнений $X = CX + F$ сходиллся, достаточно, чтобы выполнялось одно из условий:

$$\sum_{j=1}^n |c_{ij}| < 1; \quad i = 1, \dots, n, \quad \text{или} \quad \sum_{i=1}^n |c_{ij}| < 1; \quad j = 1, \dots, n, \quad \text{или} \quad \sum_{i,j=1}^n |c_{ij}|^2 < 1.$$

На практике обычно используется первое или второе условие. Метод простой итерации сходится только для хорошо обусловленных систем (для которых хорошо работают и другие, более эффективные методы).

Метод Гаусса - Зейделя

Метод Зейделя отличается от метода простой итерации тем, что при вычислении $(k+1)$ -го приближения для компоненты x_i учитываются определённые ранее $(k+1)$ -е приближения для компонент x_1, x_2, \dots, x_{i-1} .

Вычисления производим по формулам:

$$\begin{aligned}
x_1^{(k+1)} &= c_{11}x_1^{(k)} + c_{12}x_2^{(k)} + c_{13}x_3^{(k)} + \dots + c_{1n}x_n^{(k)} + f_1, \\
x_2^{(k+1)} &= c_{21}x_1^{(k+1)} + c_{22}x_2^{(k)} + c_{23}x_3^{(k)} + \dots + c_{2n}x_n^{(k)} + f_2, \\
x_3^{(k+1)} &= c_{31}x_1^{(k+1)} + c_{32}x_2^{(k+1)} + c_{33}x_3^{(k)} + \dots + c_{3n}x_n^{(k)} + f_3, \\
&\dots\dots\dots, \\
x_n^{(k+1)} &= c_{n1}x_1^{(k+1)} + c_{n2}x_2^{(k+1)} + c_{n3}x_3^{(k+1)} + \dots + c_{nn}x_n^{(k)} + f_n,
\end{aligned}$$

Таким образом,

$$x_i^{(k+1)} = \sum_{j=1}^{i-1} c_{ij}x_j^{(k+1)} + \sum_{j=1}^n c_{ij}x_j^{(k)} + f_i; \quad i=1, \dots, n; \quad k=0, 1, 2, \dots$$

Для сходимости метода Зейделя достаточно, чтобы выполнялось одно из

$$\text{условий: } \sum_{j=1}^n |c_{ij}| < 1; \quad i=1, \dots, n; \quad \sum_{i=1}^n |c_{ij}| < 1; \quad j=1, \dots, n.$$

Методы группы градиентного спуска

Если матрица СЛАУ положительно определенная, то градиентный метод с минимальными невязками имеет вид:

$$\begin{aligned}
x^{(k+1)} &= x^{(k)} + \gamma_k \cdot r^{(k)}, \\
\gamma_k &= \frac{(r^{(k)}, Ar^{(k)})}{\|Ar^{(k)}\|^2}.
\end{aligned}$$

Метод легко может быть расширен для СЛАУ с прямоугольными матрицами:

$$x^{(k+1)} = x^{(k)} + \gamma_k \cdot A^T \cdot r^{(k)},$$

в этой модификации шаг осуществляется вдоль антиградиента функционала невязки: $\text{grad } r^2 = -2 \cdot A^T \cdot r$, здесь A^T - транспонированная матрица.

Для выбора величины шага γ_k , найдем минимум функционала невязки вдоль луча $A^T \cdot r$. Следующие формулы описывают метод минимальных невязок для СЛАУ с неквадратной матрицей

$$\gamma_k = \frac{(r^{(k)}, AA^T r^{(k)})}{\|AA^T r^{(k)}\|^2},$$

здесь используется сферическая норма векторов $\|x\|^2 = \sum_{i=1}^n x_i^2$.

Метод наискорейшего спуска в применении к положительно обусловленным матрицам впервые описан в работах Канторовича

[7] и использует функцию ошибок вместо функционала невязки.

Функция ошибок имеет вид $(X^* - X^{(k)}, A(X^* - X^{(k)}))$,

X^* - точное решение.

Отличие от предыдущего метода только в выборе величины шага

$$\gamma_k = \frac{(r^{(k)}, r^{(k)})}{(Ar^{(k)}, r^{(k)})}.$$

2. Регуляризирующие алгоритмы решения СЛАУ

Рассмотрим СЛАУ $A \cdot x = b$,

где: A - матрица размером $m \cdot n$, b - вектор правой части, m - компонент, x - искомое решение, n - компонент.

Для построения устойчивого решения необходима информация о точности задания правой части и матрицы, а также априорная информация о гладкости решения.

Пусть h - относительная погрешность вычисления коэффициентов матрицы по норме $h = \frac{\|A - \tilde{A}\|}{\|A\|}$, где \tilde{A} - приближенно вычисленная

матрица и δ - относительная погрешность задания правой части СЛАУ

$$\delta = \frac{\|b - \tilde{b}\|}{\|b\|}.$$

Одна из идей, используемых для получения устойчивого решения, основана на оптимальном останове итерационных градиентных методов или

методов группы сопряженных направлений. Вторая идея основана на использовании сглаживающего функционала (метод регуляризации Тихонова).

Рассмотрим итерационные реализации обоих подходов.

Методы оптимального останова градиентных методов.

Как известно [5], любой стационарный итерационный алгоритм, оставляющий неподвижной точку решения может быть представлен в виде:

$$x^{(k+1)} = x^{(k)} + \gamma_k \cdot H_k \cdot r^{(k)},$$

где: k - номер итерации, $x^{(k+1)}$ - новое приближение для решения, $x^{(k)}$ - предыдущее приближение, γ_k - величина шага, H_k - матрица, r_k - вектор невязки: $r^{(k)} = b - A \cdot x^{(k)}$.

Ограничимся рассмотрением только градиентных методов с общей схемой: $x^{(k+1)} = x^{(k)} + \gamma_k \cdot A^T \cdot r^{(k)}$,

соответствующей шагу вдоль антиградиента функционала невязки:

$$\text{grad } r^2 = -2 \cdot A^T \cdot r, \text{ здесь } A^T - \text{транспонированная матрица.}$$

Для выбора величины шага γ_k , найдем минимум функционала невязки вдоль луча $A^T \cdot r$. Следующие формулы описывают метод минимальных невязок для СЛАУ с неквадратной матрицей

$$\gamma_k = \frac{(r^{(k)}, AA^T r^{(k)})}{\|AA^T r^{(k)}\|^2}, \text{ здесь используется сферическая норма векторов}$$

$$\|x\|^2 = \sum_{i=1}^n x_i^2.$$

Для выбора величины шага можно использовать и другие подходы, например величину шага можно выбирать с помощью датчика случайных чисел и соглашаться с выбором шага, если он приводит к уменьшению невязки. При этом наиболее сложным является ограничение величины шага (ограничение датчика случайных чисел).

Отметим, что градиентные методы обеспечивают существенно более медленную сходимость, чем методы группы сопряженных направлений, однако обеспечивают высокую степень гладкости решения.

Оптимальный останов метода минимальных невязок можно осуществить либо по величине нормы невязки, сравнивая её с погрешностями δ и $\|x^{(k)}\| \cdot h$, либо по величине изменения решения за одну итерацию. При этом следует учесть, что для переопределенных СЛАУ минимум функционала невязки не равен нулю и требуются дополнительные усилия для его количественной оценки.

Численные эксперименты показывают, что в некоторых случаях метод минимальных невязок замедляется и практически останавливается задолго до точки оптимального останова и поэтому последний не требуется.

Методы численной минимизации сглаживающего функционала.

В методе 0-регуляризации Тихонова вместо функционала невязки рассматривается функционал с “довеском” в виде нормы решения, умноженной на параметр регуляризации α [8]

$$\Phi_{\alpha} = \|b - A \cdot x_k\|^2 + \alpha \cdot \|x\|^2.$$

Использование этого функционала при малых α соответствует “умеренной” порче исходного СЛАУ и рассмотрению вместо него уравнения $A^T \cdot Ax + \alpha \cdot x = A^T b$.

Далее алгоритмы регуляризации можно условно разделить на алгоритмы численной минимизации сглаживающего функционала и на алгоритмы решения полученного СЛАУ. Для решения этой системы уравнений могут быть использованы как прямые так и итерационные методы.

Однако следует помнить, что матрица $A^T \cdot A$ существенно хуже обусловлена, чем исходная матрица A .

При использовании метода минимальных невязок для минимизации сглаживающего функционала достаточно заменить в градиентном алгоритме

$$A^T \cdot r^{(k)} \text{ на } q^{(k)} = A^T \cdot r^{(k)} - \alpha \cdot x^{(k)} \text{ и выражение } \gamma_k = \frac{(r^{(k)}, AA^T r^{(k)})}{\|AA^T r^{(k)}\|^2}$$

заменить на

$$\gamma_k = \frac{(r^{(k)}, Aq^{(k)}) - \alpha \cdot (x, q^{(k)})}{\|Aq^{(k)}\|^2 + \alpha \cdot \|q^{(k)}\|^2}.$$

Наименее понятной частью алгоритма является способ выбора параметра регуляризации α .

Известны несколько способов доопределения α , наиболее простой основан на

минимизации функции $\left\| \alpha \cdot \left(\frac{dx^{(\alpha)}}{d\alpha} \right) \right\|^2$ [9].

Если использовать схему итерационного удвоения α , то этот алгоритм сводится к нахождению α , при котором минимально изменяется решение

$$\|x_{\alpha_{i+1}} - x_{\alpha_i}\|^2.$$

Параметр α можно также подбирать эмпирически в пределах $\alpha \approx (10^{-3} \div 10^{-8}) \cdot \|b\|$.

При решении инженерных задач параметр регуляризации может быть выбран, как решение задачи компромисса между неточностью синтеза правой части и реализуемостью решения с осциллирующими источниками.

ТЕКСТ ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ МЕТОД ГАУССА РЕШЕНИЯ СЛАУ

Программа из [3], реализующая метод Гаусса с частичной выборкой ведущего элемента переписана нами с Фортрана на Паскаль, при этом исправлена небольшая ошибка в исходном коде, но использованы метки, что является «плохим тоном» в современном программировании.

```
procedure Solve ( N:integer; var b:array of real);
// N-число точек, ipvt, b - вспомогательные массивы
var   i,NM1,j,k,kb,km1,kp1,m,ib :integer;
      znorm,anorm,t,ek           :real;
begin;
  nm1:=N-1;
```

```

{1} for k:=1 to Nm1 do begin
    kp1:=k+1;
    m:=ipvt[k];
    t:=b[m];
    b[m]:=b[k];
    b[k]:=t;
    for i:=kp1 to N do b[i]:=b[i]+A[i,k]*t;
{1} end;{конец цикла по k}
{2} for kb:=1 to Nm1 do begin
    km1:=n-kb;
    k:=km1+1;
    b[k]:=b[k]/A[k,k];
    t:=-b[k];
    for i:=1 to km1 do b[i]:=b[i]+A[i,k]*t;
{2} end;{конец цикла по kp1}
    b[1]:=b[1]/a[1,1];
end; {процедуры solve}
//===== Gaus =====
procedure Gaus(N:integer;var x,f:array of real);
{стандартная процедура решения СЛАУ методом Гаусса}
// N-число точек,
// x - массив решений, f - массив правой части
var i,NM1,j,k,kb,km1,kp1,m,ib,iii :integer;
    cond,ynorm,znorm,anorm,t,ek :real;
    work :Array[0..10000] of real;
label m0,m1,m2,m3,m4,m5;
{1} begin
ipvt[n]:=1;
nm1:=n-1;
anorm:=0;
{2}for j:=1 to N do begin
    t:=0;
    for i:=1 to N do t:=t+abs(A[i,j]);
    if t > anorm then anorm:=t;
{2} end;{цикл по j}
{3}for k:=1 to Nm1 do begin
    kp1:=k+1;
    m:=k;
{4}for i:=kp1 to N do begin
    if abs(A[i,k]) > abs(A[m,k]) then m:=i;
{4}end;
ipvt[k]:=m;
if m <> k then ipvt[N]:=-ipvt[N];
t:=A[m,k];
A[m,k]:=A[k,k];

```



```

    A[k,k]:=t;
    if t = 0 then goto m0;
    for i:=kp1 to N do A[i,k]:=-A[i,k]/t;
{5}for j:=kp1 to N do begin
    t:=A[m,j];
    A[m,j]:=A[k,j];
    A[k,j]:=t;
    if t= 0 then goto m1;
    for i:=kp1 to N do A[i,j]:=A[i,j]+A[i,k]*t;
m1:{5}end;{j}
m0:{3}end;{k}
{6}for k:=1 to N do begin
    t:=0;
    if k = 1 then goto m2;
    km1:=k-1;
    for i:=1 to km1 do t:=t+A[i,k]*work[i];
    m2:ek:=1;
    if t < 0 then ek:=-1;
    if A[k,k]=0 then goto m3;
    work[k]:=-(ek+t)/a[k,k];
{6}end;{цикла по k}
{7}for kb:=1 to Nm1 do begin
    k:=N-kb;
    t:=0;
    kp1:=k+1;
    for i:=kp1 to N do t:=t+a[i,k]*work[k];
    work[k]:=t;
    m:=ipvt[k];
    if m = k then goto m4;
    t:=work[m];
    work[m]:=work[k];
    work[k]:=t;
m4:{7}end;{kb1}
ynorm:=0;
for i:=1 to N do ynorm:=ynorm+abs(work[i]);
solve(n,{ipvt,}work);
znorm:=0;
for i:=1 to N do znorm:=znorm+abs(work[i]);
cond:=anorm*znorm/ynorm;
if cond < 1 then cond:=1;
goto m5;
m3:cond:=1e32;
m5:for k:=1 to N do work[k]:=f[k];
    solve(n,{ipvt,}work);
for k:=1 to N do x[k]:=work[k];

```

```
end;{процедуры gauss}
```

Ниже приведен заново написанный нами фрагмент программы решения СЛАУ методом Гаусса без перестановки строк:

```
//Прямой ход - исключение переменных без выбора главного элемента
//(перестановки строк)
//N-число уравнений, a[ , ] - матрица, b[] - правая часть, X[]- решение
//в процессе решения "порти" матрицу и правую часть (старые значения не
//сохраняем)
for k:=1 to N-1 do
//кроме последней строки, k-номер строки, которую используем
// для исключения переменных в нижних строках
    for i:=k+1 to N do
// i-номер строки, начиная с которой исключаем переменную
begin
        coeff:=-a[i,k]/a[k,k];
//множитель для сокращения элементов в столбце под a[k,k]
        b[i]:=b[i]+coeff*b[k];
//складываем правую часть с учетом множителя
        for m:=k+1 to N do //m-номер столбца в i-ой строке
            a[i,m]:=a[i,m]+coeff*a[k,m]; //складываем строки с учетом
множителя
//вычисляем элементы только над диагональю
//(остальные дальше не используются)
end;
        x[n]:=b[n]/a[n,n]; //сразу находим последний элемент
//Обратный ход - нахождение остальных корней
        for i:=n-1 downto 1 do
begin
            coeff:=b[i];//вспомогательная переменная
            for j:=i+1 to N do coeff:=coeff-x[j]*a[i,j];
            x[i]:=coeff/a[i,i]
end;
end;
```

Лабораторная работа №3

РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ И СИСТЕМ (СНАУ)

Ключевые понятия: нелинейные и трансцендентные уравнения, методы дихотомии, золотого сечения, Ньютона, секущих, Ньютона-Рафсона.

Перед выполнением лабораторной работы рекомендуем:

1. Изучить презентацию лектора курса: «Численные методы решения нелинейных уравнений и СНАУ», материалы доступны в облаке на Mail.ru.
2. Прочитать соответствующие разделы книги:
Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ.- М.: Мир, 1998.- 575 с.

Цель работы:

1. Изучить и научиться использовать на практике наиболее эффективные численные алгоритмы решения нелинейных, трансцендентных уравнений и СНАУ.
2. Изучить методы итерационного уточнения корней: метод дихотомии (половинного деления), метод золотого сечения, метод простой итерации, метод Ньютона (касательных), метод секущих, метод обратной квадратичной интерполяции.
3. Изучить методы решения СНАУ: метод Зейделя, Ньютона-Рафсона, итерационные методы, основанные на минимизации функционала невязки.
4. Написать программу, реализующую два метода численного решения СНАУ:
1) Дихотомии, или золотое сечение + Ньютона, или секущих; 2) Решение СНАУ методом Ньютона-Рафсона. Пример взять из таблицы к лаб. работе №3, в соответствии со своим списочным номером.
5. Дополнительные задания (бонусы к оценке):
Решение нелинейной системы из 6 уравнений с 6 неизвестными.

Содержание отчета:

1. Постановка задачи.
2. Краткое описание методов решения нелинейного уравнения и СНАУ.
3. Листинг программы с подробными комментариями.
4. Результаты решения и проверки решения.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В 20-х годах XIX века Э. Галуа доказал, что решение алгебраического уравнения n -ой степени

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 = 0 \quad (3.1)$$

при $n \geq 5$ нельзя выразить через коэффициенты с помощью арифметических действий и операций извлечения корня.

Трансцендентные уравнения, включающие алгебраические, тригонометрические, экспоненциальные функции, как правило, имеют неопределённое число корней.

Необходимость решения трансцендентных уравнений возникает, например, при расчёте устойчивости систем и т.п.

Достаточно распространённой задачей является нахождение некоторых или всех решений системы из n нелинейных трансцендентных уравнений с n неизвестными.

Рассмотрим вначале методы решения нелинейных уравнений с одним неизвестным. Пусть задана непрерывная функция $f(x)$ и требуется найти корни уравнения $f(x)=0$ на всей числовой оси или на некотором интервале $a < x < b$. Методы решения нелинейных уравнений с одним неизвестным можно разделить на прямые (формула Виета для квадратного уравнения и Кардано для кубического и другие), и итерационные – для решения любых уравнений.

Итерационное решение уравнения проводится в два этапа:

1 этап. Отделение корней уравнения.

2 этап. Уточнение интересующих корней с заданной точностью.

Отделение корней нелинейного уравнения.

Отделение корней – это определение их наличия, количества и нахождение для каждого из них достаточно малого отрезка $[a,b]$, которому он принадлежит. В инженерных расчетах, как правило, необходимо определять только вещественные корни. Задача отделения вещественных корней решается аналитическими и графическими методами.

Аналитические методы основаны на функциональном анализе.

Для алгебраического многочлена n -ой степени (полинома) с действительными коэффициентами вида

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0 = 0, \quad (a_n > 0) \quad (3.2)$$

верхняя граница положительных действительных корней R_B^+ определяется по формуле Лагранжа:

$$R_B^+ = 1 + k \sqrt[k]{\frac{B}{a_n}}, \quad B = \max_{a_i < 0} |a_i|, \quad (3.3)$$

где: $k \geq 1$ – номер первого из отрицательных коэффициентов полинома;

B – максимальный по модулю отрицательный коэффициент.

Нижнюю границу положительных действительных корней R_H^+ можно определить из вспомогательного уравнения

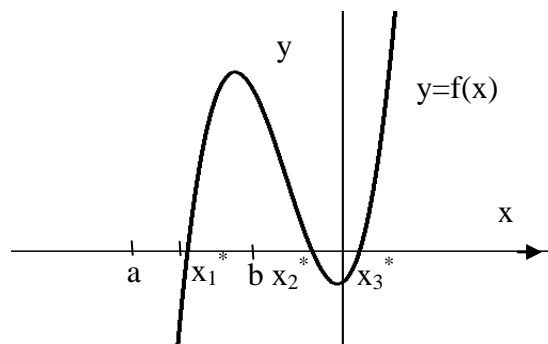
$$P'_n(x) = x^n \cdot P_n\left(\frac{1}{x}\right) \quad (3.4)$$

Для трансцендентных уравнений не существует общего метода оценки интервала, в котором находятся корни. Для этих уравнений оцениваются значения функции в особых точках: разрыва, экстремума, перегиба и других.

На практике получил большее распространение графический метод приближённой оценки вещественных корней. Для этих целей строится график функции по вычисленным её значениям.

Графически корни можно отделить 2-мя способами:

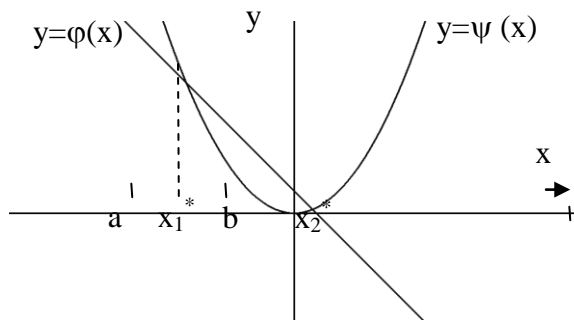
1. Построить график функции $y = f(x)$ и определить координаты пересечений с осью абсцисс – это приближенные значения корней уравнения.



На графике 3 корня.
Первый корень
 $x^* \in [a, b]$

Рис. 3.1 Отделение корней на графике $f(x)$.

2. Преобразовать $f(x)=0$ к виду $\varphi(x) = \psi(x)$, где $\varphi(x)$ и $\psi(x)$ – элементарные функции, и определить абсциссу пересечений графиков этих функций.



На графике 2 корня.
Первый корень
 $x_1^* \in [a, b]$

Рис. 3.2 Отделение корней по графикам функций $\varphi(x)$ и $\psi(x)$.

Графический метод решения нелинейных уравнений широко применяется в технических расчётах, где не требуется высокая точность.

Для отделения вещественных корней можно использовать компьютерные вычисления. Алгоритм отделения корней основан на факте изменения знака функции в окрестности корня. Действительно, если корень вещественный, то график функции пересекает ось абсцисс, а знак функции изменяется на противоположный.

Недостаток метода – использование большого количества машинного времени.

Алгоритмы уточнения корней уравнения.

Уточнение корня – это вычисление интересующего корня с заданной точностью ε . Приближённые значения корней уравнения, полученные на предыдущем этапе, уточняются различными итерационными методами. Рассмотрим некоторые из них.

2. Метод дихотомии (половинного деления, бисекций)

Постановка задачи:

Дано нелинейное уравнение $f(x) = 0$.

Корень отделен, т.е. известно, что $x^* \in [a, b]$.

Требуется вычислить корень с заданной точностью ε .

Метод реализует стратегию постепенного уменьшения отрезка существования корня, используя факт изменения знака функции в окрестности корня.

Алгоритм метода.

1. Вычислить координату середины отрезка $[a,b]$ $x = (a+b)/2$ и значение $f(x)$ в этой точке.
2. Уменьшить отрезок, отбросив ту его половину, на которой корня нет. Если знак функции в начале отрезка и в его середине одинаков, то корень находится на второй половине, первую половину можно отбросить, переместив начало отрезка в его середину:
если $f(a) \cdot f(x) > 0 \Rightarrow x^* \in [x, b] \Rightarrow a=x$, иначе $x^* \in [a, x] \Rightarrow b=x$.
3. Проверить условие завершения вычислений: длина отрезка не превышает заданную точность и значение функции близко к 0 с заданной точностью. Если условие достигнуто, расчет завершен, иначе повторить алгоритм сначала. Количество итераций n , требуемых для достижения требуемой точности ε можно оценить заранее из соотношения

$$\varepsilon = \frac{b-a}{2^n} \Rightarrow n = \frac{\ln \frac{b-a}{\varepsilon}}{\ln 2} \quad (3.6)$$

Метод дихотомии – простой и надежный метод поиска простого корня любой функции, устойчивый к погрешности округления. Даже если на отрезке есть несколько корней (нечетное количество), то будет найден один из них.

Недостатки метода: скорость сходимости низкая, не обобщается на систему уравнений.

Метод дихотомии нельзя использовать для уточнения не простого корня – корень совпадает с точкой экстремума функции, т.к. в этом случае функция не изменяет свой знак в окрестности корня.

4. Метод простых итераций

Уравнение (3.1) преобразуем к эквивалентному виду

$$x = \varphi(x). \quad (3.7)$$

Выберем начальное приближение $x_0 \in [a;b]$.

Вычислим новые приближения:

$$x_1 = \varphi(x_0)$$

$$x_2 = \varphi(x_1)$$

$$x_i = \varphi(x_{i-1}), i=1,2,\dots \text{ где } i - \text{ номер итерации.} \quad (3.8)$$

Последовательное вычисление значений x_i по формуле (3.8) называется итерационным процессом метода простых итераций.

Рассмотрим геометрическую иллюстрацию метода простых итераций. Уравнение (3.7) представим на графике в виде двух функций: $y_1 = x$ и $y_2 = \varphi(x)$. Возможные случаи взаимного расположения графиков функций, и соответственно, видов итерационного процесса показаны на рис. 3.3

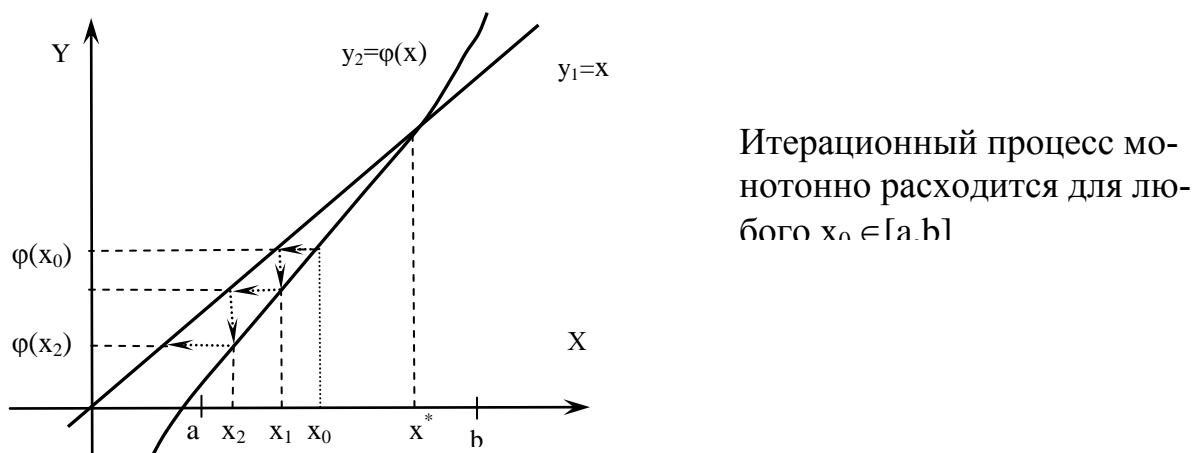
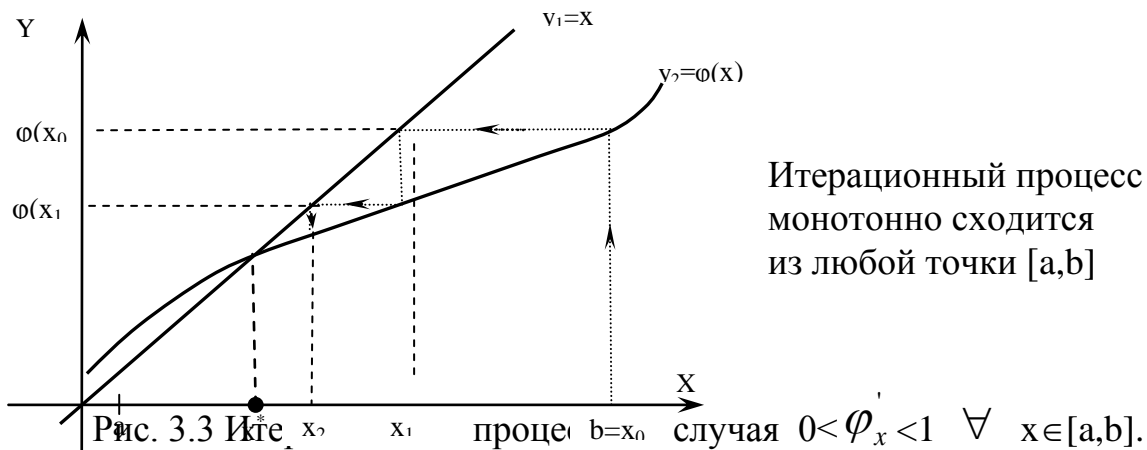


Рис. 3.4 Итерационный процесс для случая $\varphi'_x > 1 \quad \forall x \in [a,b]$.

Из анализа графиков следует, что скорость сходимости растет при уменьшении значения $|\varphi'_x|$.

Метод достаточно прост, обобщается на системы уравнений, устойчив к погрешности округления (она не накапливается).

При разработке алгоритма решения нелинейного уравнения методом простых итераций следует предусмотреть защиту итерационного процесса от заклинивания: использовать в качестве дополнительного условия завершения итерационного процесса превышение заданного максимального числа итераций.

5. Метод Ньютона (касательных).

Метод основан на стратегии постепенного уточнения корня. Формулу уточнения можно получить из геометрической иллюстрации идеи метода.

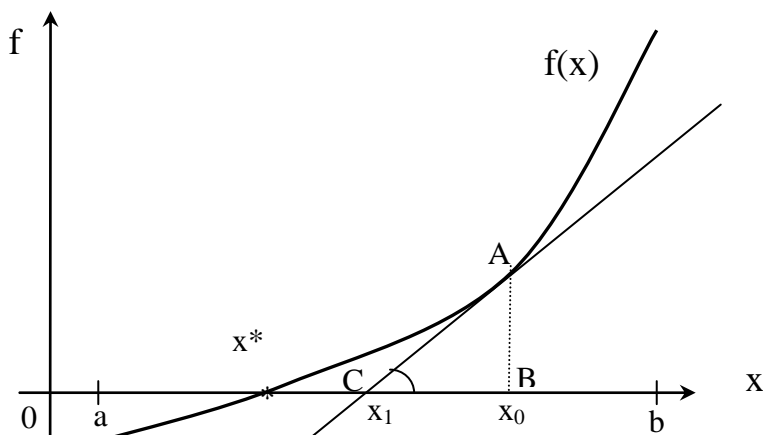


Рис. 3.15. Геометрическая иллюстрация метода Ньютона.

На отрезке существования корня выбирается начальное приближение x_0 . К кривой $f(x)$ в точке A с координатами $(x_0, f(x_0))$ проводится касательная. Абсцисса x_1 точки пересечения этой касательной с осью OX является новым приближением корня.

Из рисунка следует, что $x_1 = x_0 - CB$

Из $\triangle ABC$: $CD = \frac{AB}{\operatorname{tg} \angle ACB}$. Но $\operatorname{tg} \angle ACB = f'(x_0)$, $AB = f(x_0)$.

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

Аналогично, для i -го приближения можно записать формулу итерационного процесса метода Ньютона:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}, i = 1, 2, \dots, \text{ где } x_0 \in [a; b]. \quad (3.13)$$

$$\text{Условие окончания расчета: } |\delta| \leq \varepsilon, \quad (3.14)$$

где $\delta = x_{i-1} - x_i = \frac{f(x_{i-1})}{f'(x_{i-1})}$ –корректирующее приращение или поправка.

Условие сходимости итерационного процесса:

$$|f(x) \cdot f''(x)| < (f'(x))^2 \forall x \in [a, b] \quad (3.15)$$

Если на отрезке существования корня знаки $f'(x)$ и $f''(x)$ не изменяются, то начальное приближение, обеспечивающее сходимость, нужно выбрать из условия

$$f(x_0) \cdot f''(x_0) > 0, x_0 \in [a; b]. \quad (3.16)$$

т.е. в точке начального приближения знаки функций и ее второй производной должны совпадать.

Метод Ньютона в отличие от ранее рассмотренных методов использует значения производной, что значительно ускоряет итерационный процесс. При этом, чем больше значение модуля производной в окрестности корня (чем круче график функции), тем быстрее сходимость.

Достоинства метода: высокая скорость сходимости; обобщается на системы уравнений.

Недостатки: требуется вычисление производных; сильная зависимость сходимости от вида функции и выбора начального приближения.

Пример 3.3.

Методом Ньютона уточнить корни уравнения $x^3 = 1 - 2x$ с точностью $\varepsilon = 0,001$. Корень отделён ранее (пример 3.1), $x^* \in [0, 1]$.

Сначала нужно выбрать начальное приближение.

$$f(x) = x^3 + 2x - 1,$$

$$f'(x) = 3x^2 + 2,$$

$$f''(x) = 6x.$$

Производные имеют постоянный знак на отрезке $(0, 1]$, поэтому для выбора начального приближения достаточно использовать условие (3.16).

Знак второй производной на отрезке положительный, следовательно

$$x_0 = b = 1, \text{ т.к. } f(b) = f(1) = 1^3 + 2 \cdot 1 - 1 = 2 > 0$$

Вычислим несколько приближений:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{1^3 + 2 \cdot 1 - 1}{3 \cdot 1^2 + 2} = 1 - \frac{2}{5} = 0,6$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0,6 - \frac{0,6^3 + 2 \cdot 0,6 - 1}{3 \cdot 0,6^2 + 2} = 0,6 - 0,135065 = 0,464935$$

$$x_3 = 0,464935 - 0,011468 = 0,453467$$

$$x_3 = 0,453463 - 0,0000695 = 0,453398.$$

Решение получено за 4 итерации, так как поправка стала меньше заданной точности: $0,0000695 < \varepsilon$.

Решение систем нелинейных уравнений (СНУ).

При моделировании задача нахождения решения системы алгебраических или трансцендентных уравнений является распространенной вычислительной задачей. Например, к решению таких систем сводятся расчеты фазового и химического равновесия многокомпонентных смесей, расчеты статических режимов многих технологических процессов и др.

Запишем систему n нелинейных уравнений с n неизвестными (СНУ) в общем виде:

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \dots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \quad (3.17)$$

Эту систему можно записать в компактной, операторной форме:

$$F(X) = 0 \quad (3.18)$$

где

$$F = \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{bmatrix} \quad \text{вектор-функция} \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad \text{вектор неизвестных}$$

Решением системы называется набор значений x_i^* , $i = \overline{1, n}$ (вектор X^*), при которых все функции f_i равны 0 (система (3.17) обращается в тождество.)

СНУ могут иметь единственное решение, множество решений или вообще не иметь его. Поэтому численное решение СНУ проводят в два этапа:

1 этап – отделение решений.

2 этап – уточнение всех или только нужных решений.

Отделить решения – значит установить количество решений, определить приближенные значения каждого из них или указать область, в которой решение существует и является единственным.

Задача отделения решений достаточно просто решается только для системы двух уравнений с двумя неизвестными.

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases}$$

Для этого необходимо в координатах (x_1, x_2) построить кривые

$$f_1(x_1, x_2) = 0, \quad f_2(x_1, x_2) = 0.$$

Точки пересечения этих кривых являются решениями системы. Так как координаты точек пересечения определяются приближенно, целесообразно говорить об области существования решения. Эта область задается интервалами по каждой координате, внутри которых находятся искомые значения неизвестных.

Для систем с большим числом неизвестных ($n \geq 3$) удовлетворительных общих методов определения области существования решения нет. Поэтому при решении СНУ эта область обычно определяется при анализе решаемой задачи, например, исходя из физического смысла неизвестных.

Отделение решений позволяет:

1. Выявить число решений и область существования каждого из них.
2. Проанализировать возможность применения выбранного метода решения СНУ в каждой области.
3. Выбрать начальное приближение решения $X^{(0)}$ из области его существования, так что $X^{(0)} \in D$.

При отсутствии информации об области существования решения СНУ выбор начального приближения $X^{(0)}$ проводится методом проб и ошибок.

6. Метод Ньютона–Рафсона.

Идея метода заключается в линеаризации уравнений системы (3.18), что позволяет свести исходную задачу решения СЛУ к многократному решению системы линейных уравнений.

Рассмотрим, как были получены расчетные зависимости метода.

Пусть известно приближение $x_i^{(k)}$ решения системы нелинейных уравнений x_i^* . Введем в рассмотрение поправку Δx_i как разницу между решением и его приближением:

$$\Delta x_i = x_i^* - x_i^{(k)} \Rightarrow x_i^* = x_i^{(k)} + \Delta x_i, \quad i = \overline{1, n}$$

Подставим полученное выражение для x_i^* в исходную систему.

$$\begin{cases} f_1(x_1^{(k)} + \Delta x_1, x_2^{(k)} + \Delta x_2, \dots, x_n^{(k)} + \Delta x_n) = 0 \\ f_2(x_1^{(k)} + \Delta x_1, x_2^{(k)} + \Delta x_2, \dots, x_n^{(k)} + \Delta x_n) = 0 \\ \dots \\ f_n(x_1^{(k)} + \Delta x_1, x_2^{(k)} + \Delta x_2, \dots, x_n^{(k)} + \Delta x_n) = 0 \end{cases}$$

Неизвестными в этой системе нелинейных уравнений являются поправки Δx_i . Для определения Δx_i нужно решить эту систему. Но решить эту задачу так же сложно, как и исходную. Однако эту систему можно линеаризовать, и, решив ее, получить приближенные значения поправок Δx_i для данного приближения, т.е. $\Delta x_i^{(k)}$. Эти поправки не позволяют сразу получить точное решение x_i^* , но дают возможность приблизиться к решению, – получить новое приближение решения

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)}, \quad i = \overline{1, n} \quad (3.19)$$

Для линеаризации системы следует разложить функцию f_i в ряды Тейлора в окрестности $x_i^{(k)}$, ограничиваясь первыми дифференциалами.

Полученная система имеет вид:

$$\sum_{i=1}^n \frac{\partial f_i(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})}{\partial x_i} \Delta x_i^{(k)} = -f_j(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}), \quad j = \overline{1, n} \quad (3.20)$$

Все коэффициенты этого уравнения можно вычислить, используя последнее приближение решения $x_i^{(k)}$. Для решения системы линейных уравнений при $n=2,3$ можно использовать формулы Крамера, при большей размерности системы n – метод исключения Гаусса.

Значения поправок используются для оценки достигнутой точности решения. Если максимальная по абсолютной величине поправка меньше заданной точности ε , расчет завершается. Таким образом, условие окончания расчета:

$$\delta = \max_{i=1,n} |\Delta x_i^{(k)}| \leq \varepsilon$$

Можно использовать и среднее значение модулей поправок:

$$\delta = \frac{1}{n} \sum_{i=1}^n |\Delta x_i| < \varepsilon$$

В матричной форме систему (3.20) можно записать как:

$$W(X^{(k)}) \cdot \Delta X^{(k)} = -F(X^{(k)}) \quad (3.21)$$

где: $W(X) = \left(\frac{\partial f_j}{\partial x_i} \right)_{n,n} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$, - матрица Якоби (производных),

$$\Delta X^{(k)} = \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \\ \dots \\ \Delta x_n^{(k)} \end{bmatrix} \text{ - вектор поправок}$$

$F(X)$ - вектор-функция

$W(X^{(k)})$ – матрица Якоби, вычисленная для очередного приближения.

$F(X^{(k)})$ – вектор-функция, вычисленная для очередного приближения.

Выразим вектор поправок $\Delta X^{(k)}$ из (3.21):

$$\Delta X^{(k)} = -W^{-1}(X^{(k)}) \cdot F(X^{(k)})$$

где W^{-1} – матрица, обратная матрице Якоби.

Окончательно формула последовательных приближений метода Ньютона решения СЛУ в матричной форме имеет вид:

$$X^{(k+1)} = X^{(k)} - W^{-1}(X^{(k)}) \cdot F(X^{(k)}) \quad (3.22)$$

Достаточные условия сходимости для общего случая имеют очень сложный вид, и на практике проверяются редко. Нужно отметить, что метод сходится очень быстро (за 3 – 5 итераций), если $\det|W| \neq 0$ и начальное приближение $X^{(0)}$ выбрано близким к решению (отличаются не более чем на 10%).

Алгоритм решения СЛУ методом Ньютона состоит в следующем:

1. Задается размерность системы n , требуемая точность ε , начальное приближенное решение $X = (x_i)_n$.
2. Вычисляются элементы матрицы Якоби $W = (\partial f_i / \partial x_j)_{n,n}$.
3. Вычисляется обратная матрица W^{-1} .
4. Вычисляется вектор функция $F = (f_i)_n$, $f_i = f_i(x_1, x_2, \dots, x_n)$, $i = \overline{1, n}$.
5. Вычисляются вектор поправок $\Delta X = W^{-1} \cdot F$.
6. Уточняется решение $X_{i+1} = X_i + \Delta X$.
7. Оценивается достигнутая точность $\delta = \max_{i=1,n} |\Delta x_i^{(k)}|$ или $\delta = \frac{1}{n} \sum_{i=1}^n |\Delta x_i|$.
8. Проверяется условие завершения итерационного процесса $\delta \leq \varepsilon$. Если оно не соблюдается, алгоритм выполняется снова с пункта 2.

Для уменьшения количества арифметических действий Рафсон предложил не вычислять обратную матрицу W^{-1} , а вычислять поправки как решение СЛАУ (3.21) $W \cdot \Delta X = -F$.

Достоинством методов Ньютона является быстрая сходимость, недостатками - сложность расчетов (вычисление производных, многократное решение системы линейных уравнений), сильная зависимость от начального приближения.

Лабораторная работа №4

ИНТЕРПОЛЯЦИЯ И АППРОКСИМАЦИЯ

Ключевые понятия: аппроксимация и интерполяция, экстраполяция, полином Лагранжа, кубический сплайн, кривые Безье, численное дифференцирование, метод наименьших квадратов.

Перед выполнением лабораторной работы рекомендуем:

3. Изучить презентацию лектора курса: «Интерполяция и аппроксимация», материалы доступны в облаке на Mail.ru.
4. Прочитать соответствующие разделы книги:
Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ.- М.: Мир, 1998.- 575 с.,
Дж. Форсайт, М.Малькольм, К. Моулер. Машинные методы математических «вычислений» – М.: Мир, 1980.

Цель работы:

1. Изучить и научиться отличать термины интерполяция, аппроксимация, экстраполяция, сплайны, кривые Безье, полином Лагранжа, метод наименьших квадратов.
2. Научиться использовать на практике наиболее эффективные алгоритмы глобальной и локальной интерполяции.
3. Изучить устойчивые приемы численного дифференцирования.
4. Изучить метод численного интегрирования функций, заданных на неравномерной сетке, с помощью сплайнов.
5. Написать программу, реализующую два метода интерполяции табличных значений:
 - 1) Полином Лагранжа для глобальной интерполяции;
 - 2) Кубический сплайн.Пример подготовить самостоятельно, обеспечив возможность изменения числовых данных на неравномерной сетке.
6. Указание: с помощью численных экспериментов выявить недостатки интерполяции с помощью полинома Лагранжа. При построении кубического сплайна коэффициенты находить численно путем решения четырех диагональной СЛАУ методом прогонки.

Дополнительные задания (бонусы):

- 1) Построить сплайн Эрмита;
- 2) Построить кривые Безье;
- 3) Реализовать устойчивый алгоритм численного дифференцирования.

Содержание отчета:

1. Постановка задачи. Формульное описание алгоритма.
2. Описание компьютерной программы. Объяснение значений всех параметров.
3. Нахождение промежуточного значения с помощью полинома Лагранжа и кубического сплайна.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. Интерполяция и аппроксимация

Аппроксимация – это замена сложной функции (или табличных данных) простой функцией, усредненно проходящей через данные точки. В общем случае – это метод, состоящий в замене одних объектов другими, в том или ином смысле близкими к исходным, но более простыми.

Интерполяция (вид аппроксимации) – это способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных величин. При этом интерполянт точно проходит через имеющиеся точки данных. Рассмотрим систему несовпадающих точек $x(i)$ из некоторой области. Пусть значения функции известны только в этих точках:

Задача интерполяции состоит в поиске такой функции из заданного класса функций, которая точно проходит через узлы интерполяции.

Точки $x(i)$ - называют узлами интерполяции, а их совокупность — интерполяционной сеткой. Пары $x(i)$, $y(i)$ - называют точками данных или базовыми точками, разность между «соседними» значениями - шагом интерполяционной сетки. Он может быть как переменным так и постоянным.

Функцию $F(x)$ — называют интерполирующей функцией или интерполянтом. Данные сами по себе не могут определить интерполянт. Для фиксированного набора данных существует бесконечно много интерполянтов.

Интерполяция может быть полезной только в том случае, если данные не содержат ошибок. Экспериментальные данные, содержащие ошибки, надо аппроксимировать как-то по-другому. Например, с помощью метода наименьших квадратов.

Интерполяция – это чтение промежуточных значений между строками

таблицы. Например, пусть есть табличные данные и нужно найти

6000	15.5
6378	?
8000	19.2

промежуточное значение, соответствующее значению аргумента 6378. Линейная интерполяция дает значение:

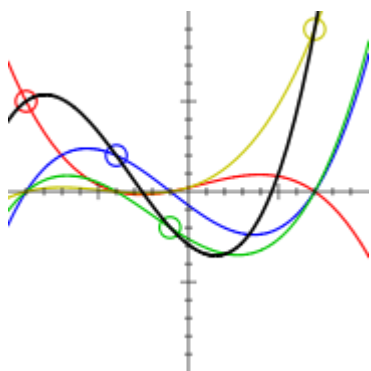
$$15.5 + \frac{6378 - 6000}{8000 - 6000} \cdot (19.2 - 15.5) \approx 16.2$$

В общем случае глобальная интерполяция функции на всем отрезке сводится к выбору базисных функций b_k , для которых получаем СЛАУ для определения коэффициентов a_k .

Обычно используют полиномиальную или кусочно полиномиальную интерполяцию (линейную, квадратичную, кубическую, или полиномами старших степеней)

2. Интерполяционный многочлен Лагранжа

Интерполяционный многочлен Лагранжа - это многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для $n + 1$ пар чисел $(x_0, y_0), (x_1, y_1) \dots, (x_n, y_n)$, где все x_i различны, существует



единственный многочлен $L(x)$ степени не более n , для которого $L(x_i) = y_i$.

В простейшем случае ($n = 1$) — это линейный многочлен, график которого — прямая, проходящая через две заданные точки.

Этот пример показывает интерполяционный многочлен Лагранжа для четырех точек $(-9, 5)$, $(-4, 2)$, $(-1, -2)$ и $(7, 9)$, а также полиномы $y_j l_j(x)$, каждый из которых проходит через одну из выделенных точек, и принимает нулевое значение в остальных x_i . Лагранж предложил способ вычисления таких многочленов:

$$L(x) = \sum_{j=0}^n y_j l_j(x) \quad (4.1)$$

где базисные полиномы определяются по формуле:

$$l_j(x) = \prod_{i=0, i \neq j}^n \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \dots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \dots \frac{x - x_n}{x_j - x_n} \quad (4.2)$$

$l_j(x)$ обладают следующими свойствами:

- являются многочленами степени n
- $l_j(x_j) = 1$
- $l_j(x_i) = 0$ при $i \neq j$

Отсюда следует, что $L(x)$, как линейная комбинация $l_j(x)$, может иметь степень не больше n , и $L(x_j) = y_j$,

Полиномы Лагранжа используются для интерполяции, а также для численного интегрирования. Пусть для функции $f(x)$ известны значения $y_j = f(x_j)$ в некоторых точках. Тогда мы можем интерполировать эту функцию как

$$f(x) \approx \sum_{j=0}^n f(x_j) l_j(x)$$

В частности,

$$\int_a^b f(x) dx \approx \sum_{j=0}^n f(x_j) \int_a^b l_j(x) dx$$

Значения интегралов от l_j не зависят от $f(x)$, и их можно вычислить заранее, зная последовательность x_i . На этой идее построен метод интегрирования Гаусса на неравномерной, специально построенной сетке.

В случае равномерного распределения узлов интерполяции x_i выражаются через расстояние между узлами интерполяции h и начальную точку x_0 :

$$x_j \equiv x_0 + jh,$$

и, следовательно,

$$x_i - x_j \equiv (i - j)h.$$

Подставив эти выражения в формулу базисного полинома, и вынеся h за знаки перемножения в числителе и знаменателе, получим

$$l_i(x) = \prod_{j=0, j \neq i}^n \frac{(x - x_j)}{(x_i - x_j)} = \frac{\prod_{j=0, j \neq i}^n (x - x_0 - jh)}{h^{n-1} \prod_{j=0, j \neq i}^n (i - j)} \quad (4.3)$$

Теперь можно ввести замену переменной

$$y = \frac{x - x_0}{h}$$

и получить полином от y , который строится с использованием только целочисленной арифметики. Недостатком данного подхода является факториальная сложность числителя и знаменателя, что требует использования алгоритмов с многобайтным представлением чисел. Интерполяционный полином Лагранжа на практике следует применять при небольшом числе точек интерполяции ($N \leq 20$). При этом метод Лагранжа будет работать достаточно быстро, т.к. все коэффициенты могут быть вычислены заранее, до многократного вызова метода интерполяции.

На Рис.4.1 представлен результат неудачного применения полинома Лагранжа для функции с резким выбросом значений (синяя ломаная проходит через точки интерполяции, красная линия – полином Лагранжа).

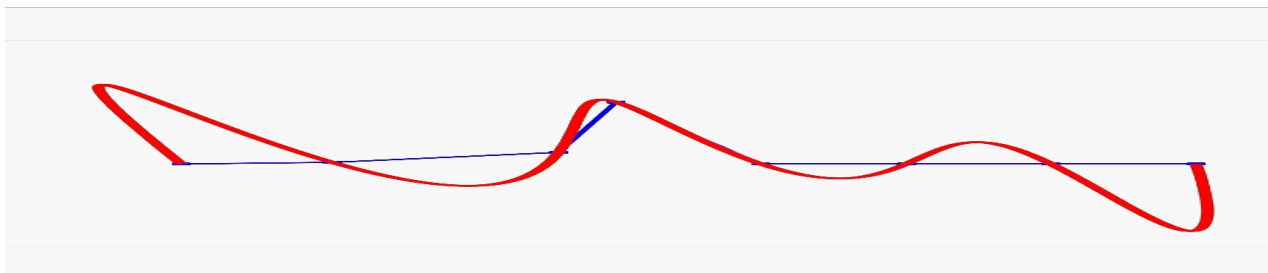


Рис.4.1

Кроме того, некоторые, достаточно гладкие функции, плохо интерполируются полиномами Лагранжа (и другими степенными полиномами), например функция Рунге, Рис.4.2

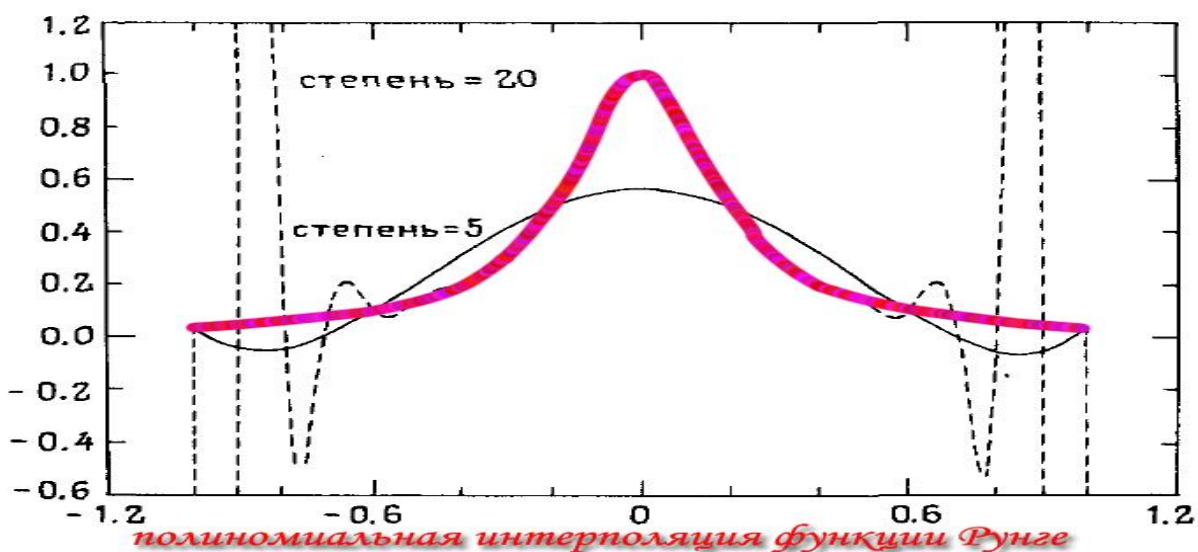


Рис.4.2

3. Интерполяция сплайнами

Описание метода:

Пусть отрезок $[a, b]$ разбит на n равных частей $[x_i, x_{i+1}]$, где $x_i = a + ih$, $i = 0, \dots, n$, $x_n = b$, $h = (b - a)/n$.

Сплайном называется функция, которая вместе с несколькими производными непрерывна на всем заданном отрезке $[a, b]$, а на каждом частичном отрезке $[x_i, x_{i+1}]$ в отдельности является некоторым алгебраическим многочленом.

Максимальная по всем частичным отрезкам степень многочленов называется *степенью сплайна*, а разность между степенью сплайна и порядком наивысшей непрерывной на $[a, b]$ производной - *дефектом сплайна*.

Определение. Функция $S_{n, \nu}(x)$ называется сплайном степени n дефекта ν (ν -целое число, $0 \leq \nu \leq n + 1$) с узлами на сетке Δ ($\Delta: a = x_0 < x_1 < \dots < x_n = b$), если:
а) на каждом отрезке $[x_i, x_{i+1}]$ функция $S_{n, \nu}(x)$ является многочленом степени n

$$S_{n, \nu}(x) = \sum_{k=0}^n a_k^i (x - x_i)^k \quad \text{для } x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1;$$

$$\text{б) } S_{n, \nu}(x) \in C^{n-\nu}[a, b]$$

(для целого $k > 0$ через $C^k = C^k[a, b]$ обозначается множество k раз непрерывно дифференцируемых на $[a, b]$ функций).

Интерполяция сплайном

На практике широкое распространение получили сплайны третьей степени, имеющие на $[a, b]$ непрерывную, по крайней мере, первую производную. Эти сплайны называются кубическими и обозначаются $S_3(x)$ (без указания дефекта).

Пусть на отрезке $[a, b]$ в узлах сетки Δ заданы значения некоторой функции $f_i = f(x_i)$, $i = 0, \dots, n$.

Интерполяционным кубическим сплайном $S_3(x)$ называется сплайн

$$S_3(x) = a_{i0} + a_{i1}(x - x_i) + a_{i2}(x - x_i)^2 + a_{i3}(x - x_i)^3, \quad x \in [x_i, x_{i+1}], \quad (4.4)$$

удовлетворяющий условиям

$$S_3(x_i) = f(x_i), \quad i = 0, \dots, n. \quad (4.5)$$

Сплайн (4.4) на каждом из отрезков $[x_i, x_{i+1}]$, $i=0, \dots, n-1$ определяется четырьмя коэффициентами, и поэтому для его построения на всем промежутке $[a, b]$ необходимо определить $4n$ коэффициентов. Для их однозначного определения необходимо задать $4n$ уравнений.

Условие (4.5) дает $2n$ уравнений, при этом функция $S_3(x_i)$, удовлетворяющая этим условиям, будет непрерывна во всех внутренних узлах.

Условие непрерывности производных сплайна $S_3^{(r)}(x)$, $r=1, 2$ во всех внутренних узлах x_i , $i=1, \dots, n-1$ сетки Δ дает $2(n-1)$ равенств.

Вместе получается $4N-2$ уравнений.

Два дополнительных условия обычно задаются в виде ограничений на значение производных сплайна на концах промежутка $[a, b]$ и называются краевыми условиями.

Наиболее употребительны следующие типы краевых условий:

- а) $S'_3(a)=f'(a)$, $S'_3(b)=f'(b)$;
- б) $S''_3(a)=f''(a)$, $S''_3(b)=f''(b)$;
- в) $S_3^{(r)}(a)=S_3^{(r)}(b)$, $r=1, 2$;
- г) $S'''_3(x_{p+0})=S'''_3(x_{p-0})$, $p=1, n-1$.

Через краевые условия в конструкцию сплайна включаются параметры, выбирая которые можно управлять его поведением, особенно возле концов отрезка $[a, b]$.

Условия типа в) носят названия периодических. Естественно требовать их выполнения в том случае, когда интерполируемая функция периодическая с периодом $(b-a)$.

Если известны $f'(x)$ или $f''(x)$ в точках a и b , то естественно воспользоваться краевыми условиями типа а) или б).

Если производные неизвестны, то в большинстве случаев наилучшим решением будет применение краевых условий типа г).

Вместо значений производных можно использовать их разностные аналоги. При этом точность интерполяции вблизи концов падает.

Иногда предлагается принимать

$$S''_3(a)=S''_3(b)=0.$$

В этом случае вблизи концов точность интерполяции функции и ее первой производной уменьшается и становится соизмеримой с точностью интерполяции сплайном первой степени, что резко ухудшает всю картину.

Алгоритм построения интерполяционного кубического сплайна

Пусть каждому значению аргумента $x_i, i=0,...,n$ соответствуют значения функции $f(x_i)=y_i$ и требуется найти функциональную зависимость в виде сплайна (4.4), удовлетворяющего перечисленным ниже требованиям:

- а) функция $S_3(x_i)$ непрерывна вместе со своими производными до второго порядка включительно;
- б) $S_3(x_i)=y_i, i=0,1,...,n$;
- в) $S''_3(x_0)=S''_3(x_n)=0$.

Сформулированная выше задача имеет единственное решение.

Вторая производная $S''_3(x)$ непрерывна и, как видно из выражения (1.1), линейна на каждом отрезке $[x_{i-1}, x_i], (i=1,...,n)$, поэтому представим ее в виде

$$S''_3(x) = m_{i-1} \frac{x_i - x}{h_i} + m_i \frac{x - x_{i-1}}{h_i}, \quad (4.6)$$

где $h_i = x_i - x_{i-1}$, $m_i = S''_3(x_i)$.

Интегрируя обе части равенства (4.6), получим

$$S_3(x) = m_{i-1} \frac{(x_i - x)^3}{6h_i} + m_i \frac{(x - x_{i-1})^3}{6h_i} + A_i \frac{x_i - x}{h_i} + B_i \frac{x - x_{i-1}}{h_i}, \quad (4.7)$$

где A_i и B_i - постоянные интегрирования.

Пусть в (1.5) $x=x_i$ и $x=x_{i-1}$, тогда используя условия б), получим

$$y_i = m_i \frac{h_i^2}{6} + B_i, y_{i-1} = m_{i-1} \frac{h_i^2}{6} + A_i, \quad i=1,...,n.$$

Из этих уравнений находим A_i и B_i , и окончательно формула (4.4) принимает вид

$$S_3(x) = \frac{x_i - x}{h_i} y_{i-1} + \frac{x - x_{i-1}}{h_i} y_i +$$

$$+ \frac{(x_i - x)^3 - h_i^2 (x_i - x)}{6h_i} m_{i-1} + \frac{(x - x_{i-1})^3 - h_i^2 (x - x_{i-1})}{6h_i} m_i. \quad (4.8)$$

Из формулы (4.5) находим односторонние пределы производной в точках x_1, x_2, \dots, x_{n-1} :

$$S'_3(x_i - 0) = \frac{h_i}{6} m_{i-1} + \frac{h_i}{3} m_i + \frac{y_i - y_{i-1}}{h_i}, \quad (4.9)$$

$$S'_3(x_i + 0) = -\frac{h_{i+1}}{3} m_i - \frac{h_{i+1}}{6} m_{i+1} + \frac{y_{i+1} - y_i}{h_{i+1}}. \quad (4.10)$$

Приравнявая выражения (4.9) и (5.0) для $i=1, \dots, n-1$, получим $n-1$ уравнение

$$\frac{h_i}{6} m_{i-1} + \frac{h_i + h_{i+1}}{3} m_i + \frac{h_{i+1}}{6} m_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \quad (4.11)$$

с $n-1$ неизвестными m_i ($i=1, \dots, n-1$). Согласно условию (1.4) $m_0 = m_n = 0$.

Система линейных алгебраических уравнений (5.1) имеет трехдиагональную матрицу с диагональным преобладанием. Такие матрицы являются неособенными. Поэтому неизвестные m_1, m_2, \dots, m_{n-1} находятся однозначно.

Они могут быть найдены итерационными и прямыми методами решения систем линейных алгебраических уравнений, в том числе и методом прогонки.

Метод прогонки

Пусть имеется система уравнений, записанная в матричном виде:

$$\begin{pmatrix} a_1 & b_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ c_2 & a_2 & b_2 & 0 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & c_{n-2} & a_{n-2} & b_{n-2} \\ 0 & 0 & 0 & 0 & \dots & 0 & c_{n-1} & a_{n-1} \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-2} \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}. \quad (4.12)$$

В нашем случае согласно (4.11)

$$c_i = \frac{h_i}{6}, \quad a_i = \frac{h_i + h_{i+1}}{3}, \quad b_i = \frac{h_{i+1}}{6}, \quad d_i = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}$$

Решение системы ищется в виде

$$m_i = \lambda_i m_{i+1} + \mu_i, \quad i=1, \dots, n-1, \quad (4.13)$$

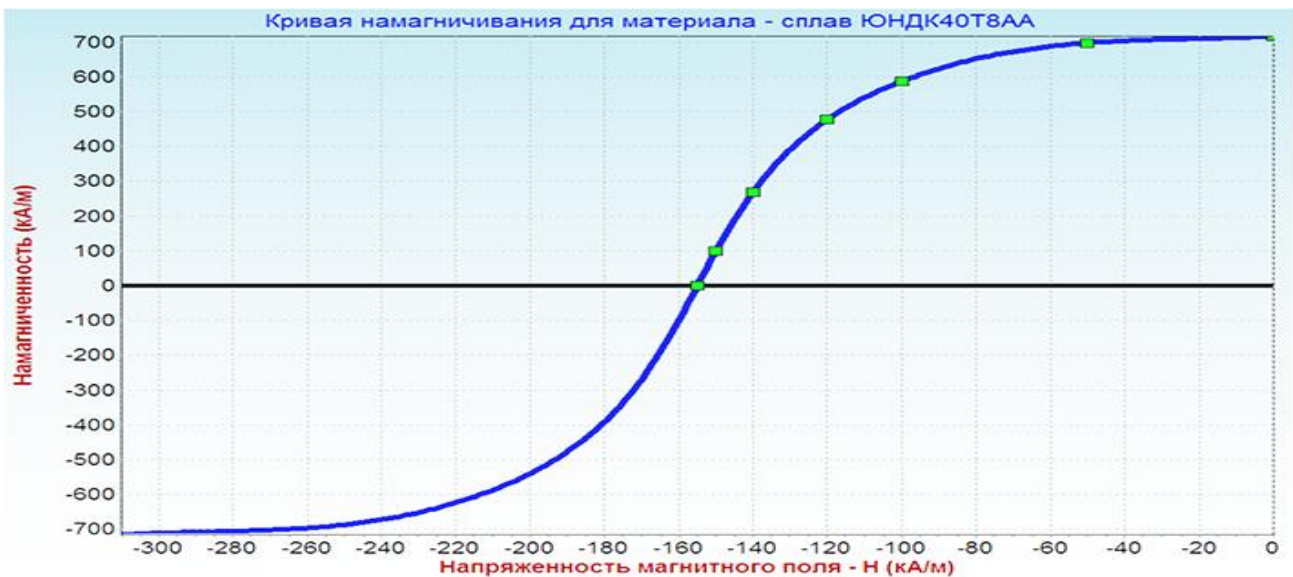
где A_i, B_i - прогоночные коэффициенты. Используя выражение для m_{i-1} из (4.13), исключим это неизвестное из i -го уравнения системы. Получаем $(a_i + c_i \lambda_{i-1})m_i + b_i m_{i+1} = d_i - c_i \mu_{i-1}$.

Сравнивая это соотношение с (4.13), выводим рекуррентные формулы для прогоночных коэффициентов λ_i, μ_i (прямая прогонка):

$$\lambda_0 = \mu_0 = 0, \quad \lambda_i = \frac{-b_i}{a_i + c_i \lambda_{i-1}}, \quad \mu_i = \frac{d_i - c_i \mu_{i-1}}{a_i + c_i \lambda_{i-1}}, \quad i = 1, \dots, n-1. \quad (4.14)$$

Примеры численной реализации:

На рис.4.3 представлен пример интерполяции табличных данных кривой намагничивания материала ЮНДК40Т8АА «Альнико» по 7 точкам во втором квадранте. Использован кубический сплайн и программа, приведенная в приложении 4.1. При слабом шевелении внутренних точек появляется характерный выброс – петелька, связанный с жестким требованием непрерывности вторых производных. Для обеспечения большей устойчивости в



этих случаях советуют использовать сплайны Эрмита.

Рис.4.3

Кривые Безье

Кривые Безье были разработаны в 60-х годах XX века независимо друг от друга Пьером Безье (Bezier) из автомобилестроительной компании «Рено» и Полем де Кастелье (de Casteljau) из компании «Ситроен», где применялись для

проектирования кузовов автомобилей. Впоследствии это открытие стало одним из важнейших инструментов систем автоматизированного проектирования и программ компьютерной графики.

Кривая Безье — это параметрическая кривая, задаваемая выражением

$$\mathbf{B}(t) = \sum_{i=0}^n \mathbf{P}_i \mathbf{b}_{i,n}(t), \quad 0 < t < 1$$

где \mathbf{P}_i — функция компонент векторов опорных вершин, а $\mathbf{b}_{i,n}(t)$ — базисные функции кривой Безье, называемые также полиномами Бернштейна.

$$\mathbf{b}_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

$$\binom{n}{i} = \frac{n!}{i!(n-i)!},$$

где n — степень полинома, i — порядковый номер опорной вершины.

Нас, прежде всего, будут интересовать кубические кривые Безье, являющиеся аналогом кубического сплайна, отличающегося возможностью визуализации параметров кубического полинома. В параметрической форме кубическая кривая Безье ($n = 3$) описывается следующим уравнением:

$$\mathbf{B}(t) = (1-t)^3 \mathbf{P}_0 + 3t(1-t)^2 \mathbf{P}_1 + 3t^2(1-t) \mathbf{P}_2 + t^3 \mathbf{P}_3, \quad t \in [0, 1].$$

Четыре опорные точки \mathbf{P}_0 , \mathbf{P}_1 , \mathbf{P}_2 и \mathbf{P}_3 , заданные в 2-х или 3-мерном пространстве определяют форму кривой. Линия берёт начало из точки \mathbf{P}_0 направляясь к \mathbf{P}_1 и заканчивается в точке \mathbf{P}_3 подходя к ней со стороны \mathbf{P}_2 . То есть кривая не проходит через точки \mathbf{P}_1 и \mathbf{P}_2 , они используются для указания её направления. Длина отрезка между \mathbf{P}_0 и \mathbf{P}_1 определяет, как скоро кривая повернёт к \mathbf{P}_3 .

В матричной форме кубическая кривая Безье записывается следующим образом:

$$\mathbf{B}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \mathbf{M}_B \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix},$$

где M_B называется базисной матрицей Безье:

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

В современных графических системах, таких как PostScript, GIMP для представления криволинейных форм используются сплайны Безье, составленные из кубических кривых.

Благодаря простоте задания и манипуляции, кривые Безье нашли широкое применение в компьютерной графике для моделирования гладких линий. Кривая целиком лежит в выпуклой оболочке своих опорных точек. Это свойство кривых Безье с одной стороны значительно облегчает задачу нахождения точек пересечения кривых (если не пересекаются выпуклые оболочки опорных точек, то не пересекаются и сами кривые), а с другой стороны позволяет осуществлять интуитивно понятное управление параметрами кривой в графическом интерфейсе с помощью её опорных точек.

В качестве дополнительного задания предлагается построить кубические Кривые Безье с использованием встроенных возможностей библиотек .Net.

Численное дифференцирование.

Очевидно, что использование методов многоточечной интерполяции в ряде случаев может существенно снизить погрешности численного дифференцирования (по существу являющегося неустойчивой численной операцией). Для получения многоточечных схем могут быть использованы изученные нами полиномы Лагранжа. На рис.4.4 приведены коэффициенты для многоточечной аппроксимации производных центральными разностями.

Аппроксимация производных центральными разностями

Производная	По трем точкам	По пяти точкам	По семи точкам
y'_0	$\frac{1}{2h}(y_1 - y_{-1}) - \left(\frac{h^2}{6} y''' \right)$	$\frac{1}{12h}(-y_2 + 8y_1 - 8y_{-1} + y_{-2}) + \left(\frac{h^4}{30} y^{(5)}_0\right)$	$\frac{1}{60h}(y_3 - 9y_2 + 45y_1 - 45y_{-1} + 9y_{-2} - y_{-3}) - \left(\frac{h^6}{140} y^{(7)}_0\right)$
y''_0	$\frac{1}{h^2}(y_1 - 2y_0 + y_{-1}) - \left(\frac{h^2}{12} y^{(4)} \right)$	$\frac{1}{12h^2}(-y_2 + 16y_1 - 30y_0 + 16y_{-1} - y_{-2}) - \left(\frac{h^4}{90} y^{(6)}_0\right)$	$\frac{1}{180h^2}(2y_3 - 27y_2 + 270y_1 - 490y_0 + 270y_{-1} - 27y_{-2} + 2y_{-3}) - \left(\frac{h^6}{560} y^{(8)}_0\right)$
y'''_0		$\frac{1}{2h^3}(y_2 - 2y_1 + 2y_{-1} - y_{-2}) - \left(\frac{h^2}{4} y^{(5)}_0\right)$	$\frac{1}{8h^3}(-y_3 + 8y_2 - 13y_1 + 13y_{-1} - 8y_{-2} + y_{-3}) + \left(\frac{7}{120} h^4 y^{(7)}_0\right)$

Примечание. В скобках за формулами указаны погрешности.

Рис.4.4

Приложение 4.1. Программа переписанная нами с Фортрана на Паскаль

```

PROCEDURE spline(N:integer; x,y:array of real;
var b,c,d:array of real);
{ spline - вызывается для данного набора данных один раз!
  N-число точек, x,y- массивы коорд. и значений функции,
  b,c,d- массивы коэфф. сплайна, вычисляемые в spline}
Var i,NM1,ib:integer; t:real;
begin
  NM1:=N-1; if N<2 then exit { меньше двух точек нельзя! }
  else if N>2 then
  begin
    d[1]:=x[2]-x[1];
    c[2:]=(y[2]-y[1])/d[1];
    for i:=2 to NM1 do begin
      d[i]:=x[i+1]-x[i];
      b[i]:=2*(d[i-1]+d[i]);
      c[i+1:]=(y[i+1]-y[i])/d[i];
      c[i]:=c[i+1]-c[i];
    end
  end

```

```

end;
b[1]:=-d[1];    b[N]:=-d[N-1];
                c[1]:=0;    c[N]:=0;
if N<>3 then begin
    c[1]:=c[3]/(x[4]-x[2])-c[2]/(x[3]-x[1]);
    c[N]:=c[N-1]/(x[N]-x[N-2])-c[N-2]/(x[N-1]-x[N-3]);
    c[1]:=c[1]*sqr(d[1])/(x[4]-x[1]);
    c[N]:=-c[N]*sqr(d[N-1])/(x[N]-x[N-3]);
end;
for i:=2 to N do begin
    t:=d[i-1]/b[i-1];
    b[i]:=b[i]-t*d[i-1];
    c[i]:=c[i]-t*c[i-1];
end;
c[N]:=c[N]/b[N];
for ib:=1 to NM1 do begin
    i:=N-ib;
    c[i]:=(c[i]-d[i]*c[i+1])/b[i];
end;
b[N]:=(y[N]-y[NM1])/d[NM1]+d[NM1]*(c[NM1]+2*c[N]);
for i:=1 to NM1 do begin
    b[i]:=(y[i+1]-y[i])/d[i]-d[i]*(c[i+1]+2*c[i]);
    d[i]:=(c[i+1]-c[i])/d[i];
    c[i]:=3*c[i];
end;
c[N]:=3*c[N];
d[N]:=d[N-1];
end
{----- для двух точек линейная интерполяция -----}
else if N=2 then begin
    b[1]:=(y[2]-y[1])/(x[2]-x[1]);
    c[1]:=0;
    d[1]:=0;
    b[2]:=b[1];

```

```

    c[2]:=0;
    d[2]:=0;
end;
end;
FUNCTION seval(var ipar:integer;N:integer;u:real;x,y,b,c,d:array of
real):real;
{ ipar- экономит время при последующих вызовах, первый раз ipar=1, u -
абсцисса, для которой вычисляется сплайн
    seval - вызывается после spline,  $y=y(i)+b*dx+c*dx^2+d*dx^3$  }
var i,j,k:integer;    dx:real;
begin
    i:=1;
{----- ищем отрезок, на который попадает u -----}
    if (u<x[i]) or (u>x[i+1]) then begin
        j:=N+1;
        repeat begin
            k:=round((i+j)/2);
            if u<x[k] then j:=k;
            if u>=x[k] then i:=k;
        end until j<=i+1;
    end;
    dx:=u-x[i]; seval:=y[i]+dx*(b[i]+dx*(c[i]+dx*d[i])); ipar:=i;
end;

```

Лабораторная работа №5

АППРОКСИМАЦИЯ ТРИГОНОМЕТРИЧЕСКИМИ ФУНКЦИЯМИ. РЯДЫ ФУРЬЕ.

Перед выполнением лабораторной работы рекомендуем: Изучить презентацию лектора курса: «Аппроксимация тригонометрическими функциями. Ряды Фурье», материалы доступны в облаке на Mail.ru.

Прочитать соответствующие разделы книги:

Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ.- М.: Мир, 1998.- 575 с.,

Цель работы:

1. Изучить и научиться использовать на практике алгоритмы, основанные на аппроксимации тригонометрическими функциями.
2. Написать программу, реализующую разложение функций в ряд Фурье.
Пример подготовить самостоятельно, обеспечив возможность изменения вида функций.
3. Указание: коэффициенты разложения в ряд находить с помощью численного интегрирования, воспользоваться собственной программой из лаб. работы №1.

Дополнительные задания (бонусы):

Реализовать алгоритм быстрого преобразования Фурье;

Содержание отчета:

1. Постановка задачи. Формульное описание алгоритма.
2. Описание компьютерной программы. Объяснение значений всех параметров.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Ряды Фурье и преобразование Фурье представляют собой одно из крупнейших достижений прикладной математики и были впервые введены в нее Даниилом Бернулли в 50-х годах восемнадцатого века при изучении колебаний струны. Почти 60 годами позднее в процессе изучения теплопроводности Жан Батист Фурье использовал аналогичный, но в большей степени формализованный метод. Ряды и преобразования Фурье используются почти во всех областях науки, включая теорию электрических цепей, теорию механических и колебательных систем, оптику, акустику и термодинамику.

На сессии Французской академии наук 21 декабря 1807 года математик и инженер Жан Батист Жозеф Фурье, которому был тогда 21 год, сделал утверждение, открывшее новую главу в истории математики. «Фурье заявил, что произвольную функцию, заданную на конечном интервале произвольным, сколь угодно причудливым графиком, всегда можно разложить в сумму чистых синусоидальных функций. Академикам, включая великого аналитика Лагранжа, это показалось совершенно невероятным. В конце концов, любая суперпозиция таких функций не может дать ничего другого, кроме бесконечно дифференцируемой функции, называемой «аналитической», что очень далеко от заявленной Фурье произвольности функции. Конечно, последующие исследования показали, что утверждение Фурье было абсолютно справедливо, хотя сам он не мог предложить строгих доказательств этого.

При более широком взгляде можно выделить 3 вопроса:

- 1) Интегральное преобразование Фурье, которое переводит вещественную функцию в пару вещественных функций или одну комплексную функцию в другую.
- 2) Ряд Фурье, который ставит в соответствие периодической функции последовательность ее коэффициентов Фурье.
- 3) Дискретное преобразование Фурье, переводящее одну последовательность чисел в другую. Его можно выполнить при помощи вычислительного метода, который называется быстрым преобразованием Фурье;

Аппроксимация периодических функций рядами Фурье

Общий вид аппроксимирующего полинома имеет вид:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^M (a_k \cos k\omega x + b_k \sin k\omega x), \quad (5.1)$$

где $a_0 / 2 = f_{\text{среднее}}$,

$$a_0 = \frac{2}{T} \cdot \int_0^T f(x) \cdot dx \quad (5.2)$$

Коэффициенты разложения в ряд вычисляются по формулам

$$a_k = \frac{2}{T} \cdot \int_0^T f(x) \cdot \cos k\omega x \cdot dx, \quad (5.3)$$

$$b_k = \frac{2}{T} \cdot \int_0^T f(x) \cdot \sin k\omega x \cdot dx. \quad (5.4)$$

Если функция – четная, то коэффициенты $b_k=0$, если нечетная, то $a_k=0$.

Если функция не периодическая, то ее значения на отрезке $[a,b]$ могут быть аппроксимированы с помощью тригонометрического ряда, но саму функцию для этого необходимо аналитически продолжить, сделав периодической.

Аппроксимация периодической функции, заданной дискретно на равномерной сетке. Пусть функция $f(x)$ задана на отрезке $[0, 2\pi]$ таблицей значений

$f(x_i)$ в равностоящих узлах $x_i = \frac{2\pi(i-1)}{2N+1}$ ($i = 1, 2, \dots, 2N+1$). Тригонометрическим

многочленом степени M называют многочлен

$$P_M(x) = \frac{a_0}{2} + \sum_{k=1}^M (a_k \cos kx + b_k \sin kx).$$

Задача тригонометрической интерполяции состоит в построении тригонометрического интерполяционного многочлена наименьшей степени, удовлетворяющего условиям $P_M(x_i) = f(x_i), i = 1, \dots, 2N+1$. Можно показать, что решением этой задачи является тригонометрический многочлен

$$P_N(x) = \frac{a_0}{2} + \sum_{k=1}^N (a_k \cos kx + b_k \sin kx),$$

коэффициенты, которого вычисляются по следующим формулам:

$$a_0 = \frac{1}{2N+1} \sum_{i=1}^{2N+1} f(x_i),$$

$$a_k = \frac{1}{2N+1} \sum_{i=1}^{2N+1} f(x_i) \cos\left(\frac{2\pi k}{2N+1} i\right),$$

$$b_k = \frac{1}{2N+1} \sum_{i=1}^{2N+1} f(x_i) \sin\left(\frac{2\pi k}{2N+1} i\right).$$

Широкие возможности тригонометрической интерполяции следуют из этого факта, что с возрастанием N многочлен $P(x)$ аппроксимирует $f(x)$ с возрастающей точностью, т.е.

$$\sup_{x \in [0, 2\pi]} |f(x) - P_N(x)| \rightarrow 0, N \rightarrow \infty,$$

это утверждение справедливо для достаточно широкого класса функций. Этим тригонометрическая интерполяция существенно отличается от алгебраической интерполяции на системе равноотстоящих узлов. При алгебраическом интерполировании разность между функцией $f(x)$ и интерполяционным многочленом может быть как угодно большой всюду, кроме узлов интерполяции. Тригонометрическое интерполирование полностью свободно от этого недостатка.

Программа, реализующая алгоритм разложения периодических функций в ряд Фурье

На рис. 5.1 показан вариант интерфейса программы, на рис. 5.2 предложен вариант интерфейса выбора периодической функции из заданного набора функций.

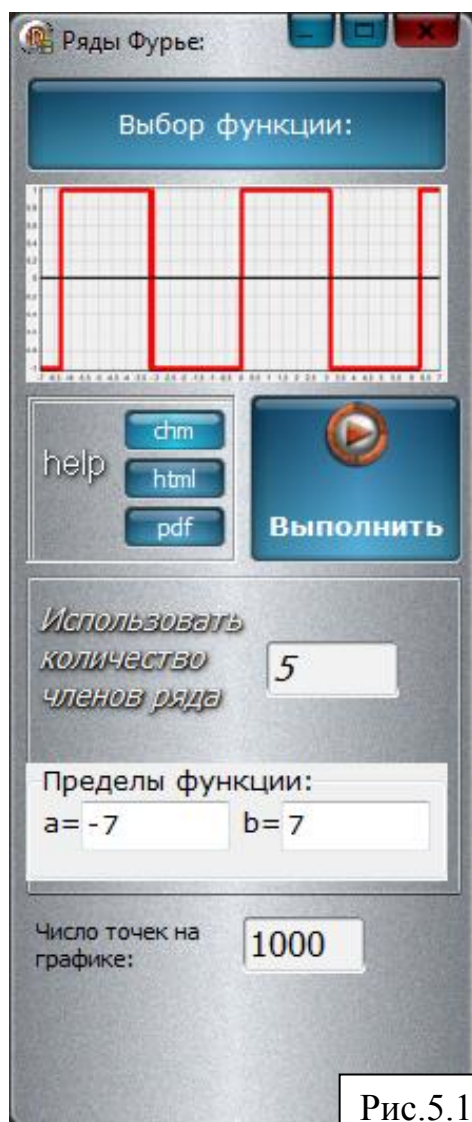


Рис.5.1

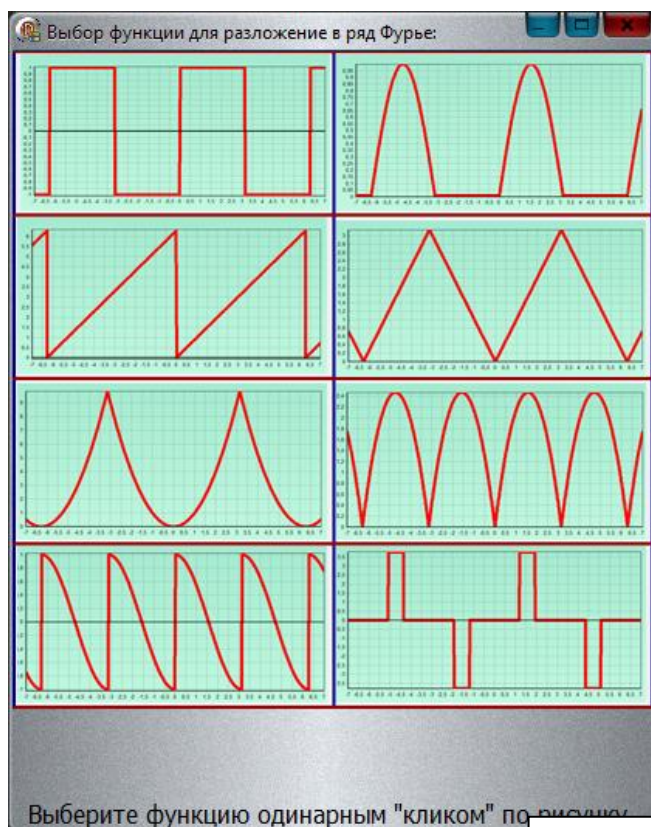
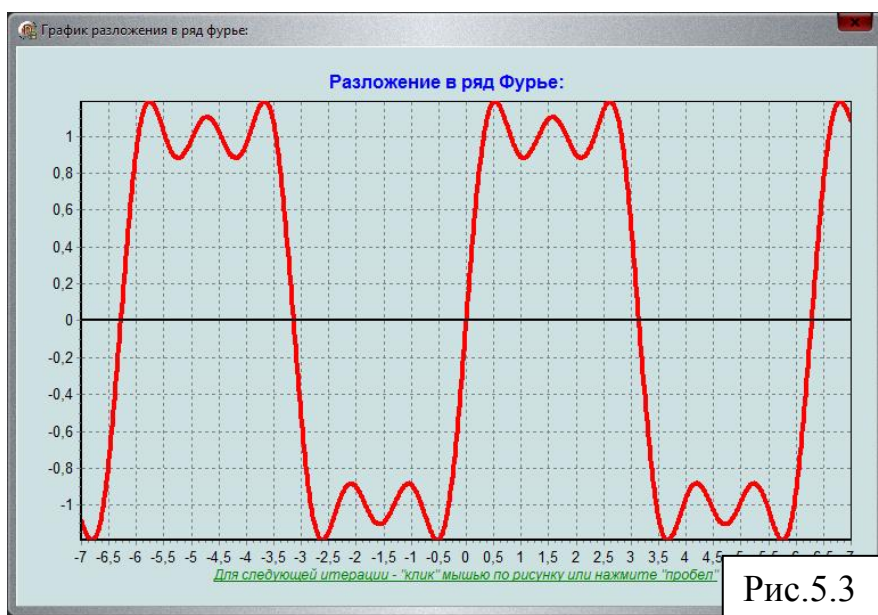


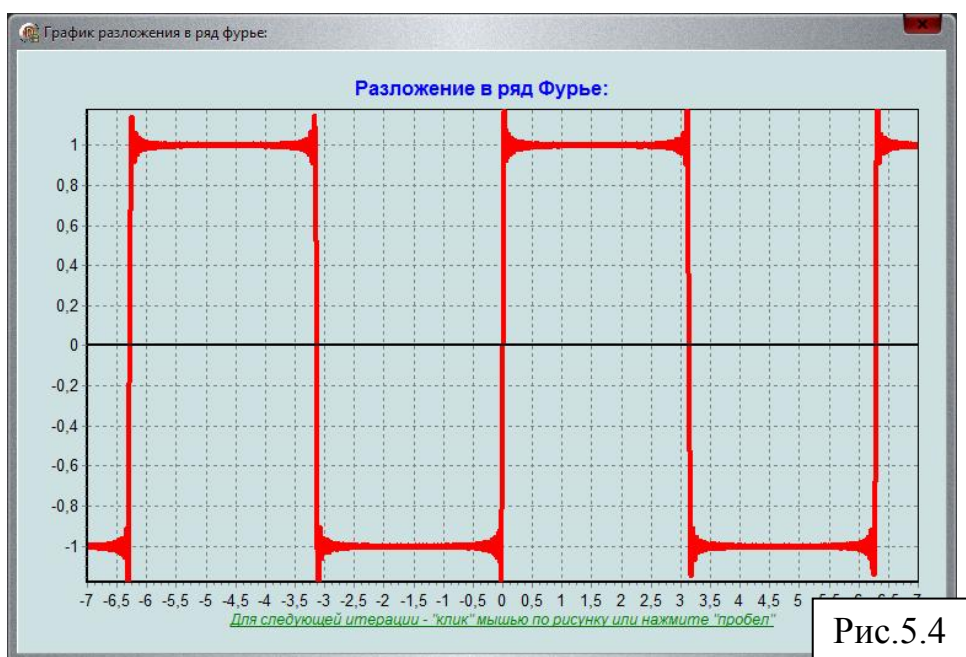
Рис.5.2

Коэффициенты разложения для приведенных на рисунке функций легко могут быть вычислены аналитически, однако, для выполнения и зачета лабораторной работы требуется использовать численный метод интегрирования, реализованный в первой лабораторной работе.

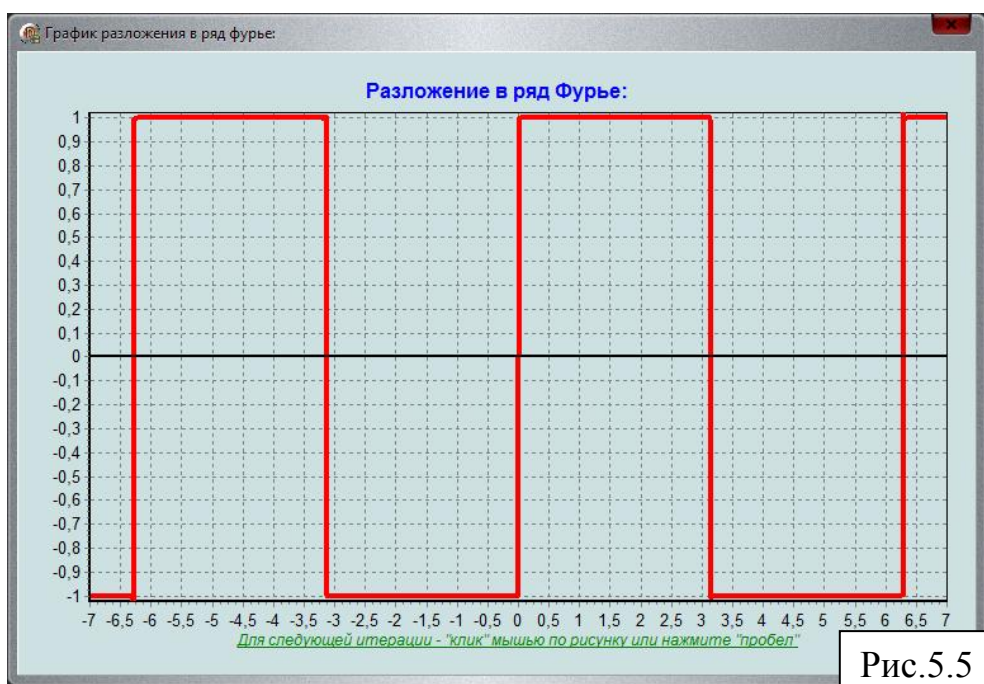
На рис. 5.3 приведен график разложения меандра в ряд Тейлора с использованием трех гармоник (1,3 и пятая).



На рис. 5.4 приведен график разложения меандра в ряд Тейлора с использованием трех гармоник (1,3 и пятая).



На рис. 5.4 приведен график разложения меандра в ряд Тейлора с использованием пятидесяти гармоник. Выбросы на фронтах связаны с плохой сходимостью ряда в точках разрыва (сумма конечного числа дифференцируемых функций является тоже дифференцируемой, в отличие от меандра). На рис. 5.5 приведен график разложения меандра в ряд Тейлора с использованием десяти тысяч гармоник.



Здесь выбросы уже не видны, но они, конечно, имеются.

Обращаем ваше внимание на тот факт, что здесь интегрирование производилось аналитически. Если использовать численное интегрирование, то скорость работы программы значительно снизится. При этом особое внимание нужно обратить на интегрирование высокочастотной составляющей. Для достижения приемлемой погрешности может потребоваться большое количество разбиений.

Лабораторная работа № 6

МНОГОМЕРНАЯ ОПТИМИЗАЦИЯ НЕПРЕРЫВНЫХ ФУНКЦИЙ

Перед выполнением лабораторной работы рекомендуем:

1. Изучить теоретические сведения в описании данной лабораторной работы.
2. Прочитать соответствующие разделы книг:

Бахвалов, Н.С. Численные методы : учебник / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. — 9-е изд. — Москва : Лаборатория знаний, 2020. — 636 с. — ISBN 978-5-00101-836-0. — Текст : электронный // Электронно-

библиотечная система «Лань» : [сайт]. — URL:

<https://e.lanbook.com/book/126099> (дата обращения: 31.10.2019). — Режим доступа: для авториз. пользователей.

Таха Х. Введение в исследование операций.: Пер. с англ. — М.: Издательский дом «Вильямс», 2001

Смородинский С.С., Батин Н.В. Оптимизация решений на основе методов и моделей математического программирования: Уч. пособие. — Минск: БГУИР, 2003. — 136 с.

Лемешко, Б.Ю. Методы оптимизации: Конспект лекций. — Новосибирск: Изд-во НГТУ, 2009. — 126 с.

Цель работы:

1. Изучить и научиться использовать на практике: а) методы решения задач линейного программирования; б) методы минимизации нелинейных функционалов без ограничений.
2. Написать программу, реализующую алгоритм: а) решения задач линейного программирования на основе симплекс-метода; б) минимизации нелинейных функционалов без ограничений.

Содержание отчета:

1. Задачи линейного программирования:
 - математическая модель, ее приведение к стандартной форме;
 - обоснование выбранной схемы симплекс-метода;

- по окончании решения задачи указать оптимальные значения всех переменных (включая остаточные и избыточные) и целевой функции, с указанием их содержательного смысла и размерности.
2. Минимизация нелинейных функционалов без ограничений: сравнить эффективности различных алгоритмов минимизации.
3. Описание компьютерной программы. Объяснение значений всех параметров.

2. Минимизация нелинейных функционалов без ограничений: сравнить эффективности различных алгоритмов минимизации.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. Постановка задачи и основные понятия линейного программирования

1.1. Понятие математической модели. Математическая модель в задачах линейного программирования

Любое описание некоторой решаемой задачи в виде формул, уравнений, алгоритмов называется математической моделью этой задачи.

Линейное программирование рассматривает задачи с линейной математической моделью. Это задачи, в которых требуется найти такие значения переменных X_1, X_2, \dots, X_N , при которых некоторая линейная функция от этих переменных принимает максимальное или минимальное значение:

$$E = C_1 X_1 + C_2 X_2 + \dots + C_n X_n \rightarrow \max / \min \quad (6.1.1)$$

при выполнении ограничений на переменные X_1, X_2, \dots, X_N , заданных линейными уравнениями или неравенствами:

[illegible]

$$A_{m1}X_1 + A_{m2}X_2 + \dots + A_{mn}X_n \leq B_m$$

Здесь p – количество ограничений, имеющих вид «не меньше», q – количество ограничений «равно», m – общее количество ограничений.

Функция (6.1.1) называется целевой функцией, а ограничения (6.1.2) – системой ограничений задачи.

Если по условию задачи требуется найти такие значения переменных X_1, X_2, \dots, X_N , при которых целевая функция (6.1.1) будет иметь максимальное значение, то говорят, что целевая функция подлежит максимизации (или направлена на максимум). Если требуется, чтобы целевая функция принимала минимальное значение, то говорят, что она подлежит минимизации (направлена на минимум).

В большинстве задач (но не всегда) требуется, чтобы переменные X_1, X_2, \dots, X_N принимали неотрицательные значения (ограничение на неотрицательность): $X_j \geq 0, j = 1, \dots, n$.

Любые значения переменных X_1, X_2, \dots, X_N , удовлетворяющие ограничениям задачи (6.1.2), называются допустимыми решениями (независимо от того, какое значение при этом принимает целевая функция). Большинство задач имеет бесконечно много допустимых решений. Все множество допустимых решений представляет собой область допустимых решений (ОДР).

Допустимые значения переменных X_1, X_2, \dots, X_N , при которых целевая функция принимает максимальное или минимальное значение (в зависимости от постановки задачи), представляют собой оптимальное решение.

Пример задач линейного программирования

Предприятие химической промышленности выпускает соляную и серную кислоту. Выпуск одной тонны соляной кислоты приносит предприятию прибыль в размере 25 ден. ед., выпуск одной тонны серной кислоты – 40 ден. ед. Для выполнения государственного заказа необходимо выпустить не менее 200 т соляной кислоты и не менее 100 т серной кислоты. Кроме того, необходимо учитывать, что выпуск кислот связан с образованием опасных отходов. При выпуске одной тонны соляной кислоты образуется 0,5 т опасных отходов, при выпуске одной тонны серной кислоты – 1,2 т опасных отходов. Общее количество опасных отходов не должно превышать 600 т, так как превышение этого ограничения приведет к выплате предприятием крупного штрафа.

Требуется определить, сколько соляной и серной кислоты должно выпустить предприятие, чтобы получить максимальную прибыль.

Составим математическую модель задачи. Для этого введем переменные. Обозначим через X_1 количество выпускаемой соляной кислоты (в тоннах), через X_2 – количество серной кислоты.

Составим ограничения, связанные с необходимостью выполнения государственного заказа. Предприятию необходимо выпустить не менее 200 т соляной кислоты. Это ограничение можно записать следующим образом:

$$X_1 \geq 200.$$

Аналогично составим ограничение, устанавливающее, что предприятие должно выпустить не менее 100 т серной кислоты:

$$X_2 \geq 100.$$

Составим ограничение на опасные отходы. При выпуске одной тонны соляной кислоты образуется 0,5 т опасных отходов; значит, общее количество опасных отходов при выпуске соляной кислоты составит $0,5X_1$ т. При выпуске серной кислоты образуется $1,2X_2$ т опасных отходов. Таким образом, общее количество опасных отходов составит $0,5X_1 + 1,2X_2$ т. Эта величина не должна превышать 600 т. Поэтому можно записать следующее ограничение:

$$0,5X_1 + 1,2X_2 \leq 600.$$

Кроме того, переменные X_1 и X_2 по своему физическому смыслу не могут принимать отрицательных значений, так как они обозначают количество выпускаемых кислот. Поэтому необходимо указать ограничения неотрицательности:

$$X_1 \geq 0; \quad X_2 \geq 0.$$

В данной задаче требуется определить выпуск кислот, при котором прибыль будет максимальной. Прибыль от выпуска одной тонны соляной кислоты составляет 25 ден. ед.; значит, прибыль от выпуска соляной кислоты составит $25X_1$ ден. ед. Прибыль от выпуска серной кислоты составит $40X_2$ ден. ед. Таким образом, общая прибыль от выпуска кислот составит $25X_1 + 40X_2$ ден. ед. Требуется найти такие значения переменных X_1 и X_2 , при которых эта величина будет максимальной. Таким образом, целевая функция для данной задачи будет иметь следующий вид:

$$E = 25X_1 + 40X_2 \rightarrow \max$$

Приведем полную математическую модель рассматриваемой задачи:

$$\begin{aligned}
X_1 &\geq 200 \\
X_2 &\geq 100 \\
0,5X_1 + 1,2X_2 &\leq 600 \\
X_1 &\geq 0, X_2 \geq 0 \\
E = 25X_1 + 40X_2 &\rightarrow \max
\end{aligned}
\tag{6.1.3}$$

$$\tag{6.1.4}$$

В этой задаче имеется два ограничения «больше или равно» и одно ограничение «меньше или равно». Целевая функция подлежит максимизации.

1.2. Приведение задач линейного программирования к стандартной форме

Для большинства методов решения задач линейного программирования требуется предварительно привести задачу к стандартной форме. Задача (или ее математическая модель) представлена в стандартной форме, если она соответствует следующим условиям:

- целевая функция подлежит максимизации;
- все ограничения имеют вид равенств;
- на все переменные накладываются ограничения неотрицательности.

Если целевая функция задачи подлежит минимизации, то для перехода к целевой функции, подлежащей максимизации, необходимо умножить исходную целевую функцию на -1 . Доказано, что максимальное значение любой функции E всегда равно минимальному значению функции $-E$, взятому с обратным знаком. Для преобразования ограничения «больше или равно» в равенство (т.е. в ограничение «равно») необходимо вычесть из левой части ограничения дополнительную переменную. Для преобразования ограничения «меньше или равно» в равенство необходимо прибавить к левой части ограничения дополнительную переменную. На все переменные, используемые для приведения задачи к стандартной форме, накладываются ограничения неотрицательности. Переменные, вычитаемые из ограничений «больше или равно» для их приведения к стандартной форме, называются избыточными, а переменные, прибавляемые к ограничениям «меньше или равно» – остаточными.

Если на какую-либо переменную не накладывается ограничение неотрицательности, то она заменяется на разность двух переменных, каждая из которых должна быть неотрицательной. Таким образом, если некоторая переменная X_j по своему физическому смыслу может принимать как положительные, так и отрицательные значения, то во всех ограничениях и в целевой функции ее следует заменить на разность двух переменных:

$$X'_j - X''_j$$

На эти переменные накладываются ограничения неотрицательности:

$$X'_j \geq 0; \quad X''_j \geq 0.$$

Приведем к стандартной форме задачу из рассмотренного примера. Из ограничений «больше или равно» необходимо вычесть избыточные переменные, к ограничению «меньше или равно» – прибавить остаточную переменную. Целевая функция задачи подлежит максимизации, и на все переменные накладывается ограничение неотрицательности; поэтому никаких других преобразований не требуется. Математическая модель задачи в стандартной форме будет иметь следующий вид:

$$\begin{aligned} X_1 - X_3 &= 200 \\ X_2 - X_4 &= 100 \\ 0,5X_1 + 1,2X_2 + X_5 &= 600 \\ X_j &\geq 0, j = 1, \dots, 5 \\ E &= 25X_1 + 40X_2 \rightarrow \max. \end{aligned}$$

Здесь переменные X_3 и X_4 – избыточные, X_5 – остаточная.

Примечание. Все переменные, которые вводятся в математическую модель для ее приведения к стандартной форме, имеют физический смысл. Так, в рассмотренном примере переменные X_3 и X_4 обозначают количество кислот, которое будет выпущено сверх государственного заказа. Переменная X_5 обозначает, насколько количество опасных отходов, образующихся при производстве кислот, будет меньше максимально допустимой величины (600 т).

2. Решение задач линейного программирования на основе симплекс-метода

2.1. Пример задачи линейного программирования: задача планирования производства

Одной из наиболее распространенных практических задач, решаемых методами линейного программирования, является задача планирования производства при ограниченных ресурсах. Основным методом решения задач линейного программирования – симплекс-метод – будет рассмотрен на примере решения такой задачи.

Пример. Один из цехов машиностроительного предприятия выпускает изделия двух видов: корпуса и задвижки. Для производства этих изделий требуются три вида сырья: алюминий, сталь и пластмасса. На выпуск одного корпуса расходуется 20 кг алюминия, 10 кг стали и 5 кг пластмассы. На выпуск одной задвижки расходуется 5 кг алюминия, 5 кг стали и 20 кг пластмассы.

сы. Запасы ресурсов ограничены: за рабочую смену цех может израсходовать не более 200 кг алюминия, 250 кг стали и 500 кг пластмассы.

Выпуск одного корпуса приносит предприятию прибыль в размере 100 денежных единиц (ден. ед.), одной задвижки – 300 ден. ед.

Требуется составить оптимальный план работы цеха, т.е. найти, сколько корпусов и задвижек требуется выпускать, чтобы получить максимальную прибыль (при соблюдении ограничений на ресурсы).

Для построения математической модели задачи введем переменные. Обозначим через X_1 количество выпускаемых корпусов, через X_2 – количество выпускаемых задвижек.

Составим ограничение на расход алюминия. На выпуск одного корпуса расходуется 20 кг алюминия; значит, расход алюминия на выпуск всех корпусов составит $20X_1$ кг. На выпуск задвижек будет израсходовано $5X_2$ кг алюминия. Таким образом, общий расход алюминия составит $20X_1 + 5X_2$ кг. Эта величина не должна превышать 200 кг, так как цех не может израсходовать за смену свыше 200 кг алюминия. Поэтому можно записать следующее ограничение:

$$20X_1 + 5X_2 \leq 200.$$

Аналогично можно составить ограничение на расход стали:

$$10X_1 + 5X_2 \leq 250$$

и на расход пластмассы:

$$5X_1 + 20X_2 \leq 500.$$

Кроме того, переменные X_1 и X_2 по своему физическому смыслу не могут принимать отрицательных значений, так как они обозначают количество изделий. Поэтому необходимо указать ограничения неотрицательности:

$$X_1 \geq 0, X_2 \geq 0.$$

В данной задаче требуется определить количество выпускаемых изделий, при котором прибыль от их производства будет максимальной. Прибыль от выпуска одного корпуса составляет 100 ден. ед.; значит, прибыль от выпуска корпусов составит $100X_1$ ден. ед. Прибыль от выпуска задвижек составит $300X_2$ ден. ед. Таким образом, общая прибыль от выпуска всех изделий составит $100X_1 + 300X_2$ ден. ед. Требуется найти такие значения переменных X_1 и X_2 , при которых эта величина будет максимальной. Таким образом, целевая функция для данной задачи будет иметь следующий вид:

$$E = 100X_1 + 300X_2 \rightarrow \max.$$

Приведем полную математическую модель рассматриваемой задачи:

$$\begin{aligned} 20X_1 + 5X_2 &\leq 200 \\ 10X_1 + 5X_2 &\leq 250 \end{aligned} \quad (6.2.1)$$

$$5X_1 + 20X_2 \leq 500$$

$$X_1 \geq 0, X_2 \geq 0.$$

$$E = 100 \cdot X_1 + 300X_2 \rightarrow \max. \quad (6.2.2)$$

Примечание. В данной задаче на переменные X_1 и X_2 накладывается также ограничение целочисленности: они должны принимать только целые значения, так как обозначают количество изделий. Если в результате решения задачи эти переменные примут дробные значения, то для получения целочисленного решения потребуется использовать специальные методы, рассматриваемые в лабораторной работе № 7

Рассмотрим решение на основе симплекс-метода, позволяющего решать задачи с любым количеством переменных. Для решения задачи симплекс-методом требуется привести ее к стандартной форме, как показано в подразделе 1.2. Все ограничения задачи имеют вид «меньше или равно». Их необходимо преобразовать в равенства. Для этого требуется добавить в каждое ограничение дополнительную (остаточную) переменную. Математическая модель задачи в стандартной форме будет иметь следующий вид:

$$\begin{aligned} 20X_1 + 5X_2 + X_3 &= 200 \\ 10X_1 + 5X_2 + X_4 &= 250 \end{aligned} \quad (6.2.3)$$

$$5X_1 + 20X_2 + X_5 = 500$$

$$X_j \geq 0, j = 1, \dots, 5.$$

$$E = 100X_1 + 300X_2 \rightarrow \max. \quad (6.2.4)$$

Здесь X_3, X_4, X_5 – остаточные переменные. Их физический смысл будет показан в п. 2.4.

2.2. Принцип работы симплекс-метода

Симплекс-метод позволяет решать задачи линейного программирования любой размерности, т.е. с любым количеством переменных. Решение задач линейного программирования на основе симплекс-метода состоит в целенаправленном переборе угловых точек ОДР в направлении улучшения значения целевой функции. Как отмечено выше, можно доказать, что экстремум (минимум или максимум) целевой функции всегда достигается при значениях переменных X_1, X_2, \dots, X_n , соответствующих одной из угловых точек ОДР. Другими словами, оптимальное решение всегда находится в угловой точке ОДР.

Принцип работы симплекс-метода состоит в следующем. Находится какое-либо допустимое решение, соответствующее одной из угловых точек ОДР. Проверяются смежные с ней угловые точки ОДР. Если ни в одной из смежных угловых точек значение целевой функции не улучшается, то решение задачи завершается; текущая угловая точка ОДР соответствует оптимальному решению задачи. Если имеются смежные угловые точки ОДР, для которых значение целевой функции улучшается, то выполняется переход в ту из них, для которой достигается наиболее быстрое улучшение значения целевой функции. Для новой угловой точки ОДР процесс повторяется, т.е. проверяются смежные угловые точки. Перебор угловых точек происходит до тех пор, пока не будет найдено оптимальное решение, т.е. пока не будет достигнута угловая точка ОДР, для которой ни в одной из смежных точек значение целевой функции не улучшается. Поиск решения на основе симплекс-метода реализуется с помощью симплекс-таблиц. Основные этапы реализации симплекс-метода следующие:

1. Задача линейного программирования приводится к стандартной форме.
2. Определяется начальное допустимое решение (начальная угловая точка ОДР).
3. Строится исходная симплекс-таблица. Выполняются преобразования симплекс-таблиц, соответствующие перебору угловых точек ОДР, до получения оптимального решения.

Реализация симплекс-метода существенно различается в зависимости от вида математической модели задачи. В данном разделе рассматривается реализация симплекс-метода для случая, когда математическая модель задачи состоит только из ограничений «меньше или равно», и целевая функция подлежит максимизации. Реализация симплекс-метода для задач с математической моделью любого вида рассматривается далее.

2.3. Определение начального допустимого решения

В задаче, представленной в стандартной форме, количество переменных обычно больше, чем количество ограничений. Так, в стандартной форме для примера имеются три ограничения ($m = 3$) и пять переменных ($k = 5$). Для определения начального решения $k - m$ переменных принимаются равными нулю. Тогда в системе из m равенств остается m переменных (неизвестных); в этом случае их значения можно определить однозначно. Эти значения используются в качестве начального решения задачи. Переменные, значения которых принимаются равными нулю, называются небазисными, остальные – базисными. Ко-

личество базисных переменных (переменных, составляющих базис), всегда равно количеству ограничений (m).

Начальный базис легко определить, если в каждом ограничении имеется переменная, входящая в это ограничение с коэффициентом, равным единице, и при этом не входящая ни в одно из других ограничений. Эти переменные принимаются в качестве базисных. Все остальные (небазисные) переменные принимаются равными нулю. Таким образом, базисные переменные принимают значения, равные правым частям ограничений.

Такой способ определения начального базиса наиболее удобен при решении задач, для которых математическая модель состоит только из ограничений «меньше или равно». В таких задачах после приведения к стандартной форме в каждом ограничении имеется переменная, входящая в данное ограничение с коэффициентом, равным единице, и не входящая ни в одно из других ограничений. Эти переменные – остаточные переменные, введенные при приведении задачи к стандартной форме. Эти переменные принимаются в качестве базисных. Все остальные переменные (т.е. переменные, входившие в исходную математическую модель задачи), принимаются в качестве небазисных, т.е. равных нулю. Таким образом, в качестве начального допустимого решения (начальной угловой точки ОДР) принимается начало координат, т.е. решение, в котором все исходные переменные математической модели равны нулю: $X_1 = X_2 = \dots = X_n = 0$.

Если базисные переменные присутствуют не во всех ограничениях задачи, то решение $X_1 = X_2 = \dots = X_n = 0$ обычно оказывается недопустимым (не соответствует системе ограничений). В этом случае начало координат не может использоваться в качестве начального допустимого решения (начальной угловой точки ОДР). Для поиска начального допустимого решения в таких случаях используются специальные методы (см. раздел 3). Обычно это требуется для задач, в которых имеются ограничения «не меньше» или «равно».

Найдем начальное допустимое решение. Для задачи, приведенной к стандартной форме (2.3), (2.4), в качестве базисных переменных следует выбрать переменные X_3, X_4, X_5 , так как каждая из них входит только в одно ограничение с коэффициентом, равным единице, и не входит в другие ограничения. Базисные переменные имеются во всех ограничениях задачи. Переменные X_1 и X_2 принимаются равными нулю, т.е. небазисными. Таким образом, начальное решение задачи следующее:

$$X_1 = 0, X_2 = 0, X_3 = 200, X_4 = 250, X_5 = 500.$$

Это решение является допустимым, так как значения $X_1 = X_2 = 0$ соответствуют системе ограничений (2.1). Таким образом, в качестве начальной угловой точки ОДР выбрано начало координат.

Выбранное решение явно не является оптимальным, так как целевая функция $E = 100X_1 + 300X_2$ при этом равна нулю. По своему смыслу это решение означает, что никакие изделия не выпускаются.

2.4. Определение оптимального решения на основе симплекс-таблиц

Как отмечено выше, поиск оптимального решения на основе симплекс-метода состоит в целенаправленном переборе смежных угловых точек ОДР в направлении улучшения значения целевой функции. Можно доказать, что переход из одной угловой точки ОДР в другую (смежную) соответствует замене одной переменной в базисе. Такая замена означает, что одна из небазисных переменных (имевшая нулевое значение) включается в базис, т.е. увеличивается, а одна из базисных переменных уменьшается до нуля, т.е. исключается из базиса. Выбор таких переменных выполняется по определенным правилам, обеспечивающим максимально быстрое увеличение целевой функции.

Рассмотрим алгоритм поиска оптимального решения на основе симплекс-таблиц.

1. Строится исходная симплекс-таблица. Общий вид симплекс-таблицы показан в табл. 6.2.1.

Таблица 6.2.1

Базис	X_1	X_2	...	X_n	X_{n+1}	X_{n+2}	...	X_k	Решение
E	$-C_1$	$-C_2$...	$-C_n$	0	0	0	0	0
X_{n+1}	A_{11}	A_{12}	...	A_{1n}	1	0	0	0	B_1
X_{n+2}	A_{21}	A_{22}	...	A_{2n}	0	1	0	0	B_2
...
X_k	A_{m1}	A_{m2}	...	A_{mn}	0	0	0	1	B_m

Симплекс-таблица строится по следующим правилам:

- в первой строке перечисляются все переменные задачи, как исходные (X_1, X_2, \dots, X_n), так и дополнительные, введенные при приведении к стандартной форме ($X_{n+1}, X_{n+2}, \dots, X_k$). Для задач, содержащих только ограничения «меньше или равно», дополнительные переменные $X_{n+1}, X_{n+2}, \dots, X_k$ – это остаточные переменные;

- в первой колонке таблицы («Базис») перечисляются переменные, составляющие начальный базис задачи. Их количество всегда равно количеству ограничений. Для задач, содержащих только ограничения «меньше или равно», начальный базис состоит из остаточных переменных $X_{n+1}, X_{n+2}, \dots, X_k$. В этой же колонке указывается обозначение целевой функции E ;
- в строке целевой функции указываются коэффициенты целевой функции с обратным знаком. Для переменных, не входящих в целевую функцию (например, для остаточных переменных $X_{n+1}, X_{n+2}, \dots, X_k$), указываются нули;
- в строках базисных переменных указываются коэффициенты ограничений, в которые входят эти переменные. Для переменных, не входящих в ограничения, указываются нулевые коэффициенты;
- в последнем столбце («Решение») указываются значения базисных переменных (они равны правым частям ограничений), а также начальное значение целевой функции (0).

Если таблица построена правильно, то в столбце каждой базисной переменной должна присутствовать только одна единица (в строке ограничения, в которое входит эта переменная); остальные коэффициенты – нулевые.

Исходная симплекс-таблица для рассматриваемой задачи приведена в табл. 2.2.

2. Проверяется условие окончания решения задачи. Если в строке целевой функции (E -строке) все коэффициенты неотрицательны, это означает, что оптимальное решение найдено. В противном случае выполняется следующий шаг.

Таблица 6.2.2

Базис	X_1	X_2	X_3	X_4	X_5	Решение
E	-100	-300	0	0	0	0
X_3	20	5	1	0	0	200
X_4	10	5	0	1	0	250
X_5	5	20	0	0	1	500

3. Определяется переменная для включения в базис. В качестве такой переменной выбирается переменная, которой соответствует максимальный по модулю отрицательный коэффициент в E -строке. Включение в базис (т.е. увеличение) такой переменной приводит к наиболее быстрому росту целевой функции. Столбец переменной, выбранной для включения в базис, называется ведущим (разрешающим).

Для рассматриваемого примера в базис необходимо включить переменную X_2 , так как ей соответствует максимальный по модулю отрицательный коэффициент E -строки (-300). Это означает увеличение выпуска задвижек. Из условия задачи и целевой функции видно, что увеличение выпуска задвижек приводит к более быстрому росту целевой функции, чем увеличение выпуска корпусов: выпуск каждой задвижки увеличивает целевую функцию (прибыль) на 300 ден. ед., а выпуск каждого корпуса – только на 100 ден. ед.

Примечание. Если в E -строке имеется несколько максимальных по модулю отрицательных элементов (равных между собой), то для включения в базис можно выбирать любую из соответствующих переменных.

4. Определяется переменная для исключения из базиса. Для этого вычисляются отношения значений базисных переменных (указанных в столбце «Решение») к соответствующим элементам ведущего столбца. Такие отношения (называемые симплексными отношениями) вычисляются только для положительных коэффициентов ведущего столбца. Переменная, которой соответствует минимальное симплексное отношение, исключается из базиса.

Строка переменной, выбранной для исключения из базиса, называется ведущей (разрешающей). Элемент на пересечении ведущей строки и столбца называется ведущим (разрешающим) элементом.

Найдем симплексные отношения для рассматриваемого примера:

$$200/5 = 40, \quad 250/5 = 50, \quad 500/20 = 25.$$

Минимальное симплексное отношение получено для последней строки, соответствующей базисной переменной X_5 . Значит, эта переменная исключается из базиса (становится равной нулю).

Такой способ определения переменной, исключаемой из базиса, имеет следующее обоснование. При включении в базис новой переменной ее значение увеличивается. Чтобы по-прежнему соблюдались ограничения (6.2.1), необходимо в каждом из ограничений уменьшать базисные переменные. Увеличение

переменной, включаемой в базис, возможно только до тех пор, пока одна из базисных переменных не станет равной нулю. Минимальное симплексное отношение показывает, какая из базисных переменных первой уменьшается до нуля (т.е. исключается из базиса). Так, для рассматриваемого примера, при увеличении переменной X_2 (т.е. при ее включении в базис) для соблюдения ограничений (2.1) необходимо уменьшать переменные X_3 , X_4 , X_5 . Например, при каждом увеличении переменной X_2 на единицу необходимо уменьшать переменную X_3 на 5, X_4 – также на 5, X_5 – на 20. Минимальное симплексное отношение показывает, что при увеличении X_2 переменная X_5 первой достигнет нулевого значения. Смысл симплексных отношений для данной задачи следующий: они показывают, что имеющегося запаса алюминия (200 кг) хватит на выпуск 40 задвижек, запаса стали (250 кг) – на 50 задвижек, запаса пластмассы (500 кг) – на 25 задвижек. Таким образом, запас пластмассы будет израсходован первым, поэтому из базиса исключается переменная X_5 . Ниже будет показано, что переменная X_5 обозначает остаток запаса пластмассы.

Примечания:

1. Если имеется несколько минимальных симплексных отношений (равных между собой), то для исключения из базиса можно выбирать любую из соответствующих переменных.

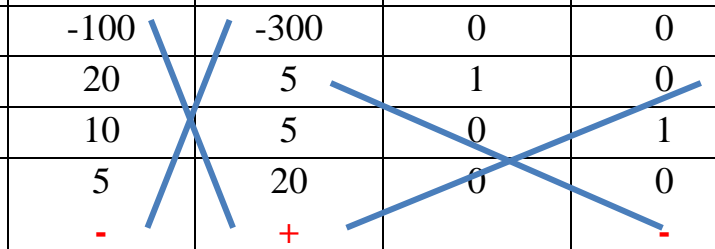
2. Если все элементы ведущего столбца оказываются отрицательными или равными нулю (т.е. невозможно вычислить ни одного симплексного отношения), это означает, что переменную, включаемую в базис, можно увеличивать на любую величину, не нарушая ни одного из ограничений задачи. Целевая функция при этом также может увеличиваться бесконечно. Обычно такой случай означает, что допущена ошибка в постановке задачи или в математической модели (не учтено некоторое ограничение).

5. Выполняется преобразование симплекс-таблицы по следующим правилам:

- в столбце «Базис» вместо переменной, исключенной из базиса на шаге 4, указывается переменная, включенная в базис на шаге 3;
- все элементы ведущей строки делятся на ведущий элемент;
- все элементы ведущего столбца (кроме ведущего элемента) заменяются нулями;

- все остальные элементы таблицы (включая E -строку и столбец «Решение») пересчитываются по «правилу прямоугольника». Этот пересчет выполняется следующим образом: ведущий и пересчитываемый элемент образуют диагональ прямоугольника; находится произведение ведущего и пересчитываемого прямоугольника; из этого произведения вычитается произведение элементов, образующих противоположную диагональ прямоугольника; результат делится на ведущий элемент.

Выполним пересчет симплекс-таблицы, приведенной в табл. 2.2. В столбце «Базис» X_5 заменяется на X_2 . Все элементы ведущей строки делятся на ведущий элемент, равный 20. Ведущий столбец (X_2) заполняется нулями. Все остальные элементы пересчитываются по «правилу прямоугольника». Например, коэффициент на пересечении E -строки и столбца X_1 пересчитывается следующим образом: $(20 \cdot (-100) - (-300) \cdot 5) / 20 = -25$. Коэффициент на пересечении строки X_3 и столбца X_5 пересчитывается следующим образом: $(20 \cdot 0 - 5 \cdot 1) / 20 = -0,25$. Расчет этих элементов иллюстрируется рис. 2.2. Полученная симплекс-таблица приведена в табл. 2.3.



Базис	X_1	X_2	X_3	X_4	X_5	Решение
E	-100	-300	0	0	0	0
X_3	20	5	1	0	0	200
X_4	10	5	0	1	0	250
X_5	5	20	0	0	1	500

Рис. 2.2. Примеры расчетов по «правилу прямоугольника»

Таблица 6.2.3

Базис	X_1	X_2	X_3	X_4	X_5	Решение
E	-25,00	0,00	0,00	0,00	15,00	7500,00
X_3	18,75	0,00	1,00	0,00	-0,25	75,00
X_4	8,75	0,00	0,00	1,00	-0,25	125,00
X_2	0,25	1,00	0,00	0,00	0,05	25,00

6. Выполняется возврат к шагу 2.

Для рассматриваемого примера на шаге 5 получено следующее решение (табл. 2.3):

$$X_2 = 25; X_3 = 75; X_4 = 125; X_1 = X_5 = 0$$

(так как переменные X_1 и X_5 – небазисные). Это решение соответствует угловой

точке ОДР, обозначенной на рис.2.1 как А ($X_1 = 0$; $X_2 = 25$). Видно, что полученное решение (табл. 2.3) не является оптимальным, так как в E -строке имеется отрицательный элемент (-25). Поэтому алгоритм продолжается. Определяется переменная для включения в базис (шаг 3). Это переменная X_1 , так как только для этой переменной в E -строке содержится отрицательный элемент. Определяется переменная, исключаемая из базиса (шаг 4). Для этого вычисляются симплексные отношения: $75/18,75 = 4$; $125/8,75 = 14,29$; $25/0,25 = 100$. Минимальное симплексное отношение соответствует переменной X_3 ; она исключается из базиса. Таким образом, ведущий столбец – X_1 , ведущая строка – X_3 , ведущий элемент равен 18,75. Симплекс-таблица преобразуется по правилам, приведенным на шаге 5. Полученная симплекс-таблица показана в табл. 2.4.

Таблица 2.4

Базис	X_1	X_2	X_3	X_4	X_5	Решение
E	0,00	0,00	1,33	0,00	14,67	7600,00
X_1	1,00	0,00	0,05	0,00	– 0,01	4,00
X_4	0,00	0,00	– 0,47	1,00	– 0,13	90,00
X_2	0,00	1,00	– 0,01	0,00	0,05	24,00

Выполняется возврат к шагу 2 (проверка оптимальности полученного решения). Решение, полученное в табл. 2.4, оптимально, так как в E -строке нет отрицательных элементов.

По окончании алгоритма в столбце «Решение» находятся оптимальные значения базисных переменных, а также значение целевой функции, соответствующее полученному решению. Оптимальные значения небазисных переменных равны нулю. Таким образом, получено следующее оптимальное решение:

$$X_1 = 4; \quad X_2 = 24; \quad X_3 = 0; \quad X_4 = 90; \quad X_5 = 0; \quad E = 7600.$$

Значения переменных $X_1 = 4$, $X_2 = 24$ показывают, что цех должен выпускать за смену 4 корпуса и 24 задвижки. В этом случае будет получена максимальная прибыль в размере 7600 ден. ед. (значение целевой функции). Остаточные переменные X_3 , X_4 , X_5 также имеют физический смысл. Например, из системы ограничений (2.1) видно, что величина $20X_1 + 5X_2$ представляет собой расход алюминия на выпуск всех изделий, а величина 200 (правая часть ограничения) – имеющийся запас алюминия. Переменная X_3 представляет собой разность этих величин, т.е. неизрасходованный остаток запаса алюминия. Так как $X_3 = 0$, значит, весь запас алюминия (200 кг) расходуется на выпуск изделий. Аналогично можно показать, что переменная X_4 представляет собой неиз-

расходованный остаток, стали, а X_5 – пластмассы. Таким образом, остается неизрасходованным 90 кг стали (расход стали на выпуск всех изделий составит $250 - 90 = 160$ кг). Неизрасходованный остаток пластмассы равен нулю, значит, все 500 кг пластмассы расходуются на выпуск изделий.

Примечания:

1. Смысл остаточных переменных (как и остальных переменных, используемых в математической модели) полностью зависит от постановки задачи.

2. По результатам решения задачи переменные X_1 и X_2 приняли целочисленные значения. Если бы они оказались дробными, то потребовалось бы использовать специальные методы для получения целочисленного решения, так как эти переменные обозначают количество изделий.

3. Решение задач линейного программирования на основе методов искусственного базиса

3.1. Назначение и принцип работы методов искусственного базиса

Методы искусственного базиса предназначены для решения задач линейного программирования, содержащих ограничения различных видов: «больше или равно», «меньше или равно», «равно».

При решении задачи линейного программирования для построения начального базиса необходимо, чтобы в каждом ограничении присутствовала базисная переменная, т.е. переменная, входящая в данное ограничение с коэффициентом, равным единице, и не входящая ни в одно из других ограничений. В ограничениях «меньше или равно» в качестве таких переменных используются остаточные переменные, добавляемые в ограничение при его приведении к стандартной форме (см. подраздел 1.2). Для приведения к стандартной форме ограничений «больше или равно» вводятся избыточные переменные со знаком «минус». В ограничения «равно» не требуется вводить никаких дополнительных переменных, так как такие ограничения уже соответствуют стандартной форме. Поэтому в задачах, содержащих ограничения «больше или равно» или «равно», после приведения к стандартной форме обычно невозможно построить начальный базис, так как базисные переменные имеются не во всех ограничениях.

Для задач, содержащих ограничения «не меньше» или «равно», обычно нельзя использовать в качестве начального допустимого решения (начальной угловой точки ОДР) начало координат, т.е. решение, в котором все исходные переменные математической модели равны нулю: $X_1 = X_2 = \dots = X_n = 0$. Такое решение, как правило, оказывается недопустимым (не соответствует ограничениям).

Методы искусственного базиса применяются во всех случаях, когда базисные переменные имеются не во всех ограничениях задачи, приведенной к стандартной форме. Принцип работы всех методов искусственного базиса следующий. Во все ограничения, не содержащие базисных переменных, вводятся искусственные переменные (по одной в каждое ограничение), используемые для построения начального базиса. После этого выполняется поиск оптимального решения на основе обычных процедур симплекс-метода.

В окончательном (оптимальном) решении задачи все искусственные переменные должны быть равны нулю. Если в оптимальном решении какая-либо из искусственных переменных оказывается ненулевой, это означает, что задача не имеет допустимых решений. Причиной может быть ошибка в математической модели или противоречия в постановке задачи (например, количество изделий, которое требуется выпустить, не может быть выпущено из-за ограничений на ресурсы).

На искусственные переменные, как и на все остальные переменные в задаче, накладывается требование неотрицательности.

Искусственные переменные не имеют никакого физического смысла: их нельзя интерпретировать как количество изделий, запасы ресурсов и т.д. Они требуются только для построения начального базиса.

Основные методы искусственного базиса – двухэтапный метод, рассматриваемый ниже, и метод больших штрафов. Поиск решения на основе этих методов выполняется с использованием симплекс-таблиц.

3.2. Двухэтапный метод

Основные этапы реализации двухэтапного метода (как и других методов искусственного базиса) следующие.

1. Строится искусственный базис. Находится начальное недопустимое решение. Выполняется переход от начального недопустимого решения к некоторому допустимому решению. Этот переход реализуется путем ми-

нимизации (сведения к нулю) искусственной целевой функции, представляющей собой сумму искусственных переменных.

2. Выполняется переход от начального допустимого решения к оптимальному решению.

Реализацию двухэтапного метода рассмотрим на следующем примере.

Пример. Изделия трех видов (А, В, С) вырезаются из стальных листов. Предприятие имеет 150 стальных листов. Каждый лист можно раскроить одним из трех способов. Количество изделий, получаемых из одного листа, и величины отходов для каждого способа раскроя приведены в табл. 6.3.1.

Таблица 6.3.1

Количество изделий	Способы раскроя		
	1	2	3
А	4	5	2
В	1	1	4
С	2	1	1
Отходы, см²	20	25	17

Предприятию необходимо раскроить листы таким образом, чтобы отходы были минимальны. При этом необходимо выпустить не менее 400 изделий А, не менее 250 изделий В и *не более* 300 изделий С (последнее требование связано с тем, что спрос на изделия С ограничен).

Составим математическую модель задачи. Обозначим через X_1 , X_2 и X_3 количество стальных листов, раскраиваемых первым, вторым и третьим способом соответственно. Математическая модель будет иметь следующий вид:

$$\begin{aligned}
 4X_1 + 5X_2 + 2X_3 &\geq 400 \\
 X_1 + X_2 + 4X_3 &\geq 250 \\
 X_1 + X_2 + X_3 &= 150 \\
 2X_1 + X_2 + X_3 &\leq 300 \\
 X_i &\geq 0, i = 1, \dots, 3.
 \end{aligned}
 \tag{6.3.1}$$

$$E = 20X_1 + 25X_2 + 17X_3 \rightarrow \min. \tag{6.3.2}$$

Здесь первое и второе ограничения устанавливают, что необходимо выпустить не менее 400 изделий А и не менее 250 изделий В. Третье ограничение указывает, что общее количество листов, раскроенных всеми тремя способами, должно составлять 150 (т.е. необходимо раскроить все листы). Четвертое огра-

значение указывает, что количество изделий С не должно превышать 300. Целевая функция представляет собой общее количество отходов.

Приведем задачу к стандартной форме, как показано в подразделе 6.1.2. Математическая модель задачи в стандартной форме имеет следующий вид:

$$\begin{aligned} 4X_1 + 5X_2 + 2X_3 - X_4 &= 400 \\ X_1 + X_2 + 4X_3 - X_5 &= 250 \\ X_1 + X_2 + X_3 &= 150 \\ 2X_1 + X_2 + X_3 + X_6 &= 300 \\ X_i &\geq 0, i = 1, \dots, 6. \end{aligned} \quad (6.3.3)$$

$$-E = -20X_1 - 25X_2 - 17X_3 \rightarrow \max. \quad (6.3.4)$$

В полученной системе уравнений базисная переменная имеется только в четвертом уравнении; это переменная X_6 . Поэтому для решения задачи требуется использовать методы искусственного базиса.

Первый этап (поиск допустимого решения)

1. Во все ограничения, где нет базисных переменных, вводятся искусственные базисные переменные. В данной задаче их требуется ввести в первое, второе и третье ограничения. Добавлять искусственную переменную в четвертое ограничение не требуется, так как оно уже содержит базисную переменную X_6 . Система ограничений с искусственными базисными переменными будет иметь следующий вид:

$$\begin{aligned} 4X_1 + 5X_2 + 2X_3 - X_4 + X_7 &= 400 \\ X_1 + X_2 + 4X_3 - X_5 + X_8 &= 250 \\ X_1 + X_2 + X_3 + X_9 &= 150 \\ 2X_1 + X_2 + X_3 + X_6 &= 300 \\ X_i &\geq 0, i = 1, \dots, 9. \end{aligned} \quad (6.3.5)$$

Таким образом, начальный базис будет состоять из искусственных переменных X_7, X_8, X_9 , а также остаточной переменной X_6 .

2. Составляется искусственная целевая функция – сумма всех искусственных переменных:

$$W = X_7 + X_8 + X_9 \rightarrow \min. \quad (6.3.6)$$

Эта целевая функция подлежит минимизации, так как для определения начального допустимого решения необходимо, чтобы все искусственные переменные приняли нулевые значения.

Примечание. Искусственная целевая функция всегда (в любой задаче) подлежит минимизации.

3. Искусственная целевая функция выражается через небазисные переменные. Для этого сначала требуется выразить искусственные переменные через небазисные:

$$\begin{aligned}X_7 &= 400 - 4X_1 - 5X_2 - 2X_3 + X_4 \\X_8 &= 250 - X_1 - X_2 - 4X_3 + X_5 \\X_9 &= 150 - X_1 - X_2 - X_3.\end{aligned}\tag{6.3.7}$$

Выраженные таким образом искусственные переменные подставляются в искусственную целевую функцию:

$$W = -6X_1 - 7X_2 - 7X_3 + X_4 + X_5 + 800 \rightarrow \min.\tag{6.3.8}$$

4. Для приведения всей задачи к стандартной форме выполняется переход к искусственной целевой функции, подлежащей максимизации. Для этого она умножается на -1 :

$$-W = 6X_1 + 7X_2 + 7X_3 - X_4 - X_5 - 800 \rightarrow \max.\tag{6.3.9}$$

Приведем полную математическую модель задачи, приведенную к стандартной форме:

$$\begin{aligned}4X_1 + 5X_2 + 2X_3 - X_4 + X_7 &= 400 \\X_1 + X_2 + 4X_3 - X_5 + X_8 &= 250 \\X_1 + X_2 + X_3 + X_9 &= 150 \\2X_1 + X_2 + X_3 + X_6 &= 300 \\X_i &\geq 0, i = 1, \dots, 9.\end{aligned}\tag{6.3.10}$$

$$-E = -20X_1 - 25X_2 - 17X_3 \rightarrow \max.\tag{6.3.11}$$

$$-W = 6X_1 + 7X_2 + 7X_3 - X_4 - X_5 - 800 \rightarrow \max.\tag{6.3.12}$$

5. Определяется начальное решение. Все исходные, а также избыточные переменные задачи являются небазисными, т.е. принимаются равными нулю. Искусственные, а также остаточные переменные образуют начальный базис: они равны правым частям ограничений. Для рассматриваемой задачи начальное решение следующее:

$$X_1 = X_2 = X_3 = X_4 = X_5 = 0, X_6 = 300, X_7 = 400, X_8 = 250, X_9 = 150.$$

Это решение является недопустимым: значения переменных

$$X_1 = X_2 = X_3 = 0$$

не удовлетворяют постановке задачи. Начальное значение целевой функции задачи

$$E = 20X_1 + 25X_2 + 17X_3$$

равно нулю. Начальное значение искусственной целевой функции

$$-W = 6X_1 + 7X_2 + 7X_3 - X_4 - X_5 - 800$$

равно -800 .

6. Составляется исходная симплекс-таблица. Она отличается от симплекс-таблицы, используемой для обычного симплекс-метода (см. подраздел 2.4) только тем, что в нее добавляется строка искусственной целевой функции. В этой строке указываются коэффициенты искусственной целевой функции (приведенной к стандартной форме, т.е. подлежащей максимизации) с обратными знаками, как и для обычной целевой функции. Исходная симплекс-таблица для рассматриваемого примера приведена в табл. 6.3.2.

Таблица 6.3.2

Базис	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Решение
$-E$	20	25	17	0	0	0	0	0	0	0
$-W$	-6	-7	-7	1	1	0	0	0	0	-800
X_7	4	5	2	-1	0	0	1	0	0	400
X_8	1	1	4	0	-1	0	0	1	0	250
X_9	1	1	1	0	0	0	0	0	1	150
X_6	2	1	1	0	0	1	0	0	0	300

7. Выполняется переход от начального недопустимого решения, содержащегося в исходной симплекс-таблице, к некоторому допустимому решению. Для этого с помощью обычных процедур симплекс-метода выполняется минимизация искусственной целевой функции W (или, то же самое, максимизация функции $-W$). При этом переменные, включаемые в базис, выбираются по строке искусственной целевой функции. Все остальные действия выполняются точно так же, как в обычном симплекс-методе. В результате минимизации искусственная целевая функция $-W$ должна принять нулевое значение. Все искусственные переменные при этом также становятся равными нулю (исключаются из базиса), так как искусственная целевая функция представляет собой их сумму.

В табл. 6.3.2 в строке искусственной целевой функции содержатся два одинаковых коэффициента, имеющих максимальные по модулю (в этой строке) отрицательные значения, равные -7 . Это коэффициенты при переменных X_2 и X_3 . Для включения в базис можно выбрать любую из этих переменных. Выберем X_2 . Столбец переменной X_2 становится ведущим.

Для определения переменной, исключаемой из базиса, найдем симплексные отношения: $400/5 = 80$; $250/1 = 250$; $150/1 = 150$; $300/1 = 300$. Минимальное

симплексное отношение получено в строке переменной X_7 ; значит, эта переменная исключается из базиса. Строка переменной X_7 становится ведущей.

Выполняются преобразования таблицы по правилам симплекс-метода. Ведущая строка (X_7) делится на ведущий элемент (в данном примере он равен 5). Ведущий столбец (X_2) заполняется нулями. Все остальные элементы таблицы (включая строки основной и искусственной целевых функций, а также столбец решений) пересчитываются по «правилу прямоугольника». Полученная симплекс-таблица приведена в табл. 6.3.3.

Таблица 6.3.3

Базис	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Решение
$-E$	0,00	0,00	7,00	5,00	0,00	0,00	-5,00	0,00	0,00	-2000,00
$-W$	-0,4	0,00	-4,20	-0,40	1,00	0,00	1,40	0,00	0,00	-240,00
X_2	0,80	1,00	0,40	-0,20	0,00	0,00	0,20	0,00	0,00	80,00
X_8	0,20	0,00	3,60	0,20	-1,00	0,00	-0,20	1,00	0,00	170,00
X_9	0,20	0,00	0,60	0,20	0,00	0,00	-0,20	0,00	1,00	70,00
X_6	1,20	0,00	0,60	0,20	0,00	1,00	-0,20	0,00	0,00	220,00

Полученное решение еще не является допустимым: в базисе есть искусственные переменные, и искусственная целевая функция не равна нулю.

Примечание. В том, что решение недопустимо, легко убедиться, подставив значения исходных переменных задачи ($X_1 = 0$, $X_2 = 80$, $X_3 = 0$) в математическую модель (6.3.1).

Минимизация искусственной целевой функции продолжается. Для включения в базис выбирается переменная X_3 , так как ей соответствует максимальный по модулю отрицательный коэффициент в строке искусственной целевой функции. Для выбора переменной, исключаемой из базиса, найдем симплексные отношения:

$$80/0,4 = 200; 170/3,6 = 47,2; 70/0,6 = 116,67; 220/0,6 = 366,67.$$

Минимальное симплексное отношение получено в строке переменной X_8 ; она исключается из базиса. Выполняются преобразования по правилам симплекс-метода. Полученная симплекс-таблица приведена в табл. 6.3.4.

Таблица 6.3.4

Базис	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Решение
$-E$	-0,39	0,00	0,00	4,61	1,94	0,00	-4,61	-1,94	0,00	-2330,56
$-W$	-0,17	0,00	0,00	-0,17	-0,17	0,00	1,17	1,17	0,00	-41,67
X_2	0,78	1,00	0,00	-0,22	0,11	0,00	0,22	-0,11	0,00	61,11
X_3	0,06	0,00	1,00	0,06	-0,28	0,00	-0,06	0,28	0,00	47,22

X_9	0,17	0,00	0,00	0,17	0,17	0,00	−0,17	−0,17	1,00	41,67
X_6	1,17	0,00	0,00	0,17	0,17	1,00	−0,17	−0,17	0,00	191,67

В базисе еще остаются искусственные переменные, поэтому поиск допустимого решения продолжается. В базис можно включить одну из переменных X_1 , X_4 или X_5 (коэффициенты в строке искусственной целевой функции при этих переменных имеют одинаковые отрицательные значения). Выберем переменную X_1 . Найдем симплексные отношения:

$$61,11/0,78 = 78,35; 47,22/0,06 = 787;$$

$$41,67/0,17 = 245,12; 191,67/1,17 = 163,82.$$

Из базиса исключается переменная X_2 . Новая симплекс-таблица приведена в табл. 6.3.5.

Таблица 6.3.5

Базис	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Решение
$-E$	0,00	0,50	0,00	4,50	2,00	0,00	−4,5	−2,00	0,00	−2300,00
$-W$	0,00	0,21	0,00	−0,21	−0,14	0,00	1,21	1,14	0,00	−28,57
X_1	1,00	1,29	0,00	−0,29	0,14	0,00	0,29	−0,14	0,00	78,57
X_3	0,00	−0,07	1,00	0,07	−0,29	0,00	−0,07	0,29	0,00	42,86
X_9	0,00	0,21	0,00	0,21	0,14	0,00	−0,21	−0,14	1,00	28,57
X_6	0,00	−1,50	0,00	0,50	0,00	1,00	−0,50	0,00	0,00	100,00

В базисе есть искусственная переменная, поэтому поиск допустимого решения продолжается. В базис включается переменная X_4 , имеющая максимальный по модулю отрицательный коэффициент в строке искусственной целевой функции. Чтобы определить переменную, исключаемую из базиса, найдем симплексные отношения:

$$42,86/0,07 = 612,29; 28,57/0,21 = 136,05; 100/0,5 = 200,$$

для строки переменной X_1 симплексное отношение не определяется, так как соответствующий коэффициент ведущего столбца имеет отрицательное значение. Из базиса исключается переменная X_9 . Новая симплекс-таблица приведена в табл. 6.3.6.

Таблица 6.3.6

Базис	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Решение
$-E$	0,00	5,00	0,00	0,00	−1,00	0,00	0,00	1,00	−21,0	−2900,00
$-W$	0,00	0,00	0,00	0,00	0,00	0,00	1,00	1,00	1,00	0,00
X_1	1,00	1,00	0,00	0,00	0,33	0,00	0,00	−0,33	1,33	116,67
X_3	0,00	0,00	1,00	0,00	−0,33	0,00	0,00	0,33	−0,33	33,33
X_4	0,00	−1,00	0,00	1,00	0,67	0,00	−1,00	−0,67	4,67	133,33

X_6	0,00	-1,00	0,00	0,00	-0,33	1,00	0,00	0,33	-2,33	33,33
-------	------	-------	------	------	-------	------	------	------	-------	-------

Как видно из табл. 6.3.6, искусственная целевая функция равна нулю, и все искусственные переменные исключены из базиса. Получено допустимое решение. Таким образом, первый этап двухэтапного метода завершен. Искусственная целевая функция и искусственные переменные исключаются из симплекс-таблицы (табл. 6.3.7).

Примечание. В том, что получено допустимое решение, легко убедиться, подставив значения исходных переменных задачи ($X_1=116,67$, $X_2=0$, $X_3=33,33$) в математическую модель (6.3.1).

Таблица 6.3.7

Базис	X_1	X_2	X_3	X_4	X_5	X_6	Решение
$-E$	0,00	5,00	0,00	0,00	-1,00	0,00	-2900,00
X_1	1,00	1,00	0,00	0,00	0,33	0,00	116,67
X_3	0,00	0,00	1,00	0,00	-0,33	0,00	33,33
X_4	0,00	-1,00	0,00	1,00	0,67	0,00	133,33
X_6	0,00	-1,00	0,00	0,00	-0,33	1,00	33,33

Второй этап (поиск оптимального решения)

Решение, полученное по результатам первого этапа (табл. 6.3.7), не является оптимальным: в строке целевой функции имеется отрицательный элемент. Поиск оптимального решения выполняется по обычным правилам симплекс-метода. В базис включается переменная X_5 . Для определения переменной, исключаемой из базиса, найдем симплексные отношения: $116,67/0,33 = 350$, $133,33/0,67 = 200$. Из базиса исключается переменная X_4 . После преобразований по правилам симплекс-метода будет получена новая симплекс-таблица (табл. 6.3.8).

Таблица 6.3.8

Базис	X_1	X_2	X_3	X_4	X_5	X_6	Решение
$-E$	0,00	3,50	0,00	1,50	0,00	0,00	-2700,00
X_1	1,00	1,50	0,00	-0,50	0,00	0,00	50,00
X_3	0,00	-0,50	1,00	0,50	0,00	0,00	100,00
X_5	0,00	-1,50	0,00	1,50	1,00	0,00	200,00

X_6	0,00	-1,50	0,00	0,50	0,00	1,00	100,00
-------	------	-------	------	------	------	------	--------

Получено оптимальное решение (признак его оптимальности – отсутствие отрицательных элементов в строке целевой функции). Основные переменные задачи приняли следующие значения: $X_1 = 50$, $X_2 = 0$ (эта переменная – небазисная), $X_3 = 100$. Это означает, что необходимо раскроить 50 стальных листов первым способом, 100 листов – третьим способом. Второй способ раскроя использовать не следует. Значение целевой функции $E = 2700$ показывает, что отходы при таком раскрое составят 2700 см^2 .

Избыточная переменная $X_4 = 0$ означает, что изделий А будет выпущено не больше минимально необходимого количества, т.е. ровно 400. Избыточная переменная $X_5 = 200$ означает, что количество выпущенных изделий В будет на 200 больше, чем минимально необходимое количество, т.е. $250 + 200 = 450$. Остаточная переменная $X_6 = 100$ означает, что количество выпущенных изделий С будет на 100 изделий меньше, чем максимально допустимое количество, т.е. $300 - 100 = 200$. Таким образом, будет выпущено 400 изделий А, 450 изделий В и 200 изделий С.

4. Минимизация нелинейных функционалов без ограничений

Задача безусловной оптимизации состоит в нахождении минимума или максимума функции в отсутствие каких-либо ограничений. Несмотря на то что большинство практических задач оптимизации содержит ограничения, изучение методов безусловной оптимизации важно с нескольких точек зрения. Многие алгоритмы решения задачи с ограничениями предполагают сведение ее к последовательности задач безусловной оптимизации. Другой класс методов основан на поиске подходящего направления и последующей минимизации вдоль этого направления. Обоснование методов безусловной оптимизации может быть естественным образом распространено на обоснование процедур решения задач с ограничениями.

Задача многомерной безусловной оптимизации формулируется в виде:

$$\min f(x), x \in X,$$

где $x = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ – точка в n -мерном пространстве $X = R^n$ – n -мерное вещественное линейное пространство, то есть целевая функция $f(x) = f(x^{(1)}, \dots, x^{(n)})$ – функция n аргументов.

Численные методы отыскания минимума, как правило, заключаются в построении последовательности точек $\{x_k\}$, удовлетворяющих условию $f(x_0) > f(x_1) > \dots > f(x_n) > \dots$.

Методы построения таких последовательностей называются *методами спуска*. В этих методах точки последовательности $\{x_k\}$ вычисляются по формуле:

$$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0, 1, 2, \dots,$$

где p_k – направление спуска, α_k – длина шага в этом направлении.

Различные методы спуска отличаются друг от друга способами выбора направления спуска p_k и длины шага α_k вдоль этого направления. Алгоритмы безусловной минимизации (минимизации без ограничений) принято делить на классы в зависимости от максимального порядка производных минимизируемой функции, вычисление которых предполагается. Так, методы, использующие только значения самой целевой функции относят к *методам нулевого порядка* (иногда их называют также *методами прямого поиска*); если требуется вычисление первых производных минимизируемой функции, то мы имеем дело с *методами первого порядка*; если же дополнительно используются вторые производные, то это *методы второго порядка* и т. д.

4.1. Многомерный поиск без использования производных (прямые методы минимизации)

Рассмотрим методы решения минимизации функции нескольких переменных f , которые опираются только на вычисление значений функции $f(x)$, не используют вычисление производных, т.е. прямые методы минимизации. Важно отметить, что для применения этих методов не требуется не только дифференцируемости целевой функции, но даже аналитического задания. Нужно лишь иметь возможность вычислять или измерять значения f в произвольных точках.

4.1.1. Метод полного перебора (метод сеток)

Многомерные задачи, естественно, являются более сложными и трудоемкими, чем одномерные, причем обычно трудности при их решении возрастают при увеличении размерности. Для того чтобы лучше почувствовать это, возьмем самый простой по своей идее приближенный метод поиска наименьшего значения функции. Покроем рассматриваемую область сеткой G с шагом h (рис. 4.1) определим значения функции в ее узлах. Сравнивая полученные числа между собой, найдем среди них наименьшее и примем его приближенно за наименьшее значение функции для всей области.

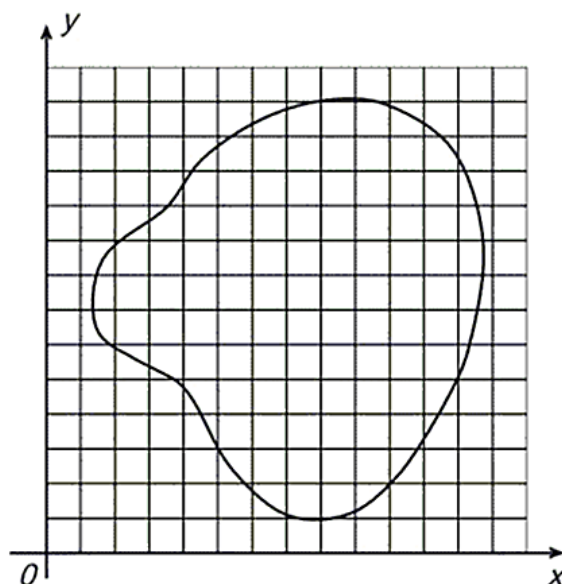


Рис. 6.4.1

Как известно, данный метод используется для решения одномерных задач. Иногда он применяется также для решения двумерных, реже трехмерных задач. Однако для задач большей размерности он практически непригоден из-за слишком большого времени, необходимого для проведения расчетов. Действительно, предположим, что целевая функция зависит от пяти переменных, а область определения G является пятимерным кубом, каждую сторону которого при построении сетки мы делим на 40 частей. Тогда общее число узлов сетки будет равно $41^5 \approx 10^8$. Пусть вычисление значения функции в одной точке требует 1000 арифметических операций (это немного для функции пяти переменных). В таком случае общее число операций составит 10^{11} . Если в нашем распоряжении имеется ЭВМ с быстродействием 1 млн. операций в секунду, то для решения задачи с помощью данного метода потребуется 10^5 секунд, что превышает сутки непрерывной работы. Добавление еще одной независимой переменной увеличит это время в 40 раз. Проведенная оценка показывает, что для больших задач оптимизации метод сплошного перебора непригоден. Иногда сплошной перебор заменяют случайным поиском. В этом случае точки сетки просматриваются не подряд, а в случайном порядке. В результате поиск наименьшего значения целевой функции существенно ускоряется, но теряет свою надежность.

4.1.2. Метод Хука – Дживса

Этот метод был разработан в 1961 году, но до сих пор является весьма эффективным и оригинальным. Поиск состоит из последовательности шагов исследующего поиска вокруг базисной точки, за которой в случае успеха следует поиск по образцу.

Описание этой процедуры представлено ниже:

А. Выбрать начальную базисную точку b_1 и шаг длиной h_j для каждой переменной x_j , $j = 1, 2, \dots, n$. В приведенном ниже алгоритме для каждой переменной используется шаг h , однако указанная выше модификация тоже может оказаться полезной.

Б. Вычислить $f(x)$ в базисной точке b_1 с целью получения сведений о локальном поведении функции $f(x)$. Эти сведения будут использоваться для нахождения подходящего направления поиска по образцу, с помощью которого можно надеяться достичь большего убывания значения функции. Функция $f(x)$ в базисной точке b_1 находится следующим образом:

1. Вычисляется значение функции $f(b_1)$ в базисной точке b_1 .
2. Каждая переменная по очереди изменяется прибавлением длины шага.

Таким образом, мы вычисляем значение функции $f(b_1 + h_1 e_1)$, где e_1 - единичный вектор в направлении оси x_1 . Если это приводит к уменьшению значения функции, то b_1 заменяется на $b_1 + h_1 e_1$. В противном случае вычисляется значение функции $f(b_1 - h_1 e_1)$, и если ее значение уменьшилось, то b_1 заменяем на $b_1 - h_1 e_1$. Если ни один из сделанных шагов не приводит к уменьшению значения функции, то точка b_1 остается неизменной и рассматриваются изменения в направлении оси x_2 , т.е. находится значение функции $f(b_1 + h_2 e_2)$ и т.д. Когда будут рассмотрены все n переменные, мы будем иметь новую базисную точку b_2 .

3. Если $b_2 = b_1$, т.е. уменьшение функции не было достигнуто, то исследование повторяется вокруг той же базисной точки b_1 , но с уменьшенной длиной шага. На практике удовлетворительным является уменьшение шага (шагов) в десять раз от начальной длины.

4. Если $b_2 \neq b_1$, то производится поиск по образцу.

В. При поиске по образцу используется информация, полученная в процессе исследования, и минимизация функции завершается поиском в направлении, заданном образцом. Эта процедура производится следующим образом:

1. Разумно двигаться из базисной точки b_2 в направлении $b_2 - b_1$, поскольку поиск в этом направлении уже привел к уменьшению значения функции. Поэтому вычислим функцию в точке образца

$$P_1 = b_1 + 2(b_2 - b_1) \quad (6.4.1)$$

В общем случае

$$P_j = b_j + 2(b_{j+1} - b_j) \quad (6.4.2)$$

2. Затем исследование следует продолжать вокруг точки P_1 (P_j).

3. Если наименьшее значение на шаге В,2 меньше значения в базисной точке b_2 (в общем случае b_{j+1}), то получают новую базисную точку b_3 (b_{j+2}), после чего следует повторить шаг В,1. В противном случае не производить поиск по образцу из точки b_2 (b_{j+1}) а продолжить исследования в точке b_2 (b_{j+1}).

Г. Завершить этот процесс, когда длина шага (длины шагов) будет уменьшена до заданного малого значения.

Ниже приведена блок-схема данного метода.

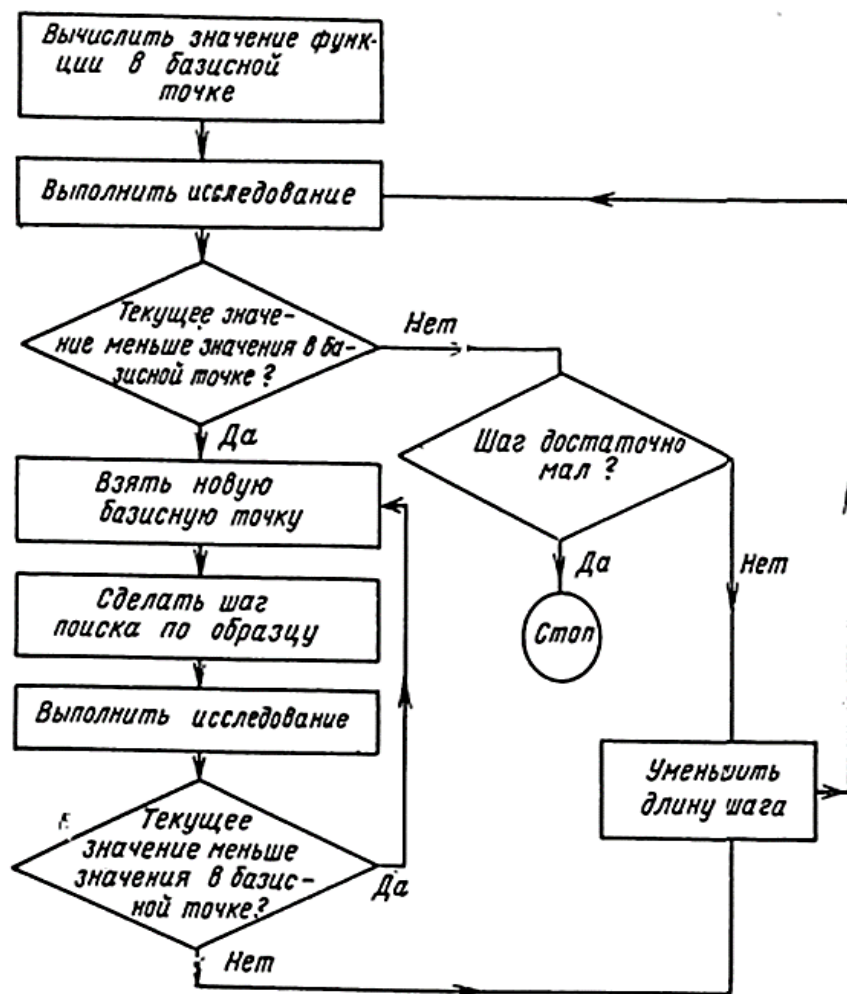


Рис. 6.4.2

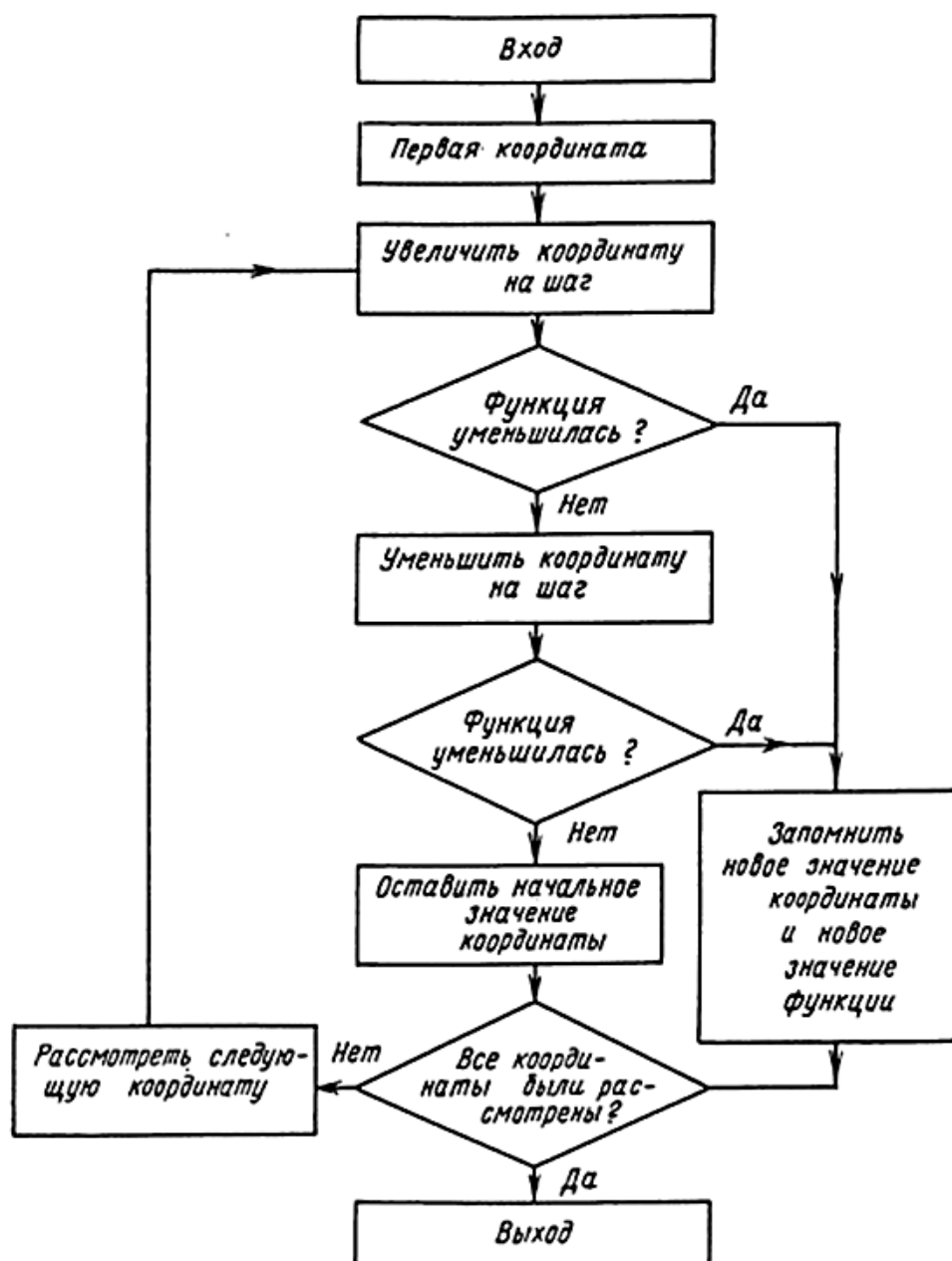


Рис. 6.4.3

4.1.3. Метод Нелдера - Мида

Метод Нелдера - Мида (называется также поиском по деформируемому многограннику) является развитием симплексного метода Спендли, Хекста и Химсворта. Множество $(n+1)$ -й равноудаленной точки в n -мерном пространстве называется регулярным симплексом. Эта конфигурация рассматривается в методе Спендли, Хекста и Химсворта. Следовательно, в двумерном пространстве симплексом является равносторонний треугольник, а в трехмерном пространстве — правильный тетраэдр. Идея метода состоит в сравнении значений функции в $(n + 1)$ вершинах симплекса и перемещении симплекса в направлении оптимальной точки с помощью итерационной процедуры. В симплексном методе,

предложенном первоначально, регулярный симплекс использовался на каждом этапе. Нелдер и Мид предложили несколько модификаций этого метода, допускающих, чтобы симплексы были неправильными. В результате получился очень надежный метод прямого поиска, являющийся одним из самых эффективных, если $n \leq 6$.

4.1.4. Анализ методов прямого поиска

Метод	Достоинства	Недостатки
Метод поиска по симплексу	<ul style="list-style-type: none"> • простота; • малое количество заранее установленных параметров; • алгоритм эффективен и тогда, когда ошибки в определении значения целевой функции достаточно велики, так как в нём используется наибольшее значение целевой функции, а не наименьшее 	<ul style="list-style-type: none"> • возникают трудности связанные с масштабированием задачи (в реальных задачах разные переменные часто не сопоставимы между собой по значениям); • алгоритм работает медленно (не используется информация предыдущих итераций); • не существует простого способа изменения размеров симплекса без пересчёта всех значений целевой функции
Метод Нелдера-Мида	<ul style="list-style-type: none"> • частично устраняет недостатки метода поиска по симплексу, используя информацию с предыдущих итераций • 	<ul style="list-style-type: none"> • метод работает эффективно при $n \leq 6$
Метод поиска Хука-Дживса	<ul style="list-style-type: none"> • простая стратегия поиска, • вычисление только значений функции, • небольшой объём требуемой памяти 	<ul style="list-style-type: none"> • алгоритм основан на циклическом движении по координатам. Это может привести к вырождению алгоритма в бесконечную последовательность исследующих поисков без поиска по образцу

4.2. Методы первого порядка многомерного поиска

К методам первого порядка относятся алгоритмы, в которых в процессе поиска кроме информации о самой функции используется информация о производных первого порядка. К группе таких методов относятся различные градиентные методы.

4.2.1. Алгоритм наискорейшего спуска

Градиент функции в любой точке показывает направление наибольшего локального увеличения $f(\bar{x})$. Поэтому при поиске минимума $f(\bar{x})$ следует двигаться в направлении противоположном направлению градиента $\nabla f(\bar{x})$ в данной точке, то есть в направлении *наискорейшего спуска*.

Итерационная формула процесса наискорейшего спуска имеет вид:

$$\bar{x}^{k+1} = \bar{x}^k - \lambda^k \cdot \nabla f(\bar{x}^k)$$

или

$$\bar{x}^{k+1} = \bar{x}^k - \lambda^k \cdot \frac{\nabla f(\bar{x}^k)}{\|\nabla f(\bar{x}^k)\|} = \bar{x}^k + \lambda^k \cdot \bar{S}^k.$$

Очевидно, что в зависимости от выбора параметра λ траектории спуска будут существенно различаться (см. рис. 4.4). При большом значении λ траектория спуска будет представлять собой колебательный процесс, а при слишком больших λ процесс может расходиться. При выборе малых λ траектория спуска будет плавной, но и процесс будет сходиться очень медленно.

Обычно λ выбирают из условия

$$\lambda^k = \arg \min_{\lambda} f(\bar{x}^k + \lambda \cdot \bar{S}^k),$$

решая одномерную задачу минимизации с использованием некоторого метода. В этом случае получаем алгоритм наискорейшего спуска.

Если λ определяется в результате одномерной минимизации, то градиент в точке очередного приближения будет ортогонален направлению предыдущего спуска $\nabla f(\bar{x}^{k+1}) \perp \bar{S}^k$.

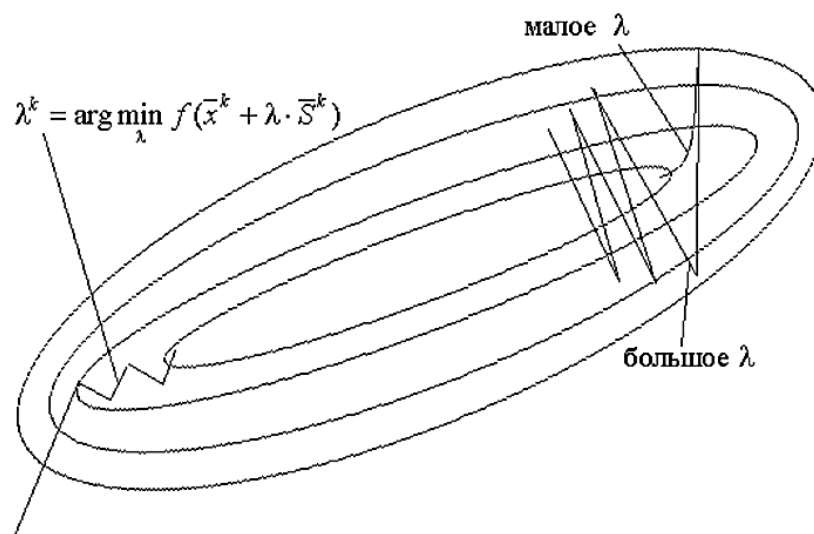


Рис. 6.4.4. Траектории спуска в зависимости от величины шага

Вообще говоря, процедура наискорейшего спуска может закончиться в стационарной точке любого типа, в которой $\nabla f(\bar{x}) = \bar{0}$. Поэтому следует

проверять, не завершился ли алгоритм в седловой точке.

Эффективность алгоритма зависит от вида минимизируемой функции. Алгоритм наискорейшего спуска сойдется за одну итерацию при любом начальном приближении для функции $f(\bar{x}) = x_1^2 + x_2^2$ (см. рис.4.5). Но сходимость будет очень медленной, например, в случае функции вида $f(\bar{x}) = x_1^2 + 100x_2^2$. В тех ситуациях, когда линии уровня минимизируемой функции представляет собой прямолинейный или, хуже того, криволинейный "овраг" эффективность алгоритма оказывается очень низкой.

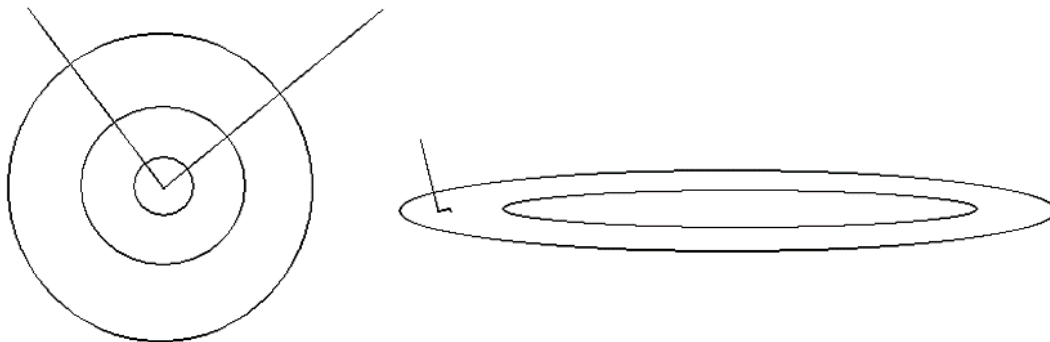


Рис. 6.4.5. Траектории спуска в зависимости от вида функций

Очевидно, что хорошие результаты может давать предварительное масштабирование функций.

Процесс наискорейшего спуска обычно быстро сходится вдали от точки экстремума и медленно в районе экстремума. Поэтому метод наискорейшего спуска нередко используют в комбинации с другими алгоритмами.

4.2.2. Многопараметрический поиск

Милем и Кентреллом предложен метод поиска, основанный на использовании двух подбираемых параметров для минимизации $f(\bar{x})$ в каждом направлении поиска. В этом алгоритме последовательность действий определяется формулой:

$$\bar{x}^{k+1} = \bar{x}^k - \lambda_0^k \cdot \nabla f(\bar{x}^k) + \lambda_1^k \cdot \Delta \bar{x}^{k-1}, \quad (6.4.3)$$

где $\Delta \bar{x}^{k-1} = \bar{x}^k - \bar{x}^{k-1}$.

На каждом шаге решается задача минимизации по двум параметрам:

$$\min_{\lambda_0, \lambda_1} f(\bar{x}^k - \lambda_0^k \cdot \nabla f(\bar{x}^k) + \lambda_1^k \cdot \Delta \bar{x}^{k-1}).$$

После чего находится очередное приближение по формуле (6.4.3). При этом можно показать, что

$$\nabla^T f(\bar{x}^k) \cdot \nabla f(\bar{x}^{k+1}) = 0,$$

$$\nabla^T f(\bar{x}^{k+1}) \cdot \Delta \bar{x}^{k+1} = 0,$$

$$\nabla^T f(\bar{x}^{k+1}) \cdot \Delta \bar{x}^k = 0.$$

На первом шаге $\Delta \bar{x}^{k-1} = 0$, а \bar{x}^0 должно быть задано. На k -м шаге:

1. Вычисляется $\bar{x}^k, \nabla f(\bar{x}^k)$ и $\Delta \bar{x}^{k-1} = \bar{x}^k - \bar{x}^{k-1}$.

2. Пользуясь одним из эффективных методов, например, методом Ньютона находят s с требуемой точностью λ_0^k, λ_1^k .

3. По соотношению (6.4.3) вычисляют \bar{x}^{k+1} и переходят к пункту 1.

4. Каждый $(n+1)$ -й шаг начинается с $\Delta \bar{x}^{k-1} = 0$.

5. Процесс заканчивается, когда $|\nabla f(\bar{x}^k)| < \varepsilon$.

На квадратичных функциях алгоритм по эффективности близок к методу сопряженных градиентов.

5. Методы штрафных функций

Естественно, что наиболее распространенные задачи на практике - это оптимизационные задачи при наличии ограничений, то есть задачи поиска оптимального решения, удовлетворяющего некоторой системе ограничений. Существование эффективных алгоритмов решения безусловных задач оптимизации всегда толкает на попытку использования этих методов для решения условных задач после соответствующего преобразования условной задачи к некоторой эквивалентной безусловной задаче.

Пусть необходимо решить задачу

$$\min \{f(\bar{x}) \mid h_j(\bar{x}) = 0, j = \overline{1, m}; g_j(\bar{x}) \leq 0, j = \overline{m+1, k}\}, \quad (6.5.1)$$

в которой целевая функция и функции системы ограничений представляют собой выпуклые функции (желательно).

Основная идея метода штрафных функций заключается в следующем. Строят такую вспомогательную функцию

$$Q(\bar{x}, \bar{r}) = f(\bar{x}) + \sum_{j=1}^m r_j \cdot H[h_j(\bar{x})] + \sum_{j=m+1}^k r_j \cdot G[g_j(\bar{x})], \quad (6.5.2)$$

чтобы приближенное решение задачи (1) находилось в результате решения *последовательности* задач безусловной минимизации функций (5.2)

$$\min Q(\bar{x}, \bar{r}). \quad (6.5.3)$$

В методе (внешних) **штрафных функций** функции $H()$ и $G()$ выбираются таким образом, чтобы они становились отличными от нуля (*положительными*) при нарушении соответствующего ограничения. А так как мы минимизируем (6.5.2), то движение в сторону нарушения ограничения становится невыгодным. В данном методе функции $H()$ и $G()$ внутри допустимой области должны быть равны нулю. Например, для ограничений неравенств:

$$G_j[g_j(\bar{x})] \rightarrow 0 \quad \text{при} \quad g_j(\bar{x}) \rightarrow 0^+.$$

Приближенное решение задачи (5.1) получается в результате решения последовательности задач (6.5.3) при $r_j \rightarrow \infty, j = \overline{1, k}$. Соответствующие методы называют еще *методами внешней точки*.

В методе **барьерных функций** функции $H(\cdot)$ и G в допустимой области выбираются отличными от нуля, причем такими, чтобы при приближении к границе допустимой области (изнутри) они возрастали, препятствуя выходу при поиске за границу области. В этом случае эти функции должны принимать малые (*положительные или отрицательные*) значения внутри допустимой области и большие *положительные* вблизи границы (внутри области). Например, для ограничений неравенств:

$$G_j[g_j(\bar{x})] \rightarrow 0 \quad \text{при} \quad g_j(\bar{x}) \rightarrow 0^-.$$

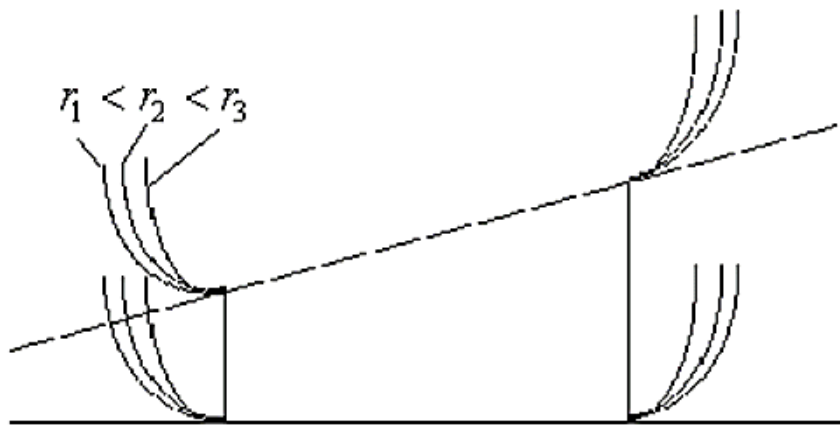


Рис. 6.5.1. Формирование штрафа при нарушении ограничений

Такие методы называют ещё *методами внутренней точки*. В алгоритмах, использующих функции штрафа данного типа (барьерные функции), требуют, чтобы в процессе поиска точка \bar{x} всегда оставалась внутренней точкой допустимой области. Приближенное решение задачи (5.1) получается в результате решения последовательности задач вида (5.3) при $r_j \rightarrow 0, j = \overline{1, k}$.

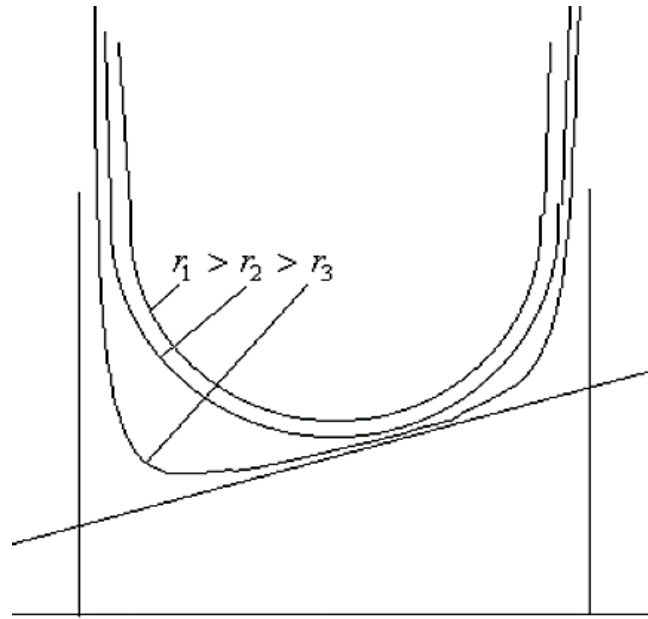


Рис. 6.5.2. Формирование барьеров вблизи границы

При выборе функций штрафов для ограничений равенств обычно требуют, чтобы

$$H_j[h_j(\bar{x})] \rightarrow 0 \quad \text{при} \quad h_j(\bar{x}) \rightarrow 0.$$

Это могут быть, например, функции следующего вида:

- 1) $H_j[h_j(\bar{x})] = |h_j(\bar{x})|$,
- 2) $H_j[h_j(\bar{x})] = |h_j(\bar{x})|^2$,
- 3) $H_j[h_j(\bar{x})] = |h_j(\bar{x})|^\alpha$, при четном α .

Для ограничений неравенств функции штрафа подбирают таким образом, чтобы

$$G_j[g_j(\bar{x})] = 0 \quad \text{при} \quad g_j(\bar{x}) \leq 0;$$

$$G_j[g_j(\bar{x})] > 0 \quad \text{при} \quad g_j(\bar{x}) > 0.$$

Этому требованию отвечают, например, функции вида:

- 1) $G_j[g_j(\bar{x})] = \frac{1}{2} \{g_j(\bar{x}) + |g_j(\bar{x})|\}$,
- 2) $G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{g_j(\bar{x}) + |g_j(\bar{x})|\} \right]^2$,
- 3) $G_j[g_j(\bar{x})] = \left[\frac{1}{2} \{g_j(\bar{x}) + |g_j(\bar{x})|\} \right]^\alpha$ при четной степени α .

В качестве барьерных функций для ограничений неравенств могут служить, например, функции вида:

- 1) $G_j[g_j(\bar{x})] = -\frac{1}{g_j(\bar{x})}$,

$$2) \quad G_j[g_j(\bar{x})] = -\ln[-g_j(\bar{x})].$$

Последовательность действий при реализации методов штрафных или барьерных функций выглядит следующим образом:

1. На основании задачи (6.5.1) строим функцию (6.5.2). Выбираем начальное приближение \bar{x} и начальные значения коэффициентов штрафа r_j .

2. Решаем задачу (6.5.3).

3. Если полученное решение не удовлетворяет системе ограничений, то в случае использования метода штрафных функций увеличиваем значения коэффициентов штрафа r_j и снова решаем задачу (6.5.3). В случае метода барьерных функций, чтобы можно было получить решение на границе, значения коэффициентов r_j уменьшаются.

4. Процесс прекращается, если найденное решение удовлетворяет системе ограничений с определенной точностью.

Лабораторная работа № 7

МНОГОМЕРНАЯ ДИСКРЕТНАЯ ОПТИМИЗАЦИЯ

Перед выполнением лабораторной работы рекомендуем:

1. Изучить теоретические сведения в описании данной лабораторной работы.
2. Прочитать соответствующие разделы книги:

Бахвалов, Н.С. Численные методы : учебник / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. — 9-е изд. — Москва : Лаборатория знаний, 2020. — 636 с. — ISBN 978-5-00101-836-0. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: <https://e.lanbook.com/book/126099> (дата обращения: 31.10.2019). — Режим доступа: для авториз. пользователей.

Смородинский С.С., Батин Н.В. Оптимизация решений на основе методов и моделей математического программирования: Уч. Пособие по курсу «Системный анализ и исследование операций». – Минск: БГУИР, 2003. – 136 с.

Цель работы:

1. Изучить и научиться использовать на практике методы многомерной дискретной оптимизации.
2. Написать программу, реализующую алгоритм многомерной дискретной оптимизации на основе метода ветвей и границ, или муравьиного алгоритма, или жадного алгоритма, или метода перебора.

Содержание отчета:

1. Постановка задачи, описание математической модели задачи, ее приведение к стандартной форме, ход решения задачи с использованием одного из методов многомерной дискретной оптимизации, результаты решения задачи.
2. Описание компьютерной программы. Объяснение значений всех параметров.
3. Для контроля полученного ответа, можно привести результаты решения задач с использованием программных средств, например, рабочий лист Excel с результатами решения задачи на основе базовой аналитической модели.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1. Решение задач оптимизации на основе методов линейного целочисленного программирования

1.1. Назначение методов целочисленного программирования

Методы целочисленного программирования предназначены для решения задач, в которых некоторые (или все) переменные по своему физическому смыслу должны принимать только целочисленные значения.

Если требование целочисленности накладывается на *все* переменные, имеющиеся в математической модели задачи (*включая все остаточные и избыточные переменные*, вводимые в модель при ее приведении к стандартной форме), то такая задача называется *полностью целочисленной*. Если требование целочисленности накладывается лишь на *некоторые переменные*, то задача называется *частично целочисленной*.

Принцип работы всех методов линейного целочисленного программирования следующий. Сначала задача решается симплекс-методом без учета требований целочисленности. Если все переменные, которые по своему смыслу должны быть целочисленными, принимают в оптимальном решении целочисленные значения, то решение задачи на этом завершается. Если хотя бы одна из этих переменных принимает дробное значение, то в задачу вводятся дополнительные ограничения, исключающие полученное оптимальное (но нецелочисленное) решение из области допустимых решений. Новые задачи (одна или несколько), включающие дополнительные ограничения, также решаются симплекс-методом. Процесс продолжается до получения оптимального целочисленного решения.

Основные методы решения задач целочисленного программирования – метод ветвей и границ и метод Гомори.

1.2. Метод ветвей и границ

Принцип работы метода ветвей и границ основан на использовании списка решаемых задач (задач-кандидатов). Если в оптимальном решении задачи какая-либо из переменных, которая по своему смыслу должна быть целочисленной, приняла дробное значение, то в список решаемых задач включаются новые задачи с дополнительными ограничениями, исключающими полученное оптимальное (но нецелочисленное) решение из области допустимых решений, т.е. делающими это решение недопустимым. Для каждой задачи определяется

также ее оценка – граница значения целевой функции (способ определения оценки будет показан ниже). Под оценкой задачи понимается величина, о которой точно известно, что оптимальное значение целевой функции не будет лучше этой величины. Из списка решаемых задач выбирается очередная задача (имеется несколько вариантов метода ветвей и границ, где правила выбора такой задачи могут быть разными). Выбранная задача решается симплекс-методом. Если решение выбранной задачи оказывается нецелочисленным, то в нее также вводятся дополнительные ограничения, и полученные задачи включаются в список решаемых задач. Если решение оказывается целочисленным, то все задачи-кандидаты, оценка которых хуже, чем полученное целочисленное решение, исключаются из списка решаемых задач, так как поиск их решения не имеет смысла (целевая функция этих задач в любом случае будет иметь худшее значение, чем в полученном целочисленном решении). Процесс продолжается, пока не будут решены все задачи, входящие в список решаемых задач. Лучшее из полученных целочисленных решений и является оптимальным решением.

В ходе реализации метода ветвей и границ может быть получено несколько целочисленных решений задачи. Будем называть лучшее из них «текущим наилучшим решением» (ТНР). Значение целевой функции, соответствующее этому решению, будем обозначать как $E_{\text{ТНР}}$. В начале решения задачи ТНР не существует (еще не найдено). Существует несколько алгоритмов реализации метода ветвей и границ. Рассмотрим один из них.

Примечание. В приведенном ниже описании предполагается, что целевая функция решаемой задачи подлежит *максимизации*.

1. Исходная задача решается симплекс-методом без учета требований целочисленности. Если все переменные, которые по своему смыслу должны быть целочисленными, принимают в оптимальном решении целочисленные значения, то решение задачи на этом завершается. В противном случае выполняется следующий шаг.

2. Пусть некоторая переменная X_i , которая по своему смыслу должна быть целочисленной, приняла дробное значение R_i (если таких переменных несколько, то выбирается *любая* из них). Пусть значение целевой функции в оптимальном (но нецелочисленном) решении равно E . В список решаемых задач включаются две задачи. В одну из них добавляется ограничение $X_i \geq [R_i + 1]$, в другую – ограничение $X_i \leq [R_i]$, где $[]$ – целая часть. Величина E используется в качестве *оценки* этих задач.

3. Из списка решаемых задач выбирается задача с *максимальным значением оценки* (если имеется несколько задач с одинаковыми оценками, то выбирается любая из них). Эта задача решается симплекс-методом.

4. В зависимости от результатов решения задачи выполняется одно из следующих действий:

- если *задача не имеет решений*, то выполняется переход к шагу 5;
- если решение задачи оказалось *нецелочисленным*, и при этом *еще не существует ТНР* (еще не найдено ни одного целочисленного решения), то выполняется возврат к шагу 2;
- если решение задачи оказалось *нецелочисленным*, и при этом *уже существует ТНР*, то значение целевой функции, полученное в задаче (E), сравнивается с $E_{\text{ТНР}}$. Если $E > E_{\text{ТНР}}$, то выполняется возврат к шагу 2; если $E \leq E_{\text{ТНР}}$, то выполняется переход к шагу 5;
- если решение задачи оказалось *целочисленным*, и при этом *еще не существует ТНР*, то найденное решение принимается в качестве ТНР, и выполняется переход к шагу 5;
- если решение задачи оказалось *целочисленным*, и при этом *уже существует ТНР*, то значение целевой функции, полученное в задаче (E), сравнивается с $E_{\text{ТНР}}$. Если $E > E_{\text{ТНР}}$, то найденное решение *принимается в качестве ТНР* и выполняется переход к шагу 5; если $E \leq E_{\text{ТНР}}$, то выполняется переход к шагу 5.

5. Если в списке решаемых задач *имеется хотя бы одна задача*, то выполняется возврат к шагу 3. Если в списке *нет ни одной задачи*, то процесс решения завершен. Если по окончании решения существует ТНР, то оно является оптимальным целочисленным решением задачи. Если ТНР не существует, то задача не имеет целочисленных решений.

Применение метода ветвей и границ рассмотрим на следующем примере.

Пример. Предприятие выпускает панели для пультов управления двух видов: стандартные (для работы в обычных условиях) и специального исполнения (для работы при повышенных температурах). При изготовлении панелей используется пластмасса и алюминий. Для изготовления одной стандартной панели требуется 12,5 кг пластмассы и 8 кг алюминия; для изготовления одной панели специального исполнения – 7,2 кг пластмассы и 14,5 кг алюминия.

Предприятие имеет 300 кг пластмассы и 400 кг алюминия. Прибыль предприятия от выпуска одной стандартной панели составляет 7 ден. ед., прибыль от выпуска одной панели специального исполнения – 9 ден. ед. Требуется определить, сколько панелей каждого вида должно выпускать предприятие, чтобы получить максимальную прибыль.

Обозначим количество выпускаемых стандартных панелей через X_1 , количество выпускаемых панелей специального исполнения – через X_2 . Математическая модель задачи будет иметь следующий вид:

$$\begin{aligned} 12,5X_1 + 7,2X_2 &\leq 300 \\ 8X_1 + 14,5X_2 &\leq 400 \\ X_1 &\geq 0, X_2 \geq 0. \\ X_1, X_2 &- \text{целые.} \\ E = 7X_1 + 9X_2 &\rightarrow \max. \end{aligned}$$

Приведем задачу к стандартной форме:

$$\begin{aligned} 12,5X_1 + 7,2X_2 + X_3 &= 300 \\ 8X_1 + 14,5X_2 + X_4 &= 400 \\ X_i &\geq 0, i = 1, \dots, 4. \\ X_1, X_2 &- \text{целые.} \\ E = 7X_1 + 9X_2 &\rightarrow \max. \end{aligned}$$

Остаточные переменные X_3 и X_4 обозначают, соответственно, неиспользованный остаток пластмассы и алюминия (в килограммах). Эти переменные могут принимать как целые, так и дробные значения. Таким образом, задача является частично целочисленной.

Для решения задачи воспользуемся методом ветвей и границ. Ход решения показан на рис. 1. номера задач соответствуют порядку их включения в список решаемых задач. Рамкой выделены целочисленные решения. Перечеркнуты постановки задач, для которых не потребовалось определять решение.

Примечание. Следует обратить внимание, что номера задач обозначают именно порядок их включения в список решаемых задач, а не порядок решения.

Исходная задача (на рис. 1 она обозначена как задача 1) решается симплекс-методом. Решение оказалось нецелочисленным: $X_1 = 11,89$, $X_2 = 21,03$. По условию задачи обе переменные должны принимать целочисленные значения. Для продолжения поиска оптимального решения выбирается любая из этих переменных, например, X_1 . В список решаемых задач включаются

задачи 2 и 3. В задачу 2 входит ограничение $X_1 \leq 11$, а в задачу 3 – ограничение $X_1 \geq 12$. Смысл этих ограничений следующий:

- эти ограничения *исключают из ОДР оптимальное, но нецелочисленное решение* $X_1 = 11,89$, $X_2 = 21,03$, так как значение $X_1 = 11,89$ не соответствует ни ограничению $X_1 \leq 11$, ни ограничению $X_1 \geq 12$. Таким образом, в ходе дальнейшего решения задачи исключается возврат к оптимальному, но нецелочисленному решению;
- эти ограничения *не исключают из ОДР ни одного допустимого целочисленного решения*, так как любое целочисленное значение X_1 соответствует либо ограничению $X_1 \leq 11$, либо ограничению $X_1 \geq 12$.

Оценкой задач 2 и 3 является величина $E = 272,46$. Эта величина соответствует оптимальному (но нецелочисленному) решению задачи 1 ($X_1 = 11,89$, $X_2 = 21,03$). В задачах 2 и 3 из-за включения дополнительных ограничений ($X_1 \leq 11$ для задачи 2, $X_1 \geq 12$ для задачи 3) будет получено какое-то другое решение. Поэтому значения целевых функций в задачах 2 и 3 не могут быть больше, чем 272,46.

Так как задачи 2 и 3 имеют одинаковые оценки, для решения можно выбрать любую из них. Пусть выбрана задача 2. Она решается симплекс-методом. Решение также оказывается нецелочисленным. В список решаемых задач включаются задачи 4 и 5 с оценкой 270,66. Для составления этих задач вводятся ограничения $X_2 \leq 21$ и $X_2 \geq 22$. Эти ограничения имеют тот же смысл, что и ограничения, используемые при составлении задач 2 и 3 (см. выше).

Таким образом, в списке решаемых задач находятся три задачи: 3 (с оценкой 272,46), 4 и 5 (с оценкой 270,66). Для решения выбирается задача с максимальной оценкой. В данном случае это задача 3. При ее решении максимальное значение целевой функции может составить не более чем 272,46, а для задач 4 или 5 – не более 270,66.

Примечание. Важно обратить внимание, что оценка задачи показывает именно максимально возможное значение целевой функции этой задачи. Оценка задачи известна до того, как она будет решена. Действительное оптимальное решение задачи можно определить, только решив ее. Поэтому в рассматриваемом примере выбор задачи 3 не означает, что ее решение обязательно будет лучше, чем у задач 4 или 5.

Решение задачи 3 также оказывается нецелочисленным. В список решаемых задач включаются задачи 6 и 7 с оценкой 271,5.

В списке решаемых задач находятся четыре задачи: 4 и 5 (с оценкой 270,66), 6 и 7 (с оценкой 271,5). Для решения можно выбрать любую из задач 6 или 7. Пусть выбрана задача 6. Ее решение оказалось нецелочисленным. В список решаемых задач включаются задачи 8 и 9 с оценкой 267,36.

В списке решаемых задач оказалось пять задач: 4 и 5 (с оценкой 270,66), 7 (с оценкой 271,5), 8 и 9 (с оценкой 267,36). Для решения выбирается задача 7. Она не имеет допустимых решений.

В списке решаемых задач осталось четыре задачи: 4 и 5 (с оценкой 270,66), 8 и 9 (с оценкой 267,36). Для решения можно выбрать задачу 4 или 5. Пусть выбрана задача 4. Ее решение оказалось целочисленным. Оно становится текущим наилучшим решением. Оценки задач 5, 8 и 9, находящихся в списке решаемых задач, сравниваются со значением целевой функции, полученным для целочисленного решения ($E_{\text{ТНР}} = 266$). Оценки всех задач лучше, чем величина $E_{\text{ТНР}}$. Это значит, что при решении этих задач могут быть получены лучшие решения, чем ТНР. Поэтому ни одна из задач не исключается.

В списке решаемых задач осталось три задачи: 5 (с оценкой 270,66), 8 и 9 (с оценкой 267,36). Для решения выбирается задача 5. Ее решение нецелочисленно. Значение целевой функции ($E = 268,88$) лучше, чем $E_{\text{ТНР}} = 266$. Поэтому в список решаемых задач включаются две новые задачи (10 и 11) с оценкой 268,66.

Таким образом, в списке решаемых задач находятся четыре задачи: 8 и 9 (с оценкой 267,36), 10 и 11 (с оценкой 268,88). Для решения необходимо выбрать задачу 10 или 11. Пусть выбрана задача 10. Ее решение нецелочисленно. Значение целевой функции ($E = 268,62$) лучше, чем $E_{\text{ТНР}} = 266$. Поэтому в список решаемых задач включаются две новые задачи (12 и 13) с оценкой 268,62.

В списке решаемых задач находятся пять задач: 8 и 9 (с оценкой 267,36), 11 (с оценкой 268,88), 12 и 13 (с оценкой 268,62). Для решения выбирается задача 11. Она не имеет допустимых решений.

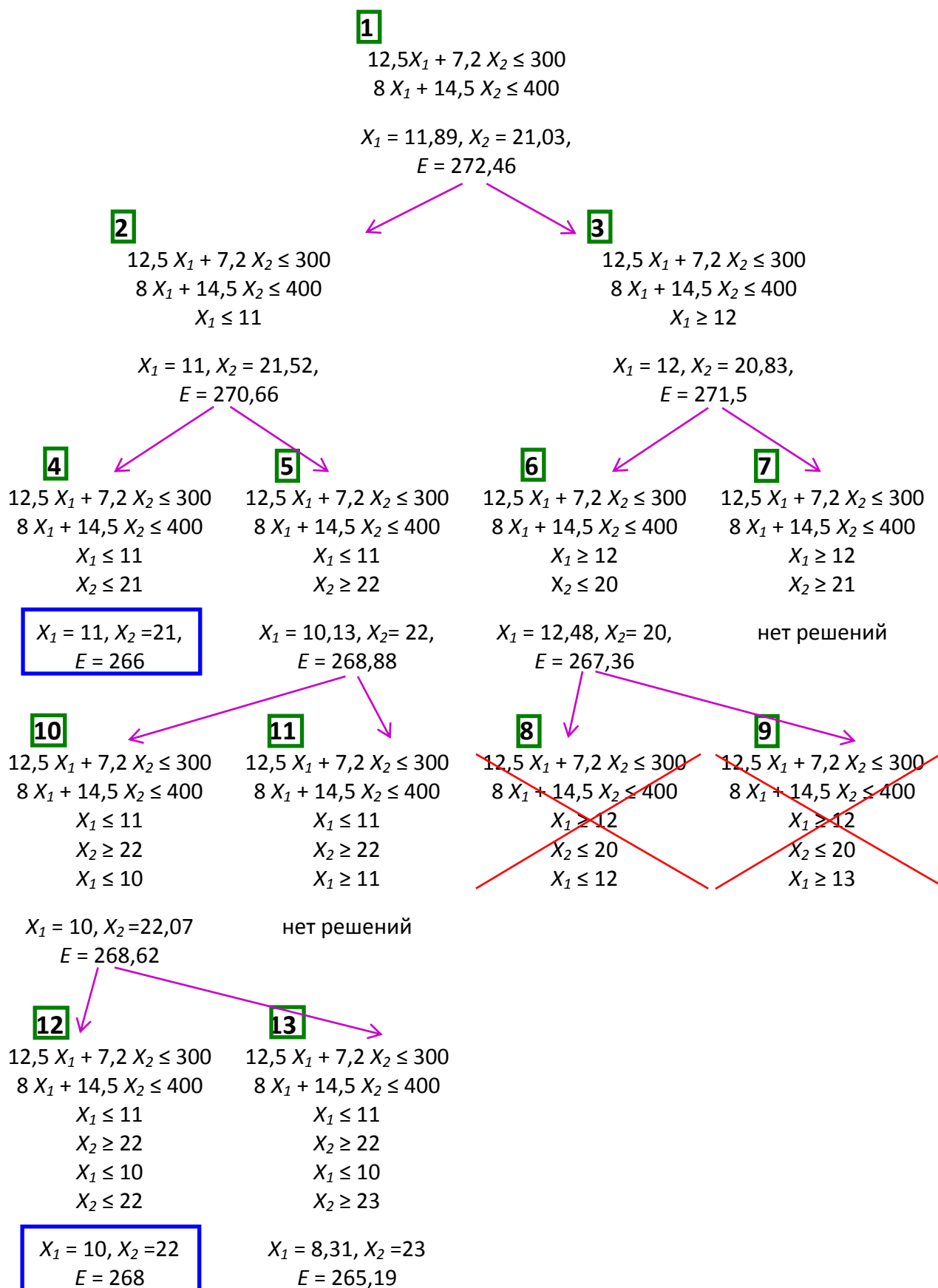


Рис. 7.1. Решение примера методом ветвей и границ

В списке решаемых задач осталось четыре задачи: 8 и 9 (с оценкой 267,36), 12 и 13 (с оценкой 268,62). Для решения можно выбрать задачу 12 или 13. Пусть выбрана задача 12. Ее решение оказалось целочисленным. Значение целевой функции ($E = 268$) лучше, чем $E_{\text{ТНР}} = 266$. Таким образом, получено лучшее решение, чем ТНР, найденное ранее в задаче 4. Полученное решение $X_1 = 10$, $X_2 = 22$ становится текущим наилучшим решением, а величина $E_{\text{ТНР}}$ принимает значение 268. Кроме того, необходимо сравнить с новым значением $E_{\text{ТНР}}$ оценки всех задач, имеющих в списке решаемых задач. Оценки задач 8 и 9 (267,36) хуже, чем $E_{\text{ТНР}}$. Поэтому решать эти задачи не имеет смысла, так как их решение будет хуже, чем уже полученное целочисленное решение (ТНР). Задачи 8 и 9 исключаются из списка решаемых задач. Исключить задачу 13 нельзя, так как ее оценка (268,62) лучше, чем $E_{\text{ТНР}}$.

В списке решаемых задач остается одна задача: это задача 13. Она решается симплекс-методом. Ее решение нецелочисленно. Значение целевой функции ($E = 265,19$) хуже, чем $E_{\text{ТНР}}$. Поэтому включать в список решаемых задач новые задачи не требуется.

В списке решаемых задач не осталось ни одной задачи. Таким образом, получено оптимальное решение: $X_1 = 10$, $X_2 = 22$, $E = 268$. Это значит, что предприятию следует выпустить 10 стандартных панелей и 22 панели специального исполнения. Прибыль предприятия в этом случае будет максимальной и составит 268 ден. ед.

Можно также определить, что неизрасходованный остаток запаса пластмассы составит 16,6 кг, а алюминия – 1 кг.

Примечание. Математические модели некоторых задач, составляемых в ходе поиска решения по методу ветвей и границ, можно упростить. Например, в задаче 10 можно заменить ограничения $X_1 \leq 11$ и $X_1 \leq 10$ на одно ограничение: $X_1 \leq 10$. В задаче 12 можно заменить ограничения $X_2 \geq 22$ и $X_2 \leq 22$ на $X_2 = 22$. Аналогичные упрощения возможны и в задачах 8, 9, 11, 13.

Метод ветвей и границ хорошо работает в *задаче коммивояжера*. Напомним, что задача заключается в нахождении маршрута, который проходит через каждый из N данных городов в точности один раз и при этом имеет наименьшую стоимость

Задача о рюкзаке в общем виде. Имеется набор предметов, каждый с определенным весом и определенной стоимостью. Из этого набора необходимо выбрать предметы с максимальной стоимостью, с учетом ограничения на максимальный вес (вес «рюкзака»).

Если бы число предметов не предполагалось целым, то задача бы решалась как задача линейного программирования, например, симплекс-методом.

Вследствие целочисленности числа предметов, задача становится задачей целочисленного линейного программирования и решается, например, методом ветвей и границ.

Лабораторная работа №8 (факультативная)
ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ И
ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ

Перед выполнением лабораторной работы рекомендуем:

1. Изучить презентацию лектора курса: «Численные методы решения дифференциальных и интегральных уравнений», материалы доступны в облаке на Mail.ru.

2. Прочитать соответствующие разделы книги:

Каханер Д., Моулера К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ. - М.: Мир, 1998. - 575 с.,

Цель работы:

1. Изучить и научиться использовать на практике алгоритмы решения дифференциальных и интегральных уравнений.
2. Написать программу, реализующую алгоритм решения дифференциального уравнения в частных производных или интегрального уравнения.

Содержание отчета:

1. Постановка задачи. Формульное описание алгоритма.
2. Описание компьютерной программы. Объяснение значений всех параметров.
3. Нахождение приближённого решения дифференциального или интегрального уравнения.
4. Оценка скорости работы алгоритма.
3. Сравнение эффективности различных алгоритмов решения дифференциальных и интегральных уравнений.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Приближенное решение дифференциальных уравнений в частных производных и для интегральных уравнений обычно сводится к приближенной аппроксимации операторов и искомой функции, при которой неизвестные коэффициенты определяются из системы линейных алгебраических уравнений. В

настоящее время созданы автоматизированные комплексы для ряда численных методов.

Рассмотрим основные идеи, на которых базируются эти методы. Пусть дано уравнение: $K\varphi = f$, где $\varphi(x, y, z)$ - неизвестная функция, $f(x, y, z)$ - известная функция (правая часть уравнения), K - дифференциальный или интегральный оператор. Рассмотрим группу методов под общим названием: метод взвешенных невязок.

Будем искать решение в виде линейной комбинации функций $\varphi = \sum C_k \varphi_k$, где C_k - коэффициенты, подлежащие определению, φ_k - базисные функции.

В общем случае задача разрешима, только если искомая функция принадлежит к оболочке, натянутой на базисные функции, поэтому в общем случае можно говорить лишь о псевдорешении, минимизирующем невязку

$$r = f - K\varphi_{\text{приближ.}}$$

Общая часть метода заключается во введении системы весовых функций $\{w_i\}$, $i = 1, 2, \dots, m$ и приравнении к нулю скалярных произведений (r, w_i) , где $r = f - K \sum_k C_k \varphi_k$.

Метод точечной коллокации.

Реализация метода заключается в следующем. Фиксируется m -точек (коллокаций), $M_i, i = 1, \dots, m$. Базисные функции выбирают в виде дельта - функций, $w_i = \delta_i(M - M_i) \Rightarrow (r, \delta_i) = 0$, т.е. невязка в m точках должна быть равна нулю, при этом система уравнений может получиться переопределённой при $m > n$.

Метод коллокации в подобластях.

При реализации этой модификации метода коллокаций пытаются интегрально минимизировать невязку на элементе

$$V = \bigcup_{i=1}^m V_i, \quad w_i(M) = \begin{cases} 1, & M \in V_i, \\ 0, & M \notin V_i. \end{cases}$$

Численные эксперименты показывают, что эта модификация, иногда встречающаяся под названием метод средних потенциалов, практически не даёт улучшения по сравнению с методом точечной коллокации.

Метод наименьших квадратов.

Коэффициенты C_i - определяют из условия минимума квадрата невязки

$$r^2 = \int \left(f - \sum_k C_k K \varphi_k \right)^2 dV,$$

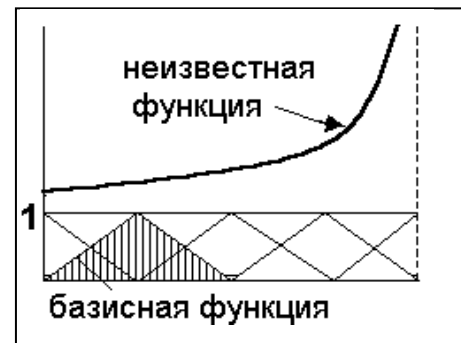
$$\frac{1}{2} \frac{\partial r^2}{\partial C_i} = \int \left(f - \sum_k C_k K \varphi_k \right) K \varphi_i dV = 0.$$

Этот метод более сложен в реализации и приводит часто к плохо обусловленным СЛАУ.

Метод Галеркина.

В качестве весовых функций $\{W_i\}$ выбирают сами функции $\{\varphi_i\}$, $m=n$, т.е. требуем, чтобы невязка была ортогональна базисным функциям.

Если $\{W_i\}$ - финитные функции, то получаем



метод конечных элементов. В методе конечных элементов обычно используется кусочно – линейная аппроксимация искомой функции. Отметим, что кусочно – линейная аппроксимация сама по себе практически не дает преимущества перед кусочно – постоянной аппроксимацией, также как метод трапеций вычисления интегралов не дает преимущества перед методом прямоугольников.

Одним из наиболее эффективных численных методов расчета полей является метод граничных интегральных уравнений. Далее мы подробно рассмотрим алгоритмы решения некоторых интегральных уравнений, возникающих в задачах электромагнетизма.

Кроме того, одним из популярных в 60-х, 70-х годах был метод сеток, доведенный до совершенства многими исследователями, он и сейчас не утратил полностью своего значения. Рассмотрим простой пример использования этого метода.

Пример 1. Решение двумерное уравнение Лапласа внутри прямоугольника методом сеток.

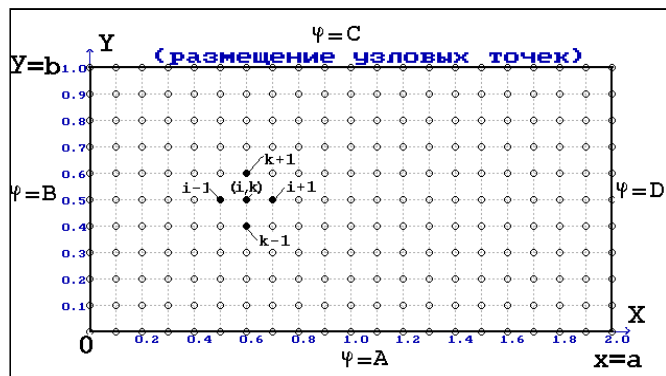


Рис. 8.1

Дано: $\Delta\varphi = 0$, на S

$$\varphi = A, \quad y = 0$$

$$\varphi = B, \quad x = a$$

$$\varphi = C, \quad y = b$$

$$\varphi = D, \quad x = 0, \text{ рис.8.1.}$$

Важнейшей частью алгоритма является численная аппроксимация оператора Лапласа. Сначала аппроксимируем первые и вторые производные:

$$\frac{\partial \varphi_i}{\partial x} \approx \frac{1}{\Delta x} (\varphi_{i+1} - \varphi_i), \quad \frac{\partial^2 \varphi_i}{\partial x^2} = \frac{1}{\Delta x^2} (\varphi_{i+1} + \varphi_{i-1} - 2\varphi_i),$$

$$\frac{\partial^2 \varphi_{i,k}}{\partial y^2} = \frac{1}{\Delta y^2} (\varphi_{i,k+1} + \varphi_{i,k-1} - 2\varphi_i), \text{ где } i - \text{индекс переменной } x, k - \text{индекс переменной } y.$$

Наконец, получаем аппроксимацию оператора Лапласа:

$$\Delta\varphi = -2\varphi_{i,k} \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) + \frac{1}{\Delta x^2} (\varphi_{i+1,k} + \varphi_{i-1,k}) + \frac{1}{\Delta y^2} (\varphi_{i,k+1} + \varphi_{i,k-1}) = 0,$$

откуда выражаем значения неизвестной функции:

$$\varphi_{i,k} = \frac{1}{2} \frac{\Delta y^2}{\Delta x^2 + \Delta y^2} (\varphi_{i+1,k} + \varphi_{i-1,k}) + \frac{1}{2} \frac{\Delta x^2}{\Delta x^2 + \Delta y^2} (\varphi_{i,k+1} + \varphi_{i,k-1}).$$

Если $\Delta x = \Delta y$, то имеем:

$$\varphi_{i,k} = \frac{1}{4} (\varphi_{i+1,k} + \varphi_{i-1,k} + \varphi_{i,k+1} + \varphi_{i,k-1}).$$

Это выражение является основой численного алгоритма.

Ядром программы является следующий фрагмент:

```
for iter:=1 to niter do begin {цикл по итерациям}
    s1:=0; s2:=0; s3:=0; {обнуление сумматоров}
```

```

for ix:=2 to nx-1 do begin {цикл по X}
  for iy:=2 to ny-1 do begin {цикл по Y}
    fi_new[ix,iy]:=0.25*fi[ix+1,iy]+fi[ix-1,iy]+
      fi[ix,iy+1]+fi[ix,iy-1]);
    tem:=abs(fi_new[ix,iy]-fi_prec[ix,iy]);
    if s1<(tem/D) then s1:=tem/D;
    s2:=s2+sqr(fi_new[ix,iy]-fi_prec[ix,iy]);
    s3:=s3+sqr(fi_prec[ix,iy]);
    end; end;
    s2:=sqrt(s2/s3);

```

Пример 2. Численное решение интегрального уравнения 1-го рода на примере расчета емкости квадратной пластины

Задача вычисления емкости квадратной пластины является удобным и хорошо изученным «полигоном» для испытания различных численных методов.

Емкость уединенной пластины равна отношению заряда пластины к потенциалу относительно бесконечно удаленной точки, $c = q/\varphi$. При этом емкость не зависит ни от заряда, ни от потенциала, поэтому для её вычисления можно задать значение или заряда или потенциала. Вследствие того, что заряд и потенциал вычисляются через значения плотности заряда, можно утверждать, что задача точного вычисления емкости сводится к задаче вычисления плотности заряда на уединенном проводнике (или, в случае конденсатора, к задаче нахождения распределения индуцированных зарядов на двух проводниках, заряженных равными по модулю, но разноименными зарядами).

Несмотря на геометрическую простоту и наличие симметрий у квадрата, до сих пор задача о собственном распределении заряда не решена аналитически, несмотря на то, что этой задачей в разные годы занимались известные ученые: Г. Кавендиш, Релей, математики Г. Поля и Г. Сеге. Отметим, что до сих пор не существует строгого обоснования гипотезы Релея о минимальности емкости круга среди всех равновеликих по площади пластин, хотя многие рассуждения и численные результаты говорят в пользу её истинности. В дальнейшем под величиной емкости будем понимать отношение емкости пластины к ем-

кости равновеликого по площади круга, емкость круга равна $8 \cdot \varepsilon_0 \cdot R$, где R - радиус круга. В таблице приведены некоторые исторические результаты вычисления электроемкости квадратной пластины.

Кавендиш Г. (19 век)	1.0055
Релей (19 век)	1
Полиа Г., Сеге Г. (примерно 1930г.)	$1 < c < 1.046$
Вишневский А.М. (1981г.)	1.018

Наши численные эксперименты показывают примерно значение 1,02. При этом мы повторили результат Вишневого для аналогичного числа разбиений квадрата (решена СЛАУ, размерностью 4900*4900).

Для нахождения плотности заряда можно использовать один из численных методов, однако на наш взгляд для решения этой задачи самым эффективным является метод граничных интегральных уравнений т.к. он позволяет сразу снизить размерность задачи и искать источники поля на поверхности, не рассматривая поле в пространстве.

Интегральное уравнение первого рода относительно плотности распределения зарядов имеет вид:

$$k \cdot \int_S \frac{\sigma(P) \cdot dS_P}{r_{PM}} = \varphi,$$

где: $k = 9 \cdot 10^9 \text{ м/Ф}$, σ - искомое распределение заряда, φ - заданный потенциал пластины, принимающий на поверхности постоянное значение.

Решение интегральных уравнений 1го рода длительное время сдерживалось из-за теоретической некорректности задачи (нарушено третье условие корректности по Адамару – условие устойчивости решения по отношению к малым отклонениям исходных данных и к погрешностям аппроксимации интегрального оператора). Поэтому полагалось, что такие уравнения должны решаться с привлечением методов регуляризации.

Однако численные результаты, полученные различными авторами, показывают высокую устойчивость результатов даже при большом числе разбиений

поверхности, достигающем 5000. На наш взгляд это связано с хорошей «физической» обусловленностью задачи, в отличие от задач гравитационной или магниторазведки, где источники поля и точки вычисления поля, как правило, далеко разнесены в пространстве.

Опишем алгоритм численного решения интегрального уравнения:

1. На первом этапе осуществляется дискретизация (разбиение) области интегрирования на элементы. В нашем случае это могут быть квадраты постоянной или переменной величины. Разбиваем четверть квадрата на $N \times N$ одина-

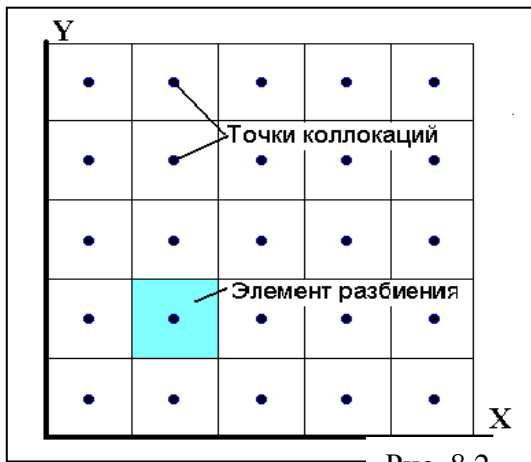


Рис. 8.2

ковых квадратов, здесь N - число разбиений половины стороны квадрата

2. Сведение интегрального уравнения к системе линейных алгебраических уравнений (СЛАУ) и вычисление матрицы СЛАУ. Наиболее просто этот этап реализуется в случае кусочно-постоянной аппроксимации искомой функции и использовании метода

точечной коллокации, когда требуется равенство левой и правой части уравнения только в центрах элементов разбиений (метод площадок).

Вычисляем коэффициенты матрицы СЛАУ.

Диагональные коэффициенты легко вычисляются аналитически:

$$A_{kk} = 8R \cdot \ln \operatorname{tg} \frac{3\pi}{8} \approx 7.051 \cdot R.$$

Для вычисления недиагональных элементов в дальней зоне для ускорения вычислений можно воспользоваться приближенной формулой, основанной на разложении потенциала по мультипольным моментам:

$$A_{ki} = \int_S \frac{dS_i}{r_{ki}} \approx \frac{4R^2}{r_{ki}} \cdot \left(1 + \frac{1}{6} \cdot \frac{R^2}{r_{ki}^2} \right), \text{ где } 2R - \text{размер элемента.}$$

Численные эксперименты показывают, что эта формула удовлетворительно работает и в ближней зоне, при вычислении потенциала соседей.

Главная, на наш взгляд, причина вычислительных погрешностей метода площадок заключается в наличии сингулярностей в искомой функции вблизи углов и гладких краев пластин.

Значения потенциала выбрать равным 1 Вольт.

3. Численное решение СЛАУ. Долгое время существовала тенденция преувеличения трудностей решения СЛАУ при решении интегральных уравнений 1го рода. Многие обратные задачи действительно приводят к плохо обусловленным СЛАУ, для решения которых необходимо использовать методы, основанные на минимизации сглаживающего функционала, либо на оптимальном останове итерационных процедур типа наискорейшего спуска или сопряженных градиентов.

При расчете собственного распределения заряда обычно возникают хорошо обусловленные СЛАУ, для решения которых можно использовать любые численные методы, например, метод исключения Гаусса. В настоящее время, когда практически сняты ограничения по объему используемой оперативной памяти компьютера и повышена точность арифметики, размерность матриц может быть повышена. Нами без труда решались методом исключения Гаусса с частичной выборкой ведущего элемента системы размерностью до 4900×4900 .

4. Вычислить электроемкость, отнесенную к емкости равновеликого по площа-

$$\text{ди круга: } C_{\text{отн}} = \frac{16R^2 \cdot \sum_{i=1}^{N \times N} \sigma_i}{8 \cdot \varepsilon_0 \cdot a \cdot \sqrt{4/\pi}}.$$

Должен получиться результат близкий к 1,02.

Основная учебная литература:

1. Абрамкин, Г.П. Численные методы [Электронный ресурс] : учебное пособие / Г.П. Абрамкин. — Электрон. дан. — Барнаул : АлтГПУ, 2016. — 260 с. — Режим доступа: <https://e.lanbook.com/book/112165>. — Загл. с экрана.
2. Бахвалов, Н.С. Численные методы : учебник / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. — 9-е изд. — Москва : Лаборатория знаний, 2020. — 636 с. — ISBN 978-5-00101-836-0. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: <https://e.lanbook.com/book/126099> (дата обращения: 31.10.2019). — Режим доступа: для авториз. пользователей.
3. Копченова, Н.В. Вычислительная математика в примерах и задачах : учебное пособие / Н.В. Копченова, И.А. Марон. — 4-е изд., стер. — Санкт-Петербург : Лань, 2017. — 368 с. — ISBN 978-5-8114-0801-6. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: <https://e.lanbook.com/book/96854> (дата обращения: 31.10.2019). — Режим доступа: для авториз. пользователей.
4. Петров, И.Б. Введение в вычислительную математику : учебное пособие / И.Б. Петров, А.И. Лобанов. — 2-е изд. — Москва : ИНТУИТ, 2016. — 351 с. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: <https://e.lanbook.com/book/100737> (дата обращения: 31.10.2019). — Режим доступа: для авториз. пользователей.

Дополнительная учебная литература:

5. Дж. Форсайт, М.Малькольм, К. Моулер. Машинные методы математических «вычислений» —М.: Мир,1980.
6. Д.К.Фаддеев, В.Н.Фаддеева, Вычислительные методы линейной алгебры.- М.: Наука.- 1963, 655.
7. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. - М.: Наука, 1987.
8. Самарский А.А, Гулин А.В. Численные методы. - М. : Наука, 1989.
9. Пирумов У.Г. Численные методы. —М.:Дрофа,2003.-224с

- 10.Боглаев Ю.П. Вычислительная математика и программирование. М. :
Высшая школа,1990.
- 11.Волков Е.А. Численные методы. - М.: Наука, 1987.
- 12.Турчак Л.И. Основы численных методов.-М.:Наука,1987.
- 13.Т. Шуп. Решение инженерных задач на ЭВМ: Практическое руководство.
Пер. с англ.— М.: Мир. 1982,—238 с, ил.
- 14.Тихонов А.Н., Костомаров Д.П. Вводные лекции по прикладной математике.
- М.: Наука, 1984.
- 15.Поттер Д. Вычислительные методы в физике. - М.:Мир,1977.
- 16.Ортега Дж., Пул У. Введение в численные методы решения
дифференциальных уравнений .- М.: Наука, 1986.
- 17.Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и
Паскаль .-Томск: МП «Раско»,1991.
- 18.Ракитин В.И., Первушин В.Е. Практическое руководство по методам
вычислений .-М.: Высшая школа, 1998.
- 19.Хэмминг Р.В. Численные методы для научных работников и инженеров. -
М.: Наука, 1984.
- 20.Д. Роджерс, Дж. Адамс Математические основы машинной графики .-М.:
«Мир», 2001.
- 21.Рашиков В. И.. Рошаль А. С. Численные методы решения физических задач:
Учебное пособие. — СПб.: Издательство «Лань», 2005.-208 с.
- 22.Гилл Ф., Мюррей У., Райт М. Практическая оптимизация .- М.:Мир, 1965 .-
509 с.
- 23.Зализняк В.Е. Основы научных вычислений. Введение в численные методы
для физиков: Учебное пособие для студентов естественно-научных и
технических специальностей высших учебных заведений . - М.: Эдиториал
УРСС. 2002.- 296 с.
24. Кандауров И.В.. Мезенцев Н.А, Пиндюрин В.Ф.. Симонов ЕА.
Моделирование физических явлений на ЭВМ .- Методическое пособие,
Часть III, Новосибирск, 2000.

- 25.Пирумов У.Г. Численные методы.- Учебное пособие .- М: Изд. МАИ, 1998.
- 26.Самарский Л. Л., Михайлов Л.П. Математическое моделирование: Идеи. Методы. Примеры. 2-е изд, испр., М.: Физматлит, 2001.- 320 с.
- 27.Тихонов А.Н., Арсенин В.Я. Методы решения некорректных задач. М.: Наука .- 1979 .- 142с.
- 28.Валов, Л.В. Численные методы решения уравнений для инженеров: Учебное пособие / Л.В. Валов. - Челябинск: Издательский центр ЮУрГУ, 2012. -110с.
- 29.Э.В. Денисова, Л.В. Кучер, Краткий курс вычислительной математики: Учебно-методическое пособие. - СПб: СПбГУ ИТМО, 2013. - 90с.
- 30.Кобельков Г.М. Курс лекций по численным методам .- М.: МГУ .- 2006.
- 31.Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: Пер. с англ.- М.: Мир, 1998.- 575 с.
- 32.Мухамадеев И.Г. Алгоритмы вычислительной математики .- Курс лекций, Уфа, 2007.
- 33.Таха Х. Введение в исследование операций.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001
- 34.Сморodinский С.С., Батин Н.В. Оптимизация решений на основе методов и моделей математического программирования: Уч. Пособие по курсу «Системный анализ и исследование операций». – Минск: БГУИР, 2003. – 136 с.
- 35.Кононюк А.Е. Основы теории оптимизации. Безусловная оптимизация. Книга 2, ч.1. Киев: "Освіта України", 2011. 544 с.
- 36.Дегтярев Ю.И. Методы оптимизации: - М.: Сов. радио, 1980. 270 с.
- 37.Гольштейн Е.Г., Юдин Д.Б. Задачи линейного программирования транспортного типа. - М.: Наука, 1969. – 382 с.
- 38.Лемешко, Б.Ю. Методы оптимизации: Конспект лекций. – Новосибирск: Изд-во НГТУ, 2009. – 126 с.