

# СПЕЦИАЛЬНЫЕ ГЛАВЫ МАТЕМАТИКИ

# 1. ЧИСЛЕННОЕ РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

## 1.1. ПОСТАНОВКА ЗАДАЧИ И ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Пусть имеется уравнение

$$f(x) = 0, \quad (1.1)$$

где  $f(x)$  – функция, которая определена и непрерывна в некотором интервале  $a < x < b$ . Всякое значение  $\xi$  (кси), обращающее функцию  $f(x)$  в нуль, т. е. такое, что  $f(\xi) = 0$ , называется *корнем* уравнения (1.1) или нулем функции  $f(x)$ .

Если функция  $f(x)$  является *алгебраическим* многочленом любой степени, *отличной от единицы*, уравнение называют *нелинейным алгебраическим*, например:  $x^3 - 6x + 2 = 0$ .

Если в функцию  $f(x)$  входят *элементарные* функции (тригонометрические, логарифмические и др.), то уравнение называют *трансцендентным*, например:  $x + \exp(x) = 0$ ,  $\arcsin(2x + 1) - x^2 = 0$ ,  $x - \ln x = 0$  и т.д.

В дальнейшем и те и другие будем называть просто *нелинейными уравнениями*.

В тех случаях, когда уравнение (1.1) достаточно сложно, его корни найти точно, как правило, не удастся (*аналитического выражения для корней не существует*). Кроме того, в инженерной практике обычно имеют дело с уравнениями, коэффициенты в которых известны лишь *приблизительно*, отсюда возникает задача *приближенного* нахождения корней уравнения и *оценки степени их точности*, иначе говоря, *погрешности* решения уравнения. Для решения такой задачи и используют *численные* методы.

Численное решение уравнения (1.1) состоит из *двух этапов*:

- *отделения* корня, т. е. установления возможно *малого промежутка*  $[a, b]$ , в котором содержится *один (и только один)* корень уравнения;
- *уточнения* корня, т. е. нахождения значения корня в промежутке  $[a, b]$  с *заданной точностью (допустимой погрешностью)*.

Таким образом, вопросы, на которые необходимо ответить при численном решении уравнения (1.1) могут быть сформулированы следующим образом:

1. Имеет ли уравнение (1.1) на отрезке  $[a, b]$  хотя бы один вещественный (не мнимый или комплексный) корень?

2. Является ли этот корень единственным?
3. Каково значение этого корня, отличающееся от точного не более чем на некоторую малую величину  $\varepsilon$  (эпсилон), т. е. каково значение  $x$ , для которого выполняется условие  $|x - \xi| \leq \varepsilon$ ?

Ответы на первые два вопроса достигаются в ходе *отделения* корней уравнения, а на последний – при *уточнении* корня.

## 1.2. ОТДЕЛЕНИЕ КОРНЕЙ УРАВНЕНИЯ

Из математического анализа известна следующая теорема: если непрерывная функция  $f(x)$  на концах отрезка  $[a, b]$  принимает значения разных знаков, т. е.  $f(a)f(b) < 0$ , то внутри этого отрезка содержится по меньшей мере один корень уравнения  $f(x) = 0$ , т. е. найдется хотя бы одно число  $\xi \in [a, b]$  (кси), принадлежащее отрезку  $[a, b]$  такое, что  $f(\xi) = 0$ . Эта теорема иллюстрируется рис. 1.1, а.

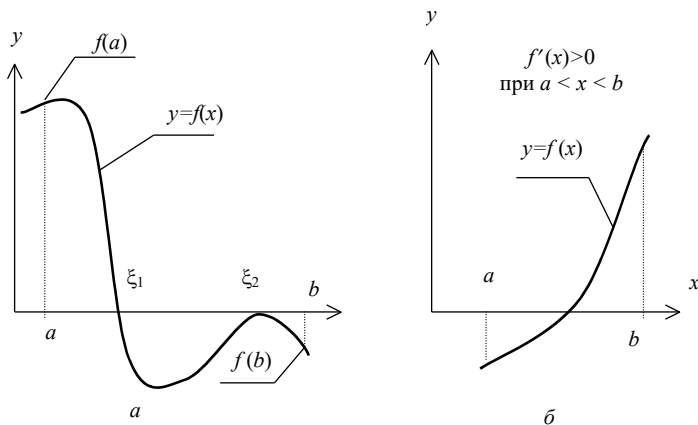


Рис. 1.1. Геометрическая иллюстрация процесса отделения корня

Для ответа на второй вопрос следует воспользоваться другой теоремой из математического анализа: если производная  $f'(x)$  существует и сохраняет постоянный знак внутри промежутка (интервала)  $[a, b]$ , т. е. если  $f'(x) > 0$  (или  $f'(x) < 0$ ) для  $a < x < b$ , то уравнение  $f(x) = 0$  на данном промежутке имеет *один единственный* корень. Эта теорема иллюстрируется рис. 1.1, б.

Процесс установления возможно малого промежутка  $[a, b]$ , в котором содержится только один корень уравнения, называется

*отделением корня.* При наличии любого (программируемого) средства вычислительной техники (ВТ) наиболее рациональным представляется следующий алгоритм отделения корней уравнения  $f(x) = 0$  на заданном отрезке  $[a, b]$ :

- 1) составление программы табулирования функции  $f(x)$ ;
- 2) табулирование функции  $f(x)$  на отрезке  $[a, b]$  с шагом  $\Delta x = (b - a)/n$  (можно рекомендовать для начала принять  $n = 10$ );
- 3) построение графика функции  $f(x)$  и отделение всех корней уравнения  $f(x) = 0$  на отрезке  $[a, b]$ .

### 1.3. УТОЧНЕНИЕ КОРНЯ УРАВНЕНИЯ

Для ответа на вопрос, *каково* значение корня уравнения  $f(x) = 0$  на некотором отрезке  $[a, b]$ , на котором он (корень) отделен, отличающееся от точного не более чем на некоторую малую величину  $\varepsilon$ , необходимо каким-то образом найти это значение корня. Такая операция называется *уточнением* корня. Существует много численных методов уточнения корня нелинейного уравнения, из которых далее рассматриваются три: метод *половинного деления*, метод *итераций* и метод *Ньютона*.

#### 1.3.1. Метод половинного деления

Метод половинного деления заключается в последовательном уменьшении вдвое отрезка, на котором находится отделенный корень, до тех пор, пока величина уменьшенного отрезка не станет меньше допустимой погрешности (иногда говорят заданной точности)  $\varepsilon$ . Идея этого метода может быть проиллюстрирована схемой, показанной на рис. 1.2. Словесное описание алгоритма уточнения корня методом половинного деления выглядит так:

- 1) вычисляется и запоминается значение функции  $f(x)$  при  $x = a$ , т. е.  $f_a = f(a)$ ;
- 2) отрезок делится пополам, т. е. вычисляется  $x = (b - a)/2$ ;
- 3) вычисляется значение функции  $f(x)$  при  $x = (b - a)/2$ , т. е.  $f_x$ ;
- 4) проверяется условие  $f_a f_x > 0$ , т. е. имеет ли функция  $f(x)$  на левом конце отрезка и в его середине одинаковые знаки;
- 5) если это условие выполняется, то за левую границу нового отрезка принимается середина прежнего, и за значение функции

на левом конце отрезка принимается ранее вычисленное значение в середине прежнего (отрезка), т. е. производится *переприсваивание*:  $a = x, f_a = f_x$  (левый конец отрезка переносится в середину);

6) в противном случае (при невыполнении условия  $f_a \cdot f_x > 0$ ) в середину переносится правый конец отрезка, т. е.  $b = x$ ;

7) после очередного деления отрезка пополам проверяется, не стала ли оставшаяся часть отрезка меньше допустимой погрешности  $\varepsilon$ , т. е. проверяется условие  $b - a < \varepsilon$ ;

8) если это условие выполняется, то задача уточнения корня решена, и вычисления прекращаются. За уточненное значение корня  $\tilde{x}$  принимается последнее вычисленное значение  $x$  (середина последнего оставшегося отрезка). В случае невыполнения условия  $b - a < \varepsilon$  вычисления продолжаются, т. е. происходит переход к пункту 2) настоящего алгоритма.

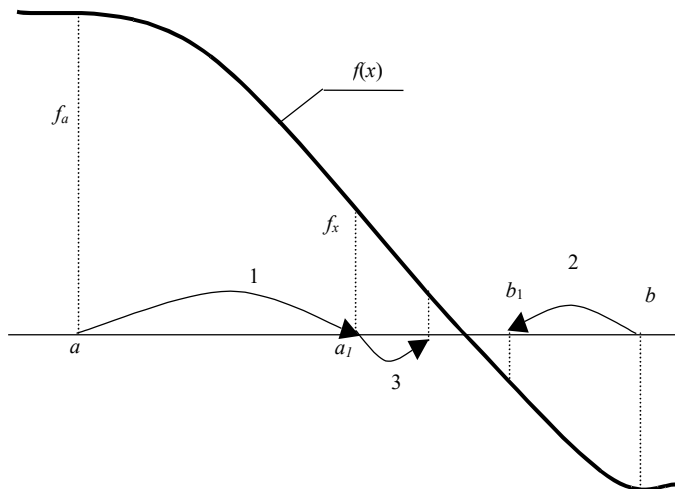


Рис. 1.2. Схема, иллюстрирующая метод половинного деления

Характеристикой качества решения уравнения является *невязка*, которая определяется величиной  $f(\tilde{x})$ , т. е. значением функции  $f(x)$  при аргументе, равном уточненному с заданной точностью  $\varepsilon$  значению корня. При уточнении корня методом половинного деления невязка равна  $f_x$  — последнему вычисленному значению функции  $f(x)$  при последнем делении отрезка.

Необходимое количество операций деления отрезка  $[a, b]$  пополам для уточнения корня с точностью  $\varepsilon$  определяется следующим образом. Так как при каждом делении отрезка длина его

уменьшается вдвое, то после  $n$  делений она будет равна  $(b-a)/2^n$ . Эта величина должна быть меньше  $\varepsilon$ , т. е. должно выполняться условие (неравенство)  $\varepsilon > (b-a)/2^n$ , или  $2^n > \frac{b-a}{\varepsilon}$ .

Отсюда

$$n \geq \log_2 \frac{b-a}{\varepsilon} \quad (1.2)$$

– количество делений отрезка пополам при уточнении корня методом половинного деления должно быть не меньше, чем логарифм по основанию два (двоичный логарифм) величины  $(b-a)/\varepsilon$ , где  $a$  и  $b$  – границы отрезка, на котором уточняется корень (на котором корень отделен),  $\varepsilon$  – допустимая погрешность (заданная точность). Так, при  $b-a = 1$  и  $\varepsilon = 0.001$ ,  $n = 10$ , так как  $(b-a)/\varepsilon = 1000$ , а  $2^{10} = 1024$ . Блок-схема алгоритма уточнения корня нелинейного уравнения методом половинного деления изображена на рис. 1.3.

В алгоритме, изображенном на блок-схеме, после каждого деления отрезка пополам и вычисления значения функции в точке деления предусмотрена проверка условия  $|f_i| < (\varepsilon/10)$ . Выполнение этого условия означает, что в середине отрезка невязка уравнения по абсолютной величине достаточно мала и дальнейшие вычисления можно прекратить.

В алгоритме также предусмотрена проверка правильности (корректности) исходных данных. Перед выводом результатов решения (уточнения корня) проверяется условие  $i = 0$  – не равно ли количество делений отрезка пополам нулю. Выполнение этого условия означает, что при вводе исходных данных допущена ошибка (чаще всего перепутаны границы отрезка, на котором отделен корень). В этом случае выводится сообщение на экране компьютера «Исходные данные неверны».

При реализации алгоритма на языке QBASIC можно рекомендовать использовать цикл с предусловием при проверке условия  $b-a < \varepsilon$ :

DO UNTIL  $b-a < \text{eps}$  – выполнить, пока не выполняется условие  $b-a < \text{eps}$ .

При правильных исходных данных в качестве результатов уточнения выводится количество делений отрезка пополам  $i$ , зна-

чение корня  $x$  и значение функции  $f_x$ , вычисленное при уточненном значении корня – невязка уравнения.

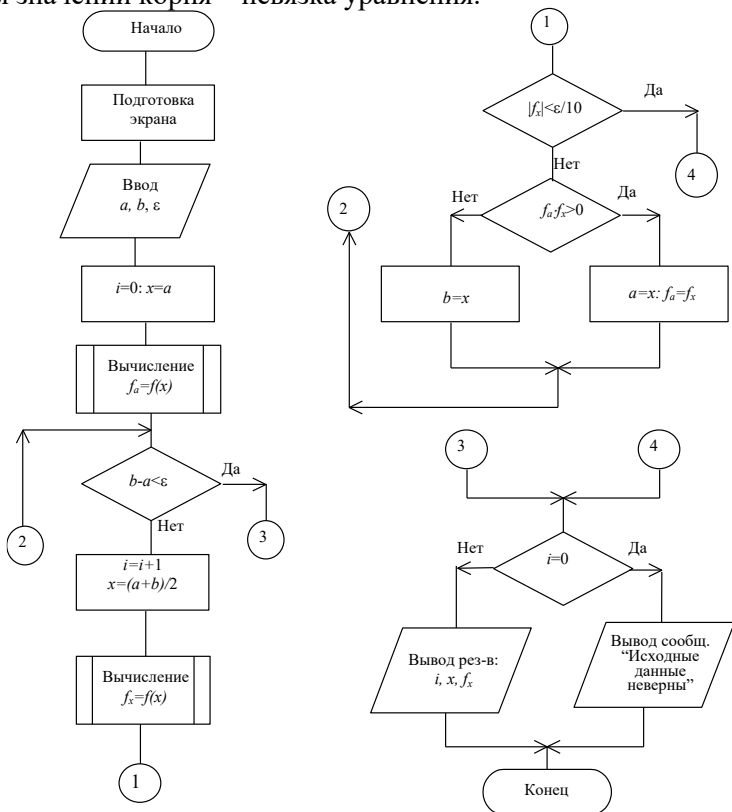


Рис. 1.3. Блок-схема алгоритма уточнения корня нелинейного уравнения методом половинного деления

В качестве примера в табл. 1.1 приведены результаты уточнения корня уравнения  $x^4 + 2x^3 - x - 1 = 0$  на отрезке  $[0, 1]$  с помощью программы, реализующей описанный выше алгоритм.

Т а б л и ц а 1.1

Результаты уточнения корня нелинейного уравнения методом половинного деления

№ п/п	$\varepsilon$	$i$	$x$	$f(x)$
1	$1 \cdot 10^{-2}$	7	0,8672	$2,612 \cdot 10^{-3}$

2	$1 \cdot 10^{-3}$	10	0,8662	$-3,356 \cdot 10^{-3}$
3	$1 \cdot 10^{-4}$	14	0,8668	$-8,879 \cdot 10^{-7}$
4	$1 \cdot 10^{-5}$	14	0,8668	$-8,879 \cdot 10^{-7}$

Из таблицы видно, что при  $\varepsilon = 1 \cdot 10^{-4}$  и  $1 \cdot 10^{-5}$  количество делений отрезка пополам одинаково, 14. Это как раз тот случай, когда при очередном, четырнадцатом, делении отрезка значение функции  $f(x)$  в середине отрезка (невязка уравнения) по абсолютной величине меньше допустимой погрешности  $\varepsilon$ , хотя условие  $b-a < \varepsilon$  и не выполняется.

### 1.3.2. Метод итераций

Итерация – это результат повторного применения совокупности математических (вычислительных) операций. Метод итераций – это метод последовательных приближений.

Основными этапами алгоритма уточнения корня методом итераций являются следующие:

- 1) уравнение  $f(x) = 0$  преобразовывают к виду  $x = \varphi(x)$ ;
- 2) на отрезке  $[a, b]$ , на котором отделен корень, выбирают произвольную точку  $x_0$  – начальное (нулевое) приближение корня;
- 3) последовательно вычисляют  $x_1 = \varphi(x_0)$ ,  $x_2 = \varphi(x_1)$ , ...,  $x_i = \varphi(x_{i-1})$  – очередные (последовательные) приближения корня уравнения;
- 4) после вычисления очередного приближения корня производят проверку условия прекращения итераций (как правило,  $|x_i - x_{i-1}| < \varepsilon$ );
- 5) если условие прекращения итераций не выполняется, то переходят к вычислению очередного приближения корня – пункту 3) данного алгоритма;
- 6) если условие прекращения итераций выполняется, то вычисляют невязку уравнения, выводят результаты на экран и прекращают вычисления.

Доказано, что если величина  $|x_i - x_{i-1}|$  уменьшается при увеличении  $i$ , то при  $i \rightarrow \infty$  разность  $x_i - \xi$  станет сколь угодно малой по абсолютной величине, т. е. уточнение корня нелинейного уравнения методом итераций с заданной точностью возможно [1]. Напомним, что  $\xi$  – точное значение корня.



Достаточным условием сходимости метода итераций является следующее:

$$|\varphi'(x)| \leq q < 1, \quad a < x < b \quad (1.3)$$

– первая производная функции  $\varphi(x)$  на отрезке  $[a, b]$  должна быть *меньше единицы* (по абсолютной величине). При невыполнении условия (1.3) процесс итерации может не сходиться, как показано на рис. 1.4.

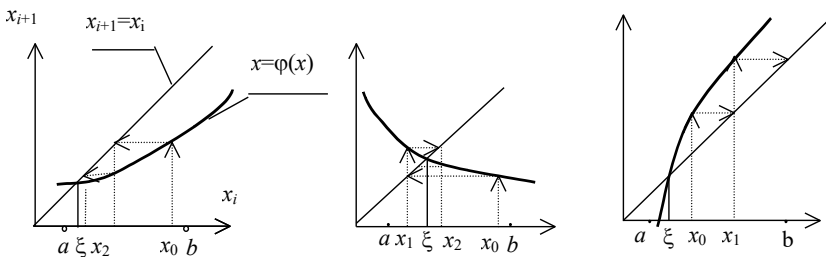


Рис. 1.4. Геометрическая иллюстрация метода итераций

Процесс итерации следует продолжать до тех пор, пока не выполнится условие

$$|x_i - x_{i-1}| < \frac{1-q}{q} \cdot \varepsilon, \quad (1.4)$$

где  $\varepsilon$  – заданная точность (допустимая погрешность);  $q$  – максимальное (по модулю) значение  $\varphi'(x)$  на отрезке  $[a, b]$ .

Если  $q \leq 0.5$ , то вместо условия (1.4) можно пользоваться условием

$$|x_i - x_{i-1}| < \varepsilon. \quad (1.5)$$

При использовании метода итераций уравнение  $f(x) = 0$  необходимо стараться преобразовать к виду  $x = \varphi(x)$  таким образом, чтобы условие (неравенство)  $|\varphi'(x)| < 1$  выполнялось на всем отрезке  $[a, b]$ . Например, уравнение  $x^3 + x - 1000 = 0$ , имеющее корень на отрезке  $[9, 10]$ , можно преобразовать двумя способами:

$$x = 1000 - x^3, \text{ т. е. } \varphi_1(x) = 1000 - x^3;$$

$$x = \sqrt[3]{1000 - x}, \text{ т. е. } \varphi_2(x) = \sqrt[3]{1000 - x}.$$

Совершенно очевидно, что для  $\varphi_1(x)$  условие (1.3) не выполняется, так как  $\varphi'_1(x) = -3x^2$  и  $|\varphi'_1(x)|$  при  $9 < x < 10$  много больше

единицы. В то же время  $|\varphi'_2(x)| = \frac{1}{3 \cdot \sqrt[3]{(1000 - x)^2}}$  и при  $9 < x < 10$

$|\varphi'_2(x)| < q \ll 1$  — производная  $\varphi'_2(x)$  по абсолютной величине *много меньше* единицы, следовательно, можно задавать довольно *большую* допустимую погрешность  $\varepsilon$ , что приведет к *уменьшению* количества итераций.

Уточнение корня данного уравнения методом итераций при  $x_0 = 9.5$ ,  $\varepsilon = 0.001$  и использовании функции  $\varphi_2(x) = \sqrt[3]{1000 - x}$  приводит к результату  $\tilde{x} = 9.967$  и получим  $f(\tilde{x}) = 1.1504 \cdot 10^{-4}$  после трех итераций.

### 1.3.3. Метод Ньютона

Метод Ньютона уточнения корня нелинейного уравнения  $f(x) = 0$  на отрезке  $[a, b]$  основан на следующих рассуждениях. Разложим функцию  $f(x)$  в ряд Тейлора в окрестности точки  $x_i$ , ограничившись двумя первыми членами разложения

$$f(x_i + \Delta x) = f(x_i) + \frac{\Delta x}{1!} \cdot f'(x_i). \quad (1.6)$$

Поправку (приращение)  $\Delta x$  следует выбрать такой, чтобы  $f(x_i + \Delta x) = f(x_{i+1})$  обратилась в нуль, следовательно,  $f(x_i) + \Delta x f'(x_i) = 0$ , откуда  $\Delta x = -\frac{f(x_i)}{f'(x_i)}$ . Таким образом, выражение

для вычисления очередного,  $i + 1$ -го, приближения корня примет следующий вид:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \quad (1.7)$$

Так же как и в методе итераций, алгоритм уточнения корня методом Ньютона сводится к выбору начального приближения

корня  $x_0$  и последовательному нахождению (вычислению) следующих приближений:

$$x_1 = x_0 - f(x_0)/f'(x_0), \quad x_2 = x_1 - f(x_1)/f'(x_1) \text{ и т.д.}$$

Процесс вычисления приближений (фактически процесс итераций) прекращают при выполнении условия [2]

$$|x_{i+1} - x_i| < \sqrt{\frac{2 \cdot m_1}{M_2}} \cdot \varepsilon, \quad (1.8)$$

где  $m_1$  – наименьшее значение  $|f''(x)|$  на отрезке  $[a, b]$ ;  $M_2$  – наибольшее значение  $|f''(x)|$  на отрезке  $[a, b]$ .

Начальное приближение  $x_0$  при использовании метода Ньютона должно удовлетворять следующему условию:

$$f(x_0) \cdot f''(x_0) > 0, \quad (1.9)$$

т. е. функция  $f(x_0)$  и ее вторая производная  $f''(x_0)$  в точке  $x_0$  должны иметь одинаковые знаки. Невыполнение этого требования (условия) может привести к тому, что решение не будет найдено (*несходимость* метода). Метод Ньютона иногда называют методом *касательных*.

Следует заметить, что метод Ньютона эффективен в тех случаях, когда значение модуля первой производной функции  $|f'(x)|$  близ к корню достаточно велико, т. е. график функции  $f(x)$  в окрестности данного корня имеет *большую крутизну*.

Для уточнения корня нелинейного уравнения методом Ньютона можно (и нужно) воспользоваться тем же алгоритмом (той же головной программой), что и для метода итераций, изменив только процедуру вычисления очередного приближения корня. Блок-схема алгоритма уточнения корня методом итераций (Ньютона) приведена на рисунке 1.5.

Для примера рассмотрим уравнение  $\sin x - x + 0.15 = 0$  на отрезке  $[0.5, 1.5]$ . Протабулировав функцию  $f(x) = \sin x - x + 0.15$  на заданном отрезке, найдем, что при  $x = 0.9$   $f(x) = 0.033$ , а при  $x = 1.0$   $f(x) = -0.009$ . Вторая производная  $f''(x) = -\cos x$  на всем отрезке отрицательна. Поэтому в качестве начального приближения примем  $x_0 = 1.0$ . После двух итераций при  $\varepsilon = 0.01$  получим  $\tilde{x} = 0.9811$  и  $f(\tilde{x}) = -4.23 \cdot 10^{-8}$ .

#### 1.4. ПРИМЕР РЕШЕНИЯ ИНЖЕНЕРНОЙ ЗАДАЧИ

При производстве электротехнологических установок конические обечайки изготавливаются из листового материала вальцовкой (гибкой). На рис. 1.6 изображены эскизы обечайки и заготовки для нее.

Исходными данными для расчета (определения) размеров заготовки являются:

$d$  – малый диаметр обечайки, измеряемый по внутренней кромке верхнего основания;

$D$  – большой диаметр обечайки, измеряемый по наружной кромке нижнего основания;

$H$  – высота обечайки;

$\delta$  – толщина листа.

Результатами решения задачи (расчета) должны быть:

$\alpha$  – угол развертки конуса, образованного средней линией сечения заготовки;

$r$  – малый радиус заготовки;

$R$  – большой радиус заготовки;

$L$  и  $h$  – габаритные размеры заготовки.

Из рис. 1.6 можно вывести формулы для определения искоемых размеров заготовки:

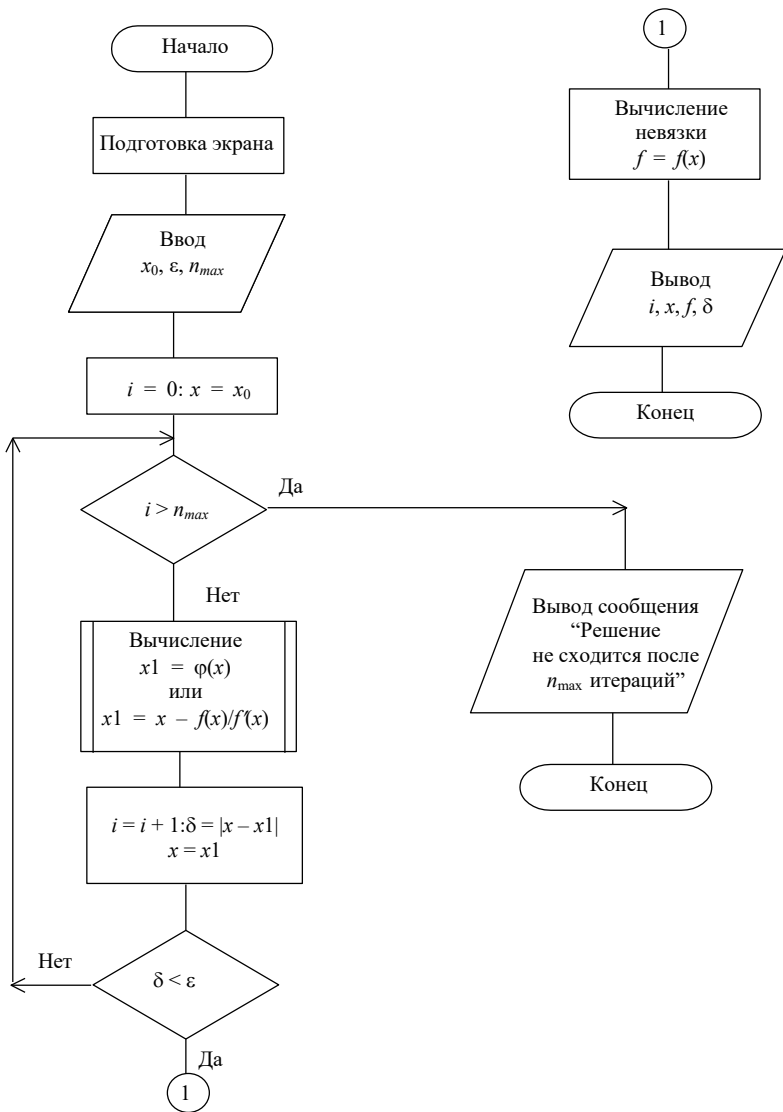


Рис. 1.5. Блок-схема алгоритма уточнения корня методом итераций (Ньютона)

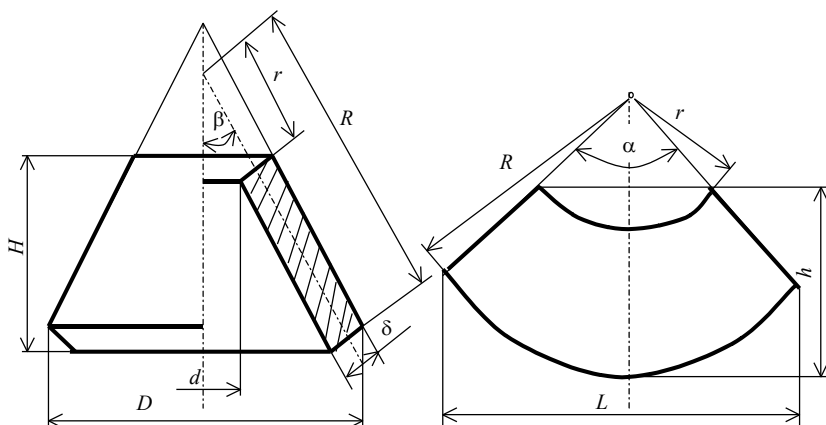


Рис. 1.6. Коническая обечайка и заготовка для нее

$$\left. \begin{aligned}
 r &= \frac{d + \delta \cdot \cos \beta}{2 \cdot \sin \beta}, \\
 R &= \frac{D - \delta \cdot \cos \beta}{2 \cdot \sin \beta}, \\
 \alpha &= 2 \cdot \pi \cdot \sin \beta, \\
 L &= \begin{cases} 2 \cdot R \cdot \sin \frac{\alpha}{2}, & \text{если } \alpha \leq \pi, \\ 2 \cdot R, & \text{если } \alpha > \pi; \end{cases} \\
 h &= \begin{cases} R - r \cdot \cos \frac{\alpha}{2}, & \text{если } \alpha \leq \pi, \\ R - R \cdot \cos \frac{\alpha}{2}, & \text{если } \alpha > \pi. \end{cases}
 \end{aligned} \right\} \quad (1.10)$$

Из формул (1.10) видно, что для определения размеров заготовки необходимо знать величину угла  $\beta$  – половины угла при вершине конуса обечайки. Из эскиза обечайки (рис. 1.6) можно записать уравнение

$$\beta = \arctg \frac{\frac{D-d}{2} - \delta \cdot \cos \beta}{H - \delta \cdot \sin \beta}. \quad (1.11)$$

Уравнение (1.11) не имеет аналитического решения, следовательно, его надо решать численно. Удобным в данном случае является метод итераций, так как уравнение представлено в виде  $\beta = \varphi(\beta)$ . Для определения начального приближения корня уравнения можно использовать выражение

$$\beta_0 = \frac{D-d}{H}. \quad (1.12)$$

Воспользовавшись программой расчета заготовки (см. приложение), получим следующие результаты (см. табл. 1.2).

Т а б л и ц а 1.2

**Результаты расчета заготовки для конической обечайки**

№ п/п	Исходные данные				Результаты расчета				
	$d$ , мм	$D$ , мм	$H$ , мм	$\delta$ , мм	$r$ , мм	$R$ , мм	$\alpha^\circ$	$L$ , мм	$h$ , мм
1	2500	3500	1000	10.0	2836.0	3946.1	159.2	7763.1	3435.2
2				20.0	2879.0	3981.2	157.4	7808.4	3417.8
3				30.0	2924.0	4018.5	155.6	7854.9	3399.7
4				40.0	2971.4	4058.2	153.6	7902.5	3380.6
5				50.0	3021.1	4100.4	151.7	7951.2	3360.6
6				100.0	3316.3	4360.2	140.7	8212.3	3244.7

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие уравнения называются нелинейными?
2. Какие уравнения называются трансцендентными?
3. Что понимается под отделением корня?
4. Как отделить корень уравнения  $f(x) = 0$ ?
5. Что понимается под уточнением корня?
6. Какие методы уточнения корня вы знаете?
7. В чем заключается идея уточнения корня методом половинного деления?

8. От чего зависит необходимое количество делений отрезка пополам?
9. Какова идея метода итераций?
10. Что такое итерация?
11. Как необходимо преобразовать уравнение для уточнения корня методом итераций?
12. Какие требования предъявляются к функции  $f(x)$  при уточнении корня методом итераций?
13. Чем отличается метод Ньютона от метода итераций?
14. Как вычисляются очередные приближения корней в методах итераций и Ньютона?
15. Как выбираются начальные приближения корня при использовании методов итераций и Ньютона?
16. Что такое невязка уравнения и как она вычисляется?
17. Что является критерием прекращения делений отрезка пополам при уточнении корня уравнения методом половинного деления?
18. Что является критерием прекращения итераций при уточнении корня уравнения методами итераций и Ньютона?

## 2. ИНТЕРПОЛИРОВАНИЕ ТАБЛИЧНО ЗАДАННЫХ ФУНКЦИЙ

### 2.1. ПОСТАНОВКА ЗАДАЧИ И ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Пусть некоторая функция  $y = f(x)$  задана таблицей (аналитическое выражение  $f(x)$  неизвестно)

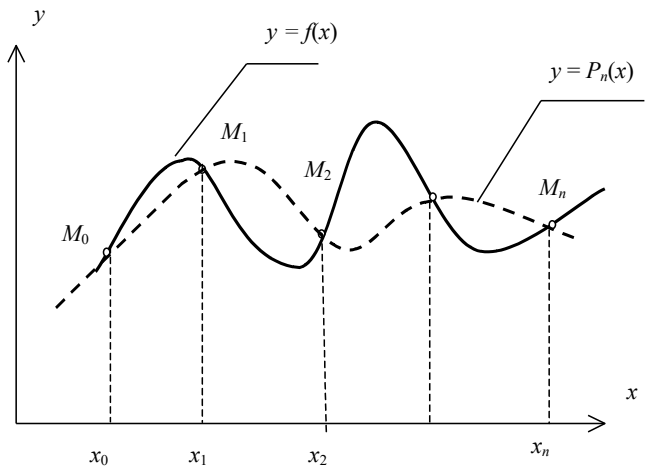
$i$	$x_i$	$y_i$
0	$x_0$	$y_0$
1	$x_1$	$y_1$
2	$x_2$	$y_2$
...	...	...
$n$	$x_n$	$y_n$

Задача *интерполирования* ставится в следующей форме: найти (сформировать) многочлен  $P(x)$  степени не выше  $n$ , значения которого в точках  $x_i$  ( $i = 0, 1, 2, \dots, n$ ) – узлах – совпадают со значениями таблично заданной функции, т. е.  $P_n(x_i) = y_i$ . Геометрически это означает, что нужно найти алгебраическую кривую ви-



да  $y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ , проходящую через заданную систему точек  $M_i(x_i, y_i)$  для  $i = 0, 1, \dots, n$  (рис. 2.1).

Рис. 2.1. Геометрическая иллюстрация задачи интерполирования



Вполне естественно, задача интерполирования не ограничивается формированием многочлена  $P_n(x)$ . Многочлен нужен для того, чтобы можно было находить (вычислять) с заданной точностью значения  $y(x) \approx P_n(x)$  в точках между  $x_i$  ( $i = 0, 1, \dots, n$ ), называемыми *узлами* интерполяции (функции). Узлы интерполяции называют *равноотстоящими*, если выполняется условие

$$\Delta x = x_{i+1} - x_i = \text{const}, \quad i = 0, 1, 2, \dots, n - 1. \tag{2.1}$$

*Конечными разностями* таблично заданной функции  $y = f(x)$  называют разности следующего вида:

- $\Delta y_i = y_{i+1} - y_i$  – конечные разности *первого* порядка;
- $\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$  – конечные разности *второго* порядка;
- .....
- $\Delta^n y_i = \Delta^{n-1} y_{i+1} - \Delta^{n-1} y_i$  – конечные разности *n-го* порядка.

Ниже приведена таблица конечных разностей для  $n = 5$  (табл. 2.1).

Т а б л и ц а 2.1

Таблица конечных разностей

$x$	$y$	$\Delta y$	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
$x_0$	$y_0$	$\Delta y_0 = y_1 - y_0$	$\Delta^2 y_0 = \Delta y_1 - \Delta y_0$	$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0$	$\Delta^4 y_0 = \Delta^3 y_2 - \Delta^3 y_1$	$\Delta^5 y_0 = \Delta^4 y_1 - \Delta^4 y_0$
$x_1$	$y_1$	$\Delta y_1 = y_2 - y_1$	$\Delta^2 y_1 = \Delta y_2 - \Delta y_1$	$\Delta^3 y_1 = \Delta^2 y_2 - \Delta^2 y_1$	$\Delta^4 y_1 = \Delta^3 y_2 - \Delta^3 y_1$	
$x_2$	$y_2$	$\Delta y_2 = y_3 - y_2$	$\Delta^2 y_2 = \Delta y_3 - \Delta y_2$	$\Delta^3 y_2 = \Delta^2 y_3 - \Delta^2 y_2$		
$x_3$	$y_3$	$\Delta y_3 = y_4 - y_3$	$\Delta^2 y_3 = \Delta y_4 - \Delta y_3$			
$x_4$	$y_4$	$\Delta y_4 = y_5 - y_4$				
$x_5$	$y_5$					

Практически задача интерполирования обычно ставится следующим образом: для таблично заданной  $n$  узлами функции разработать алгоритм и соответствующую программу, позволяющую находить (вычислять) значения функции в промежутках между узлами, т. е. для  $x_i \leq x \leq x_{i+1}$  ( $i = 0, 1, \dots, n-1$ ), с помощью интерполяционного многочлена порядка (степени)  $m \leq n$ , т. е.  $y(x) = P_m(x)$ .

Для инженерных расчетов порядок многочлена  $m$  обычно не превышает *трех*. При  $m = 1$  интерполяцию называют *линейной*, при  $m = 2$  – *квадратичной*, при  $m = 3$  – *кубической*.

## 2.2. ИНТЕРПОЛИРОВАНИЕ ФУНКЦИЙ С РАВНООТСТОЯЩИМИ УЗЛАМИ

Для интерполирования таблично заданных функций с *равноотстоящими* узлами чаще всего применяют интерполяционные формулы Ньютона. *Первая* интерполяционная формула Ньютона имеет вид

$$y(x) = P_m(x) = y_0 + q \cdot \Delta y_0 + \frac{q \cdot (q-1)}{2!} \cdot \Delta^2 y_0 + \dots + \frac{q \cdot (q-1) \cdot \dots \cdot (q-m+1)}{m!} \Delta^m y_0, \quad (2.2)$$

где  $x_0 \leq x \leq x_1$  – заданное значение аргумента, для которого вычисляется значение функции;  $m$  – порядок (степень) интерполяционного многочлена;  $\Delta x = x_1 - x_0$  – шаг изменения аргумента таблично заданной функции;  $q = \frac{x - x_0}{\Delta x}$ .

В формуле (2.2) используется верхняя горизонтальная строка таблицы (2.1). При наличии дополнительного узла  $x_{m+1}$  остаточ-

ный член  $R_m(x)$  формулы, характеризующий погрешность интерполирования, может быть вычислен по следующему выражению [3]:

$$R_m(x) = \frac{\Delta^{m+1} y_0}{(m+1)!} \cdot q \cdot (q-1) \cdot \dots \cdot (q-m). \quad (2.3)$$

Первая интерполяционная формула Ньютона используется для интерполирования в точках, близких к началу таблицы (интерполирование *вперед*). При  $m = 1$  из (2.2) получается формула *линейного* интерполирования (линейной интерполяции)

$$y(x) = y_0 + q \cdot \Delta y_0 = \frac{x - x_0}{x_1 - x_0} \cdot (y_1 - y_0). \quad (2.4)$$

Линейная интерполяция наглядно иллюстрируется рис. 2.2.

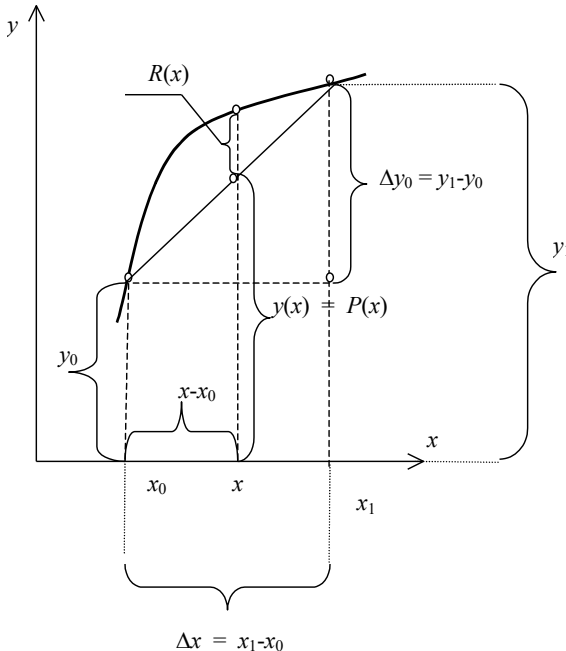


Рис. 2.2. Геометрическая иллюстрация формулы линейного интерполирования

При  $m = 2$  получается формула *квадратичной* интерполяции

$$y(x) = y_0 + q \cdot \Delta y_0 + \frac{q(q-1)}{2} \Delta^2 y_0. \quad (2.5)$$

Вторая интерполяционная формула Ньютона имеет вид

$$y(x) = y_m + q \cdot \Delta y_{m-1} + \frac{q(q+1)}{2!} \Delta^2 y_{m-2} + \dots, \quad (2.6)$$

$$\dots + \frac{q(q+1) \cdot \dots \cdot (q+m-1)}{m!} \cdot \Delta^m y_0$$

где  $x_{m-1} \leq x \leq x_m$  – заданное значение аргумента, для которого вычисляется значение функции;  $\Delta x = x_m - x_{m-1}$  – шаг изменения аргумента таблично заданной функции;

$$q = \frac{x - x_m}{\Delta x}.$$

В формуле (2.6) используется нижняя наклонная (диагональная) строка табл. 2.1 конечных разностей. Формула (2.6) используется для интерполирования в точках, лежащих между узлами  $x_m$  и  $x_{m-1}$  – производится интерполирование *назад*. Следует заметить, что для интерполирования многочленом порядка  $m$  необходимо  $m + 1$  узлов:  $x_0, x_1, \dots, x_{m-1}, x_m$ .

### 2.3. ПРИМЕР ИСПОЛЬЗОВАНИЯ ФОРМУЛ НЬЮТОНА

Рассмотрим следующую задачу. Характеристика некоторого нелинейного электрического сопротивления задана табл. 2.2,а). Из нее видно, что функция  $R = f(I)$  задана таблицей с равноотстоящими узлами. Задача состоит в том, чтобы разработать алгоритм и программу для компьютера, с помощью которой можно было бы вычислять значение сопротивления  $R$  для любого задаваемого значения  $I$  (в пределах таблицы) в промежутках между узлами, используя интерполяционные многочлены Ньютона первого, второго и третьего порядков.

Т а б л и ц а 2.2

Таблично заданные функции

а) Характеристика нелинейного сопротивления			б) Вольт-амперная характеристика полого катода		
$N_{\text{п/п}}$	$I$	$R$	$N_{\text{п/п}}$	$U$	$I$
1	1.0	0.5752	1	-3.05	-75.00
2	1.1	0.6475	2	-2.05	-74.50
3	1.2	0.7247	3	-0.90	-70.30
4	1.3	0.8073	4	0.35	-66.70
5	1.4	0.8961	5	1.10	-55.00
6	1.5	0.9917	6	2.00	-20.00
7	1.6	1.0950	7	3.00	5.00
8	1.7	1.2060	8	4.10	17.50
9	1.8	1.3270	9	5.00	35.80
10	1.9	1.4580	10	6.00	38.00
11	2.0	1.6010			

Исходными данными для решения задачи являются:

- $n$  – количество узлов таблично заданной функции;
- $m = 1, 2, 3$  – порядок (степень) интерполяционного многочлена;
- $I_s$  – заданное значение тока (аргумента), для которого необходимо найти (вычислить) значение сопротивления (функции) т. е.  $R(I_s)$ .

Результатами работы программы должны быть значения  $R(I_s)$ , полученные с помощью интерполяционных многочленов Ньютона первой, второй и третьей степени. Желательно эти результаты свести в таблицу.

Алгоритм решения данной задачи должен состоять из следующих основных операций:

- 1) ввод количества узлов таблично заданной функции  $n$ ;
- 2) описание массивов  $I(n)$ ,  $R(n)$ ,  $x(3)$ ,  $y(3)$ ;
- 3) формирование исходных массивов  $I(n)$ ,  $R(n)$ ;
- 4) вывод исходной таблицы (в левой части экрана);
- 5) вывод названия и шапки (головки) таблицы результатов (в правой части экрана);
- 6) вычисление  $\Delta x = I_2 - I_1$  (вообще  $\Delta x = I_{j+1} - I_j$ ,  $j = 1, 2, \dots, n-1$ );
- 7) ввод заданного значения  $I_s$ , для которого необходимо вычислить  $R(I_s)$ ;
- 8) проверка, входит ли  $I_s$  в пределы таблично заданной функции, т. е. выполняется ли условие  $I_1 \leq I_s \leq I_n$ . Если это условие не выполняется, то программа прекращает работу;

- 9) вычисление  $R(I_s)$ ;
- 10) вывод результатов вычисления;
- 11) переход к пункту 7) данного алгоритма.

Алгоритм вычисления  $R(I_s)$  выглядит следующим образом:

1) поиск в массиве  $I$  промежутка, в который входит  $I_s$ , т. е. определение номера узла  $j_0$ , для которого выполняется условие  $I_{j_0} \leq I_s \leq I_{j_0+1}$ ;

2) присваивание  $x = I_s$ ;

3) организация цикла для  $m = 1(1)3$ ;

4) проверка условия  $j_0 \leq n - m$ . Если это условие выполняется, то вычисление  $R(I_s)$  по первой интерполяционной формуле Ньютона, в противном случае – по второй.

Вычисление  $R(I_s)$  по первой интерполяционной формуле Ньютона производится в следующем порядке:

1) формирование массивов  $x$  и  $y$  длиной  $m$  ( $x_0 = I_{j_0}$ ,  $y_0 = R_{j_0}$ ,  $x_1 = I_{j_0+1}$ ,  $y_1 = R_{j_0+1}$ , ...,  $x_m = I_{j_0+m}$ ,  $y_m = R_{j_0+m}$ );

2) вычисление и формирование таблицы (матрицы) конечных разностей;

3) вычисление  $q = (x - x_0)/\Delta x$ ;

4) вычисление  $R(I_s) = P_m(x)$  по первой интерполяционной формуле Ньютона.

Вычисление  $R(I_s)$  по второй интерполяционной формуле Ньютона производится в следующем порядке:

1) формирование массивов  $x$  и  $y$  ( $x_m = I_{j_0+1}$ ,  $y_m = R_{j_0+1}$ ,  $x_{m-1} = I_{j_0}$ ,  $y_{m-1} = R_{j_0}$ , ...,  $x_0 = I_{j_0-m+1}$ ,  $y_0 = R_{j_0-m+1}$ );

2) вычисление и формирование таблицы (матрицы) конечных разностей;

3) вычисление  $q = (x - x_m)/\Delta x$ ;

4) вычисление  $R(I_s) = P_m(x)$  по второй интерполяционной формуле Ньютона.

На рис. 2.3 изображена блок-схема алгоритма решения задачи в целом (головной программы), а на рис. 2.4 – блок-схемы алгоритмов процедур вычисления по первой и второй интерполяционным формулам Ньютона (NWT1 и NWT2), процедуры формирования таблицы конечных разностей (FRMTab) и процедуры определения  $j_0$  (Searchj0).

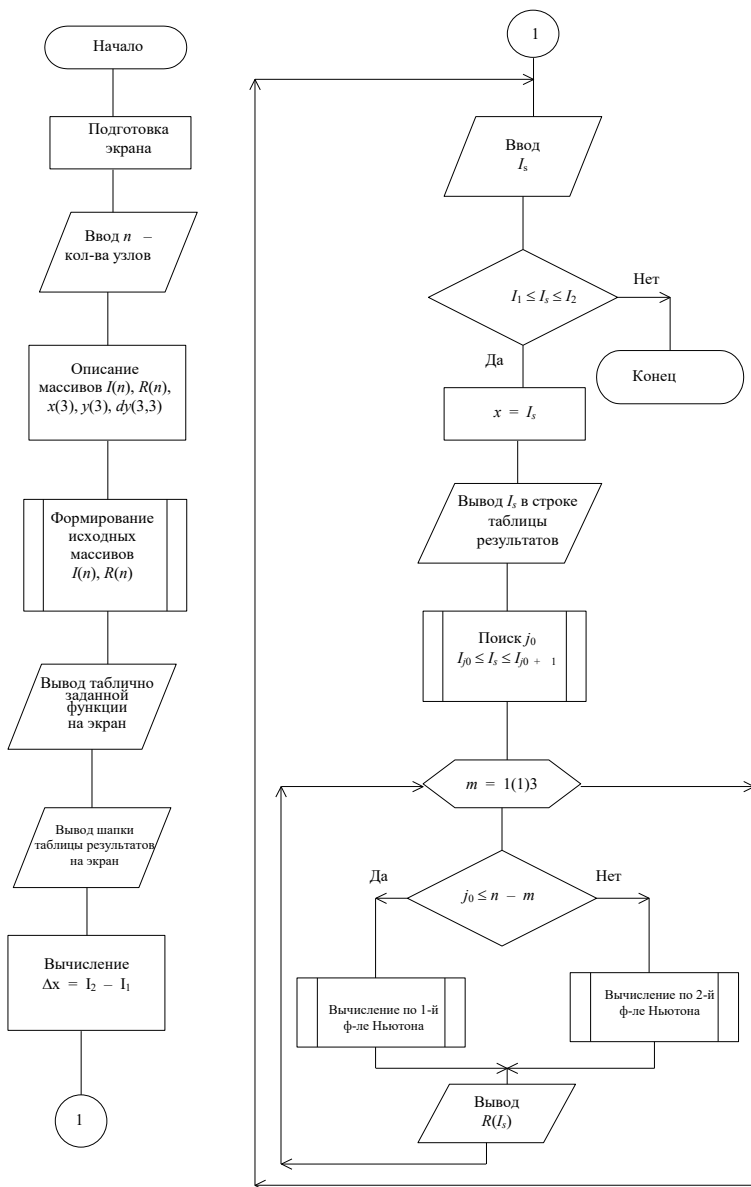
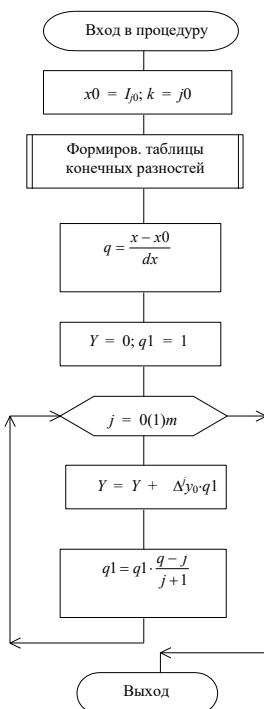


Рис. 2.3. Блок-схема алгоритма решения задачи интерполирования таблично заданной функции с равноотстоящими узлами с помощью формул Ньютона

NWT1(x, j0, m, dx, Y)



NWT2(x, j0, m, dx, Y)

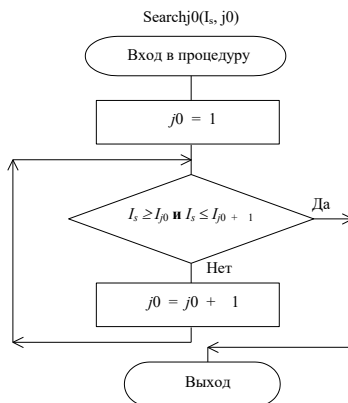
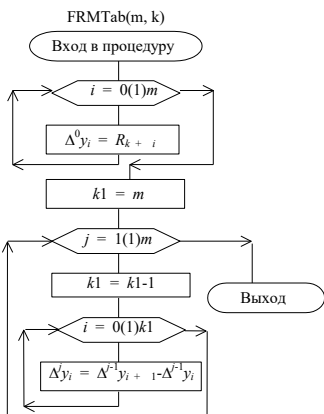
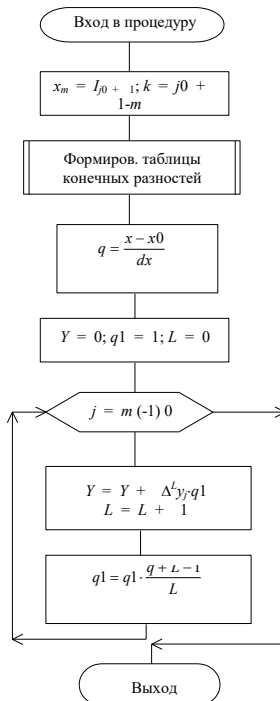


Рис.2.4. Блок-схемы алгоритмов процедур для интерполирования таблично заданной функции с равноотстоящими узлами



Таблица конечных разностей (матрица, иначе говоря, двумерный массив) формируется по столбцам. Первый столбец – *узловые значения* таблично заданной функции, т. е.  $y_0, y_1, \dots, y_m$ . Каждый элемент последующего столбца определяется вычитанием (разностью) двух последовательных элементов предыдущего. Количество элементов каждого последующего столбца на единицу меньше количества элементов предыдущего.

В скобках после имен процедур указаны параметры, которые передаются процедуре при обращении к ней и имена переменных, значения которых должны быть получены при выходе из процедуры.

Результатом работы процедуры FRMTab является таблица конечных разностей – двумерный массив, который в программе описан как глобальный, т. е. доступный всем процедурам программы.

В табл. 2.3 даны результаты интерполирования таблично заданной функции из табл. 2.2,*a*).

Т а б л и ц а 2.3

**Результаты интерполирования таблично заданной функции с равноотстоящими узлами**

$I_s$	$R(I_s)$		
	$m = 1$	$m = 2$	$m = 3$
1.050	0.6113	0.6107	0.6108
1.1170	0.7015	0.7010	0.7010
1.380	0.8783	0.8778	0.8778
1.750	1.2665	1.2652	1.2654
1.870	1.4187	1.4174	1.4176
1.980	1.5724	1.5714	1.5713

#### **2.4. ИНТЕРПОЛИРОВАНИЕ ТАБЛИЧНО ЗАДАННЫХ ФУНКЦИЙ С НЕРАВНООТСТОЯЩИМИ УЗЛАМИ**

Для интерполирования функций с неравноотстоящими узлами применяют интерполяционную формулу Лагранжа [3]

$$L_m(x) = \sum_{i=0}^m y_i \cdot \frac{(x-x_1) \cdot (x-x_2) \cdot \dots \cdot (x-x_{i-1}) \cdot (x-x_{i+1}) \cdot \dots \cdot (x-x_m)}{(x_i-x_0) \cdot (x_i-x_1) \cdot \dots \cdot (x_i-x_{i-1}) \cdot (x_i-x_{i+1}) \cdot \dots \cdot (x_i-x_m)}, \quad (2.7)$$

где  $m$  – порядок (степень) многочлена;  $x$  – значение аргумента, для которого вычисляется значение функции;  $x_0, x_1, \dots, x_m$  – значения аргумента в узлах функции;  $x_i$  – значение аргумента в текущем  $i$ -м узле;  $y_i$  – значение функции в  $i$ -м узле.

Значение  $x$  должно удовлетворять условию  $x_0 \leq x \leq x_m$ .

Более компактная (сокращенная) запись формулы Лагранжа выглядит так

$$L_m(x) = \sum_{i=0}^m y_i \cdot \prod_{\substack{j=0 \\ j \neq i}}^m \frac{x - x_j}{x_i - x_j}. \quad (2.8)$$

В формуле (2.8) знак  $\Pi$  означает произведение  $m$  сомножителей вида  $(x - x_j)/(x_i - x_j)$  для фиксированного  $i$  и всех  $j$  от 0 до  $m$ , кроме случая, когда  $j = i$ .

Рассмотрим следующую задачу. Вольт-амперная характеристика полого катода, полученная в результате эксперимента, представлена табл. 2.2,б). Из нее видно, что функция  $I(U)$  задана таблицей с *неравноотстоящими* узлами. Задача состоит в том, чтобы разработать алгоритм и программу для компьютера, с помощью которой можно было бы вычислять значения тока  $I$  для любого задаваемого значения напряжения  $U$  (в пределах таблицы) в промежутках между узлами, используя интерполяционный многочлен Лагранжа первого, второго и третьего порядка.

Исходными данными для решения задачи являются:

$n = 10$  – количество узлов таблично заданной функции;

$m = 1, 2, 3$  – порядок (степень) интерполяционного многочлена Лагранжа;

$U_s$  = задаваемое значение аргумента, для которого вычисляется значение  $I(U)$ .

Результатами работы программы должны быть значения тока  $I(U_s)$ , полученные с помощью интерполяционного многочлена Лагранжа первой, второй и третьей степени (порядка). Результаты должны быть сведены в таблицу. Форму таблицы результатов примем аналогичной предыдущему примеру.

Алгоритм интерполирования с помощью формулы Лагранжа может быть представлен следующими основными этапами:

- 1) ввод количества узлов  $n$ ;
- 2) описание массивов  $U(n), I(n), x(3), y(3)$ ;
- 3) формирование массивов  $U(n), I(n)$ ;
- 4) вывод исходной таблицы (в левой части экрана);

- 5) вывод названия и шапки (головки) таблицы результатов;
- 6) ввод заданного значения  $U_s$ , для которого необходимо вычислить  $I(U_s)$  и вывод его в таблицу результатов;
- 7) проверка условия  $U_1 \leq U_s \leq U_n$ . Если это условие не выполняется, то работа программы прекращается;
- 8) организация цикла для  $m = 1(1)3$ ;
- 9) поиск в массиве  $U$  промежутка, в который входит  $U_s$ , т. е. определение номера (индекса)  $j_0$ , для которого выполняется условие  $U_{j_0} \leq U_s \leq U_{j_0+1}$ ;
- 10) проверка условия  $j_0 \leq n - m$ . Если это условие не выполняется, то принимается  $j_0 = n - m$ ;
- 11) формирование массивов  $x(m)$  и  $y(m)$  для вычислений по формуле Лагранжа:  $x_0 = U_{j_0}$ ,  $y_0 = I_{j_0}$ ,  $x_1 = U_{j_0+1}$ ,  $y_1 = I_{j_0+1}$ , ...,  $x_m = U_{j_0+m}$ ,  $y_m = I_{j_0+m}$ ;
- 12) вычисление  $I(U_s)$  по формуле Лагранжа и вывод результата;
- 13) переход к пункту 6 данного алгоритма.

На рис. 2.6 приведена блок-схема алгоритма решения задачи, на рис. 2.7 – блок-схема алгоритма процедуры вычисления значения многочлена Лагранжа. Процедуры формирования и вывода исходных массивов, поиска промежутка, в который укладывается  $U_s$  (определения  $j_0$ ), вывода шапки таблицы результатов могут быть заимствованы (с некоторыми изменениями) из предыдущей задачи. В табл. 2.4 приведены результаты интерполирования таблично заданной функции из табл. 2.2,б).

Т а б л и ц а 2.4

Результаты интерполирования таблично заданной функции с неравноотстоящими узлами

$U_s$	$I(U_s)$		
	$m = 1$	$m = 2$	$m = 3$
- 2.0	- 74.32	- 74.30	- 74.03
0.0	- 67.71	- 69.71	- 68.78
1.0	- 56.56	- 57.48	- 58.00
2.5	- 7.50	- 5.88	- 4.41
4.5	25.63	27.54	26.14
5.8	37.56	39.09	40.36

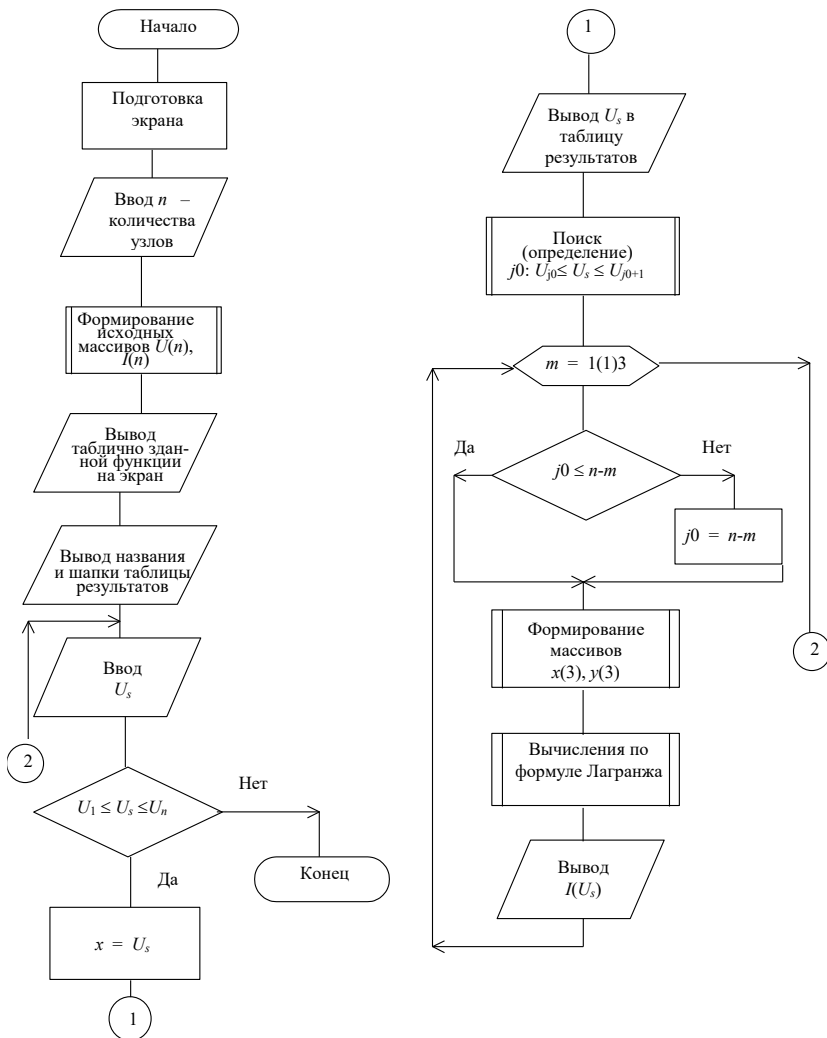


Рис. 2.6. Блок-схема алгоритма решения задачи интерполирования таблично заданной функции с неравноотстоящими узлами с помощью формулы Лагранжа

Lagrn(x, m, x(), y(), L)

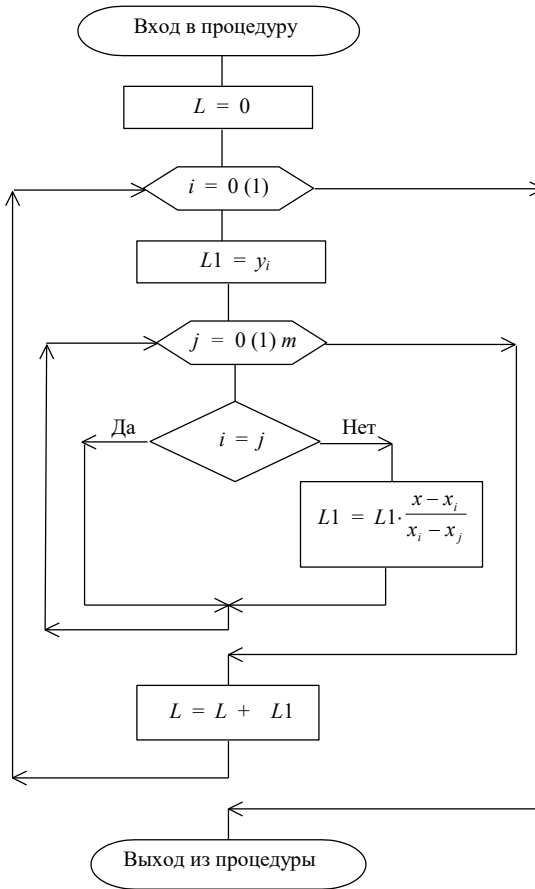


Рис. 2.7. Блок-схема алгоритма процедуры вычисления значения многочлена Лагранжа

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под таблично заданной функцией?
2. Что понимается под узлом таблично заданной функции?
3. Что является признаком таблично заданной функции с равноотстоящими узлами?
4. Что является признаком таблично заданной функции с неравноотстоящими узлами?
5. Что является задачей интерполирования таблично заданной функции?
6. Как производится линейное интерполирование?
7. Для интерполирования каких таблично заданных функций используются интерполяционные формулы Ньютона?
8. В каких случаях используется первая интерполяционная формула Ньютона?
9. В каких случаях используется вторая интерполяционная формула Ньютона?
10. Для каких таблично заданных функций используется интерполяционная формула Лагранжа?
11. Объясните работу процедуры, реализующей вычисления по формуле Лагранжа.
12. Можно ли использовать формулу Лагранжа для интерполирования таблично заданных функций с равноотстоящими узлами?

## 3. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ ИНТЕГРАЛОВ

### 3.1. ПОСТАНОВКА ЗАДАЧИ И ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

При решении научных и инженерных задач часто приходится вычислять значения определенных интегралов. Пусть необходимо найти

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

где  $I$  – значение определенного интеграла;  $a$  – нижний предел интегрирования (начало или левый конец отрезка интегрирования);  $b$  – верхний предел интегрирования (правый конец отрезка интегрирования);  $f(x)$  – подынтегральная функция.

Из математического анализа известна формула

$$\int_a^b f(x)dx = F(b) - F(a), \quad (3.2)$$

где  $F(x)$  – первообразная для  $f(x)$  функция в интервале, включающем  $[a, b]$ .

Однако зачастую выразить первообразную функцию через элементарные функции *не удастся* и приходится определенный интеграл вычислять приближенно. Существует много численных методов решения этой задачи. Мы рассмотрим два из них: метод *трапеций* и метод *Симпсона* [1].

### 3.2. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННОГО ИНТЕГРАЛА МЕТОДОМ ТРАПЕЦИЙ

Как известно, значение определенного интеграла численно равно площади криволинейной трапеции, ограниченной графиком подынтегральной функции  $f(x)$ , осью абсцисс и двумя прямыми  $x = a$  и  $x = b$  (рис. 3.1).

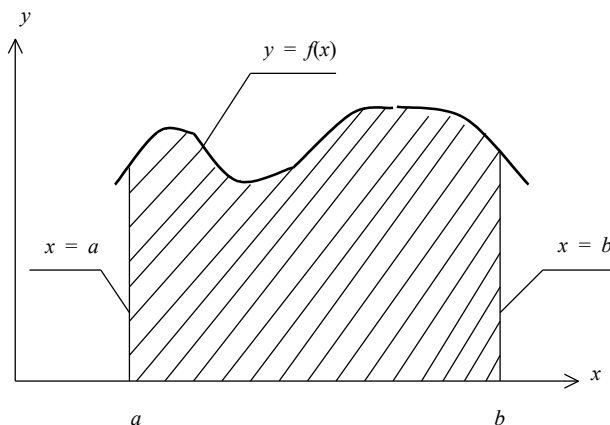


Рис. 3.1. Геометрическая иллюстрация задачи вычисления определенного интеграла

Разобьем отрезок интегрирования  $[a, b]$  на  $n$  равных частей длиной  $\Delta x = (b - a)/n$ . В точках  $x_0 = a, x_1 = x_0 + \Delta x, \dots, x_n = b$ , проведем прямые, параллельные оси  $Oy$ , до пересечения с кривой  $y = f(x)$  (графиком подынтегральной функции) в точках  $y_0, y_1, \dots, y_n$ . Очевидно, что  $y_0 = f(x_0), y_1 = f(x_1), \dots, y_n = f(x_n)$ . Соединим эти точки прямолинейными отрезками. Тогда площадь криволинейной трапеции  $ay_0 \dots y_nb$  будет приближенно равна площади фигуры, ограниченной ломаной линией  $ay_0y_1y_2 \dots y_{n-1}y_nb$ . Площадь этой фигуры  $S$  равна сумме площадей трапеций (рис. 3.2):

$$\begin{aligned} S &= \Delta x \left( \frac{y_0 + y_1}{2} + \frac{y_1 + y_2}{2} + \dots + \frac{y_{n-1} + y_n}{2} \right) = \\ &= \frac{b-a}{2n} (y_0 + 2y_1 + 2y_2 + \dots + 2y_{n-1} + y_n). \end{aligned}$$

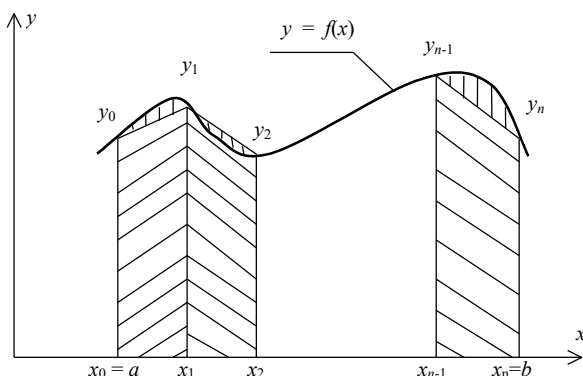


Рис. 3.2. Геометрическая иллюстрация метода трапеций

Таким образом, формула трапеций для вычисления приближенного значения определенного интеграла имеет следующий вид:

$$I = \int_a^b f(x) \cdot dx \approx \frac{b-a}{2 \cdot n} (y_0 + 2y_1 + \dots + 2y_{n-1} + y_n). \quad (3.3)$$

Остаточный член (погрешность) формулы трапеций определяется выражением [1]:



$$R_{mp} = -\frac{h^3}{12} f''(\xi), \quad (3.4)$$

где  $h = \Delta x$ ,  $\xi \in [a, b]$ .

Погрешность метода (формулы) трапеций пропорциональна кубу (третьей степени) шага  $h$  и значению второй производной подынтегральной функции в некоторой (к сожалению, неизвестно точно, какой) точке  $\xi$ , принадлежащей отрезку  $[a, b]$ . Формула трапеций дает точное значение интеграла, когда подынтегральная функция линейна, ибо в этом случае  $f''(x) = 0$ . Уменьшить погрешность вычисления значения определенного интеграла по формуле трапеций можно (как правило) уменьшением шага интегрирования  $h = \Delta x$ , т. е. увеличением  $n$  – числа разбиений отрезка интегрирования.

Алгоритм вычисления определенного интеграла методом трапеций включает в себя следующие основные операции:

- 1) ввод значений границ отрезка  $a$  и  $b$ ;
- 2) ввод числа разбиений отрезка  $n$ ;
- 3) вычисление шага интегрирования  $h = (b - a)/n$ ;
- 4) вычисление  $S = f(a) + f(b)$  – суммы значений подынтегральной функции на концах отрезка;
- 5) организация цикла для  $j = 1(1) n - 1$ ;
- 6) вычисление  $x = x_0 + h \cdot j$ ;
- 7) вычисление  $f(x)$  – значения подынтегральной функции;
- 8) вычисление  $S = S + 2 \cdot f(x)$ ;
- 9) после выхода из цикла (вычисления суммы  $n + 1$  слагаемых в скобке формулы трапеций) вычисляется значение интеграла  $I = S \cdot h/2$ ;
- 10) вывод значений  $a, b, n, I$ ;
- 11) переход к п. 2), если вычисления надо повторить с другим значением  $n$ . В противном случае – прекращение вычислений.

Блок-схема алгоритма вычисления определенного интеграла методом трапеций приведена на рис. 3.3.

Вычисление определенного интеграла  $I = \int_{0.6}^{1.4} \frac{\cos x}{1+x} \cdot dx$  с помощью программы, реализующей данный алгоритм при  $n = 8$  и  $n = 16$  дает следующие результаты:  $I_8 = 0.224618$ ,  $I_{16} = 0.223149$ .

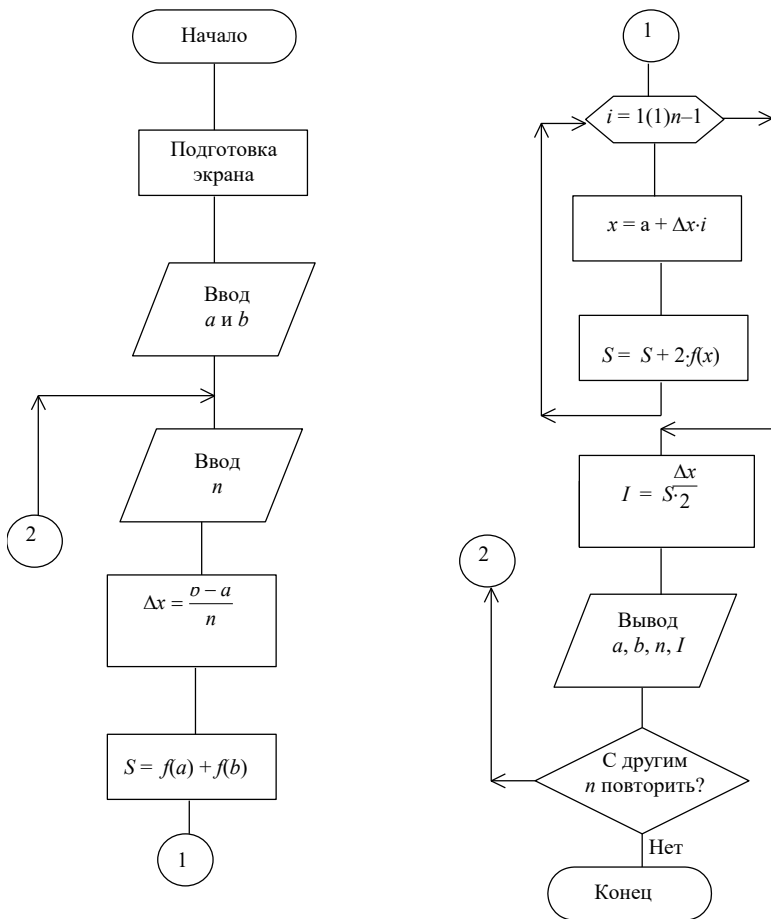


Рис. 3.3. Блок-схема алгоритма вычисления определенного интеграла методом трапеций

### 3.3. ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННОГО ИНТЕГРАЛА МЕТОДОМ СИМПСОНА

Идея метода Симпсона заключается в том, что участок графика подынтегральной функции между тремя последовательными точками  $y_{j-1}$ ,  $y_j$ ,  $y_{j+1}$  заменяют *параболой* (кривой второго порядка), проходящей через эти точки. Уравнение параболы записывают в виде интерполяционного многочлена Ньютона

$$f(x) = y_0 + \Delta y_0 \cdot q + \Delta^2 y_0 \cdot \frac{q \cdot (q-1)}{2}, \quad (3.5)$$

где  $q = (x-x_0)/\Delta x$ ,  $\Delta y_0 = y_1 - y_0$ ,  $\Delta y_1 = y_2 - y_1$ ,  $\Delta^2 y_0 = \Delta y_1 - \Delta y_0 = y_2 - 2 \cdot y_1 + y_0$ .

Интегрируя выражение (3.5) на отрезках  $[x_0, x_2]$ ,  $[x_2, x_4]$ , ...,  $[x_{2n-2}, x_{2n}]$  и суммируя результаты, получим формулу Симпсона

$$I = \int_a^b f(x) dx \approx \frac{b-a}{2 \cdot n \cdot 3} (y_0 + 4y_1 + 2y_2 + \dots + 2y_{2n-2} + 4y_{2n-1} + y_{2n}). \quad (3.6)$$

Остаточный член (погрешность) формулы Симпсона

$$R_c = -\frac{h^5}{90} f^{IV}(\xi), \quad (3.7)$$

где  $h = \Delta x = \frac{b-a}{2 \cdot n}$ ,  $\xi \in [a, b]$  – погрешность метода (формулы

Симпсона пропорциональна пятой степени шага интегрирования  $h$  и четвертой производной подынтегральной функции в некоторой точке  $\xi$  (кси), принадлежащей отрезку  $[a, b]$ . Формула Симпсона является точной для подынтегральной функции в виде алгебраического многочлена до третьей степени включительно, так как в этом случае  $f^{IV}(x) = 0$ .

При разработке алгоритма вычисления интеграла методом Симпсона необходимо учесть, что количество слагаемых в скобке (без  $y_0$  и  $y_{2n}$ ) равно  $2n - 1$ . Каждое вычисляемое значение подынтегральной функции  $y_i = f(x_i)$  для  $i = 1, 2, \dots, 2n - 1$  необходимо умножать либо на 4, либо на 2. Для этого вводится переменный множитель  $m$ . При ограниченной точности вычислителя (микропроцессора компьютера) очередные значения аргумента предпочтительнее вычислять как  $x_i = x_0 + \Delta x \cdot i$ , нежели как  $x_i = x_{i-1} + \Delta x$ , когда погрешность накапливается и может значительно снизить точность конечного результата. На рис. 3.4 изображена блок-схема алгоритма вычисления значения определенного интеграла методом Симпсона.

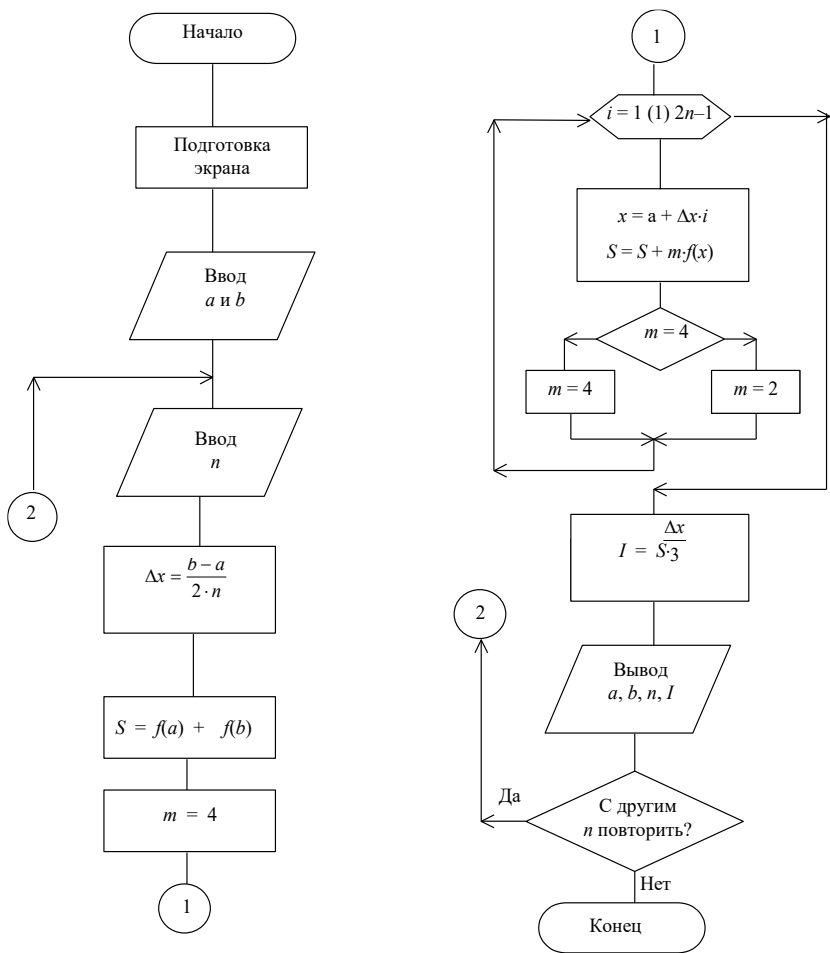


Рис. 3.4. Блок-схема алгоритма вычисления значения определенного интеграла методом Симпсона

Вычисление определенного интеграла  $I = \int_{0.6}^{1.4} \frac{\cos x}{1+x} \cdot dx$  при  $n = 5$  и  $n = 10$  с помощью программы, реализующей данный алгоритм, дает следующие результаты:  $I_5 = 0.222266$ ,  $I_{10} = 0.222266$ . Оценка погрешности  $\delta = |I_n - I_{2n}|$  для метода Симпсона при  $n = 5$  дает нулевой результат, что говорит о высокой точности метода.

### 3.4. ОБЕСПЕЧЕНИЕ ЗАДАННОЙ ТОЧНОСТИ

Для обеспечения заданной точности при вычислении значения определенного интеграла обычно прибегают к методу двойного пересчета, который состоит в следующем:

а) определяют начальный шаг интегрирования по формуле  $\Delta x = \sqrt[m]{\varepsilon}$ , где  $\varepsilon$  – заданная точность (допустимая погрешность),  $m = 2$  для метода трапеций и  $m = 4$  для метода Симпсона;

б) по выбранному шагу интегрирования определяют число разбиений отрезка интегрирования  $n = (b - a)/\Delta x$  для метода трапеций и  $n = \frac{b - a}{2 \cdot \Delta x}$  для метода Симпсона;

в) вычисляют значение интеграла по выбранной формуле (методу) дважды: сначала с  $n$  разбиениями –  $I_n$ , затем с  $2n$  разбиениями –  $I_{2n}$ ;

г) вычисляют оценку погрешности  $\delta = |I_n - I_{2n}|$ ;

д) если  $\delta < \varepsilon$ , то полагают  $I = I_{2n}$ , в противном случае вычисление интеграла повторяют при  $4n$  разбиениях и т.д. В некоторых случаях вычисляют относительную оценку погрешности

$$\delta_{\text{отн}} = \frac{|I_n - I_{2n}|}{I_{2n}}.$$

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как вычисляется значение определенного интеграла с помощью первообразной функции?
2. Почему не всегда возможно вычисление значения определенного интеграла с помощью первообразной функции?
3. Какие численные методы вычисления значения определенного интеграла Вы знаете?

- #### 4. РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

где  $A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix}$  – матрица коэффициентов,

$\bar{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_{n-1} \\ b_n \end{pmatrix}$  – вектор-столбец свободных членов,

$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{n-1} \\ x_n \end{pmatrix}$  – вектор-столбец неизвестных.

Количество уравнений (неизвестных) называют *порядком* системы.

Если матрица  $A$  неособенная, т. е.

$$D = \det A = |a_{ij}| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \bullet & \bullet & \bullet & \bullet \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \neq 0 \quad (4.3)$$

– главный определитель системы не равен нулю, то система (4.1) имеет единственное решение, т. е. существует единственная комбинация значений неизвестных  $x_1, x_2, \dots, x_{n-1}, x_n$  (корней системы), обращающих систему уравнений в тождество (систему тождеств). В этом случае с теоретической точки зрения решение системы (нахождение корней) не представляет труда. Корни системы

могут быть найдены по формулам *Крамера*:  $x_j = \frac{D_j}{D}$ , где  $D_j$  – определитель, получающийся из главного определителя системы  $D$  заменой  $j$ -го столбца столбцом свободных членов  $\bar{b}$ . Такой способ решения системы линейных алгебраических уравнений

требует вычисления  $n + 1$  определителей порядка  $n$ , что весьма трудоемко при сколько-нибудь большом  $n$  (количество вычислительных операций пропорционально  $n^3$ ).

Применяемые в настоящее время методы решения систем линейных алгебраических уравнений можно разделить на две группы: *прямые* и *косвенные*.

*Прямыми* называются такие методы, которые в предположении, что вычисления ведутся *точно* (без округлений), приводят к точным значениям корней  $x_j$  после выполнения *конечного, заранее известного*, числа вычислительных операций.

Фактически, при использовании вычислительной техники *все вычисления ведутся с округлением* и найденные значения корней, как правило, отличаются от точных. Поэтому главным признаком, характеризующим прямые методы, является именно *число вычислительных операций*. К прямым методам решения систем линейных алгебраических уравнений относятся метод *Крамера*, метод *Гаусса*, метод *квадратичных корней* и др.

*Косвенными* называют такие методы, при использовании которых решение системы находится путем последовательных приближений (итераций), *количество которых заранее предсказать невозможно*. Это количество зависит от заданной точности (допустимой погрешности) и от близости начальных приближений корней, которые необходимо задавать в начале процесса итераций, к их точным значениям. Теоретически точные значения корней могут быть получены в результате *бесконечного вычислительного процесса*. К косвенным методам относятся метод *простой итерации* и метод *Зейделя*.

#### 4.2. МЕТОД ГАУССА (ПОСЛЕДОВАТЕЛЬНОГО ИСКЛЮЧЕНИЯ НЕИЗВЕСТНЫХ)

Метод Гаусса наиболее распространенный метод решения системы линейных алгебраических уравнений. В его основе лежит идея *последовательного исключения неизвестных* из уравнений системы. Рассмотрим этот метод на примере системы трех уравнений с тремя неизвестными:

$$\left. \begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3. \end{aligned} \right\} \quad (4.4)$$



Исключим из второго и третьего уравнений системы (4.4) неизвестное  $x_1$ . Для этого необходимо уравнивать коэффициенты при  $x_1$  во всех трех уравнениях и почленно вычесть из второго и третьего уравнений первое. Разделим первое уравнение на  $a_{11}$  (при условии, что  $a_{11} \neq 0$ ) и получим

$$x_1 + b_{12} x_2 + b_{13} x_3 = b_{14}, \quad (4.5)$$

где  $b_{1j} = a_{1j}/a_{11}$  для  $j = 2, 3$  и  $b_{14} = b_{14}/a_{11}$ .

Умножив теперь уравнение (4.5) на  $a_{21}$  и  $a_{31}$  и вычтя почленно результаты соответственно из второго и третьего уравнений системы (4.4), получим:

$$\left. \begin{aligned} x_1 + b_{12} x_2 + b_{13} x_3 &= b_{14}, \\ a_{22}^{(1)} x_2 + a_{23}^{(1)} x_3 &= a_{24}^{(1)}, \\ a_{32}^{(1)} x_2 + a_{33}^{(1)} x_3 &= a_{34}^{(1)}, \end{aligned} \right\}, \quad (4.6)$$

где коэффициенты  $a_{ij}^{(1)}$  вычисляются по формуле

$$a_{ij}^{(1)} = a_{ij} - a_{i1} \cdot b_{1j}, \quad i = 2, 3; \quad j = 2, 3, 4. \quad (4.7)$$

Разделив второе уравнение системы (4.6) на  $a_{22}^{(1)}$ , получим

$$x_2 + b_{23}^{(1)} x_3 = b_{24}^{(1)}. \quad (4.8)$$

Умножив теперь уравнение (4.8) на  $a_{32}^{(1)}$  и вычтя результат из третьего уравнения системы (4.6), получим

$$\left. \begin{aligned} x_1 + b_{12} x_2 + b_{13} x_3 &= b_{14}, \\ x_2 + b_{23}^{(1)} x_3 &= b_{24}^{(1)}, \\ a_{33}^{(2)} x_3 &= a_{34}^{(2)}, \end{aligned} \right\} \quad (4.9)$$

где  $a_{ij}^{(2)} = a_{ij}^{(1)} - a_{i2}^{(1)} \cdot b_{2j}^{(1)}$  для  $i = 3, j = 3, 4$ .

Таким образом, система (4.4) приведена к эквивалентной системе (4.9) с *треугольной* матрицей коэффициентов. Из послед-

него уравнения этой системы находится  $x_3 = a_{34}^{(2)} / a_{33}^{(2)}$ , затем из второго –  $x_2 = b_{24}^{(1)} - b_{23}^{(1)} \cdot x_3$  и, наконец, из первого уравнения системы (4.9) определяется  $x_1 = b_{14} - b_{13} \cdot x_3 - b_{12} \cdot x_2$ . Приведение системы (4.4) к виду (4.9) называют *прямым ходом*, а вычисление значений неизвестных, начиная с  $x_n$  и заканчивая  $x_1$ , – *обратным ходом*.

При прямом ходе необходимо, чтобы элементы  $a_{11}, a_{22}^{(1)}, \dots, a_{nn}^{(n-1)}$ , которые называют *ведущими*, были *отличны от нуля*. Это необходимо учитывать при разработке алгоритма и программы.

На рис. 4.1 изображена блок-схема алгоритма решения системы линейных алгебраических уравнений методом Гаусса. Под расширенной матрицей коэффициентов и свободных членов системы понимают матрицу коэффициентов с присоединенным в качестве  $n + 1$ -го столбцом свободных членов. При этом приняты обозначения  $a_{i, n+1} = b_i$ .

Особенностью данного алгоритма является то, что при исключении очередного  $k$ -го неизвестного из всех уравнений с  $k + 1$ -го до  $n$ -го производится поиск *максимального по абсолютной величине* элемента (коэффициента при  $x_k$ ) в  $k$ -м столбце. Это делается для уменьшения ошибки при делении на этот элемент. Кроме того, после нахождения такого элемента и замены (перемены местами) строк производится проверка *особенности* (равенства нулю или малости главного определителя) матрицы.

В качестве контрольного примера рекомендуется следующая система уравнений:

$$\left. \begin{aligned} 6x_1 - 3x_2 + 2x_3 + x_4 - x_5 + x_6 &= 11, \\ -3x_1 - 7x_2 + 4x_4 - 2x_5 + x_6 &= -5, \\ 4x_1 - 3x_2 + 6x_3 + x_4 + 2x_5 + x_6 &= 28, \\ 2x_1 + 4x_2 + 5x_3 - 7x_4 - 3x_5 + 2x_6 &= -6, \\ -x_1 + 5x_5 - 4x_3 + 8x_5 - 2x_6 &= 25, \\ 3x_1 + 4x_3 - 2x_4 + 5x_5 - 6x_6 &= -4. \end{aligned} \right\} \quad (4.10)$$

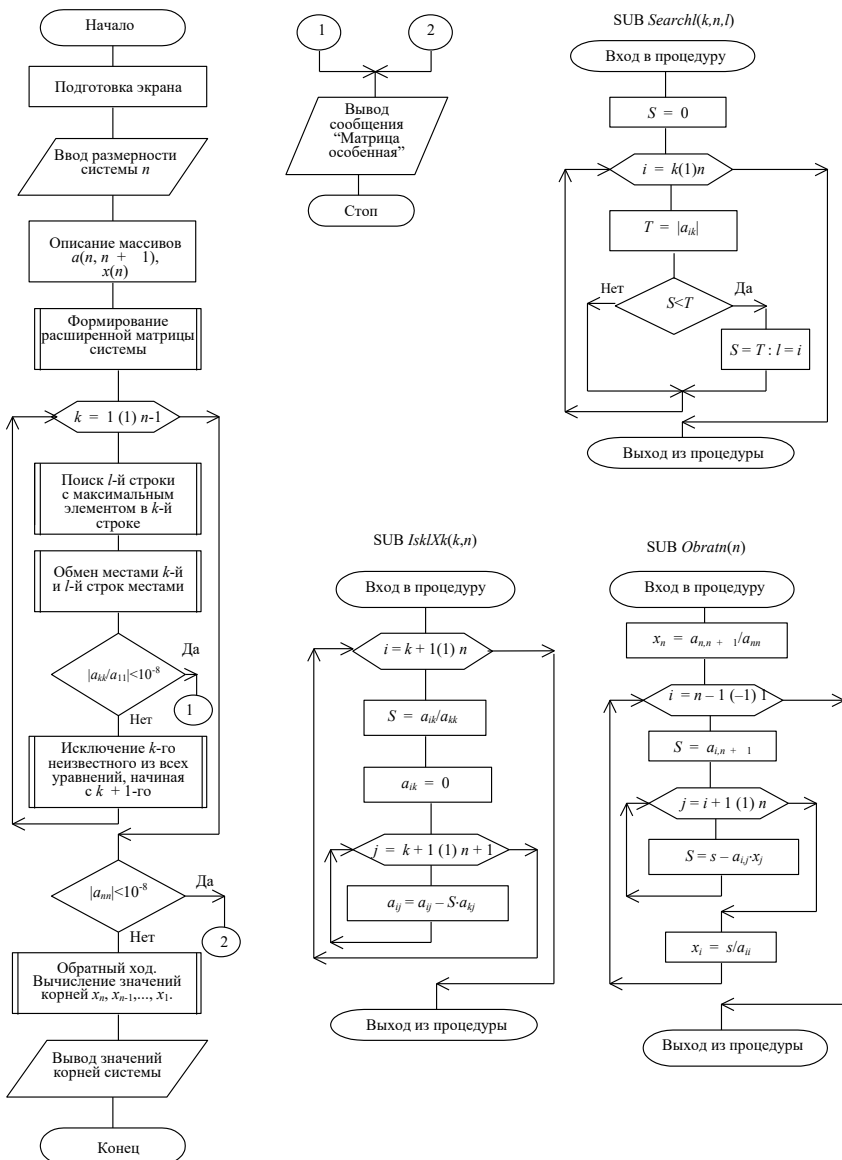


Рис.4.1. Блок-схемы алгоритмов головной программы и основных процедур решения системы линейных алгебраических уравнений методом Гаусса

Матрица коэффициентов и свободных членов этой системы (расширенная матрица системы) выглядит следующим образом:

$$A = \begin{pmatrix} 6 & -3 & 2 & 1 & -1 & 1 & 11 \\ -3 & -7 & 0 & 4 & -2 & 1 & -5 \\ 4 & -3 & 6 & 1 & 2 & 1 & 28 \\ 2 & 4 & 5 & -7 & -3 & 2 & -6 \\ -1 & 5 & -4 & 0 & 8 & -2 & 25 \\ 3 & 0 & 4 & -2 & 5 & -6 & -4 \end{pmatrix}.$$

Решение системы (4.10) дает результаты:  $x_1 = 1$ ,  $x_2 = 2$ ,  $x_3 = 3$ ,  $x_4 = 4$ ,  $x_5 = 5$ ,  $x_6 = 6$ .

Для контроля правильности решения системы уравнений рекомендуется вычислять невязки каждого уравнения системы. Обычно переносят свободные члены в левые части уравнений и вычисляют значения функций  $f_1(x_1, x_2, \dots, x_n)$ ,  $f_2(x_1, x_2, \dots, x_n)$ , ...,  $f_n(x_1, x_2, \dots, x_n)$  при найденных значениях неизвестных (корней системы).

При этом, если невязки вычисляются непосредственно после решения системы с помощью специальной процедуры и с использованием значений корней, находящихся в памяти компьютера (не округленных), то значения невязок могут отличаться от тех, которые получены при округленных (выведенных на экран или принтер) значениях корней.

#### 4.3. КОСВЕННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Метод простой итерации решения системы линейных алгебраических уравнений  $A \cdot \bar{x} = \bar{b}$  заключается в следующем:

а) систему  $A \cdot \bar{x} = \bar{b}$  приводят (преобразуют) к виду

$$\bar{x} = C \cdot \bar{x} + \bar{f}, \quad (4.11)$$

где  $C$  – некоторая матрица коэффициентов при  $\bar{x}$ ;  $\bar{f}$  – вектор-столбец свободных членов;

б) исходя из некоторого вектора *начальных* (нулевых) приближений корней  $\bar{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$ , строится *итерационный* процесс (процесс последовательных приближений)

или в развернутой форме

в) производя итерации, получим последовательность векторов  $\bar{x}^{(1)}, \bar{x}^{(2)}, \dots;$

$$\delta = \max_{i=1, \dots, n} \left\{ \frac{2 \cdot |x_i^{(k)} - x_i^{(k+1)}|}{|x_i^{(k)}| + |x_i^{(k+1)}|} \right\}. \quad (4.14)$$
$$\sum_{j=1}^n |C_{i,j}| < 1 \text{ для } i = 1, 2, \dots, n \quad (4.15)$$

Если для исходной системы уравнений  $A \cdot \bar{x} = \bar{b}$  выполняется условие

– диагональные элементы матрицы коэффициентов по абсолютной величине больше, чем сумма абсолютных величин остальных элементов данной строки, то эта система легко преобразуется к виду  $\bar{x} = C \cdot \bar{x} + \bar{f}$  для решения итерационным методом.

При разработке алгоритма для составления программы необходимо учитывать следующее:

- во избежание *зацикливания* программы надо отслеживать количество итераций  $k$  и прекращать вычисления при превышении некоторого максимального допустимого числа  $k_{\max}$ ;
- для вычислений необходимо хранить (иметь возможность хранить) значения неизвестных на двух итерациях –  $k$ -й и  $k + 1$ -й;
- определение  $\delta$  – максимальной относительной разности между значениями неизвестных (корней) на двух последовательных итерациях – производится попутно, при вычислении  $k + 1$ -го приближения очередного неизвестного  $x_i$ ;
- перед очередной итерацией производится переприсваивание –  $\bar{x}^{(k)} = \bar{x}^{(k+1)}$ ;
- в качестве начальных приближений корней системы можно рекомендовать принимать (после преобразования исходной системы уравнений) значения  $f_i/C_{ii}$ .

В качестве примера рассмотрим следующую систему:

$$\left. \begin{aligned} 20.9x_1 + 1.2x_2 + 2.1x_3 + 0.9x_4 &= 21.70, \\ 1.2x_1 + 21.2x_2 + 1.5x_3 + 2.5x_4 &= 27.46, \\ 2.1x_1 + 1.5x_2 + 19.8x_3 + 1.3x_4 &= 28.76, \\ 0.9x_1 + 2.5x_2 + 1.3x_3 + 32.1x_4 &= 49.72. \end{aligned} \right\} \quad (4.17)$$

Нетрудно убедиться, что для системы (4.17) выполняется условие (4.16). Для преобразования этой системы к виду  $\bar{x} = C \cdot \bar{x} + \bar{f}$  представим элементы главной диагонали матрицы коэффициентов  $A$  в виде двух слагаемых, например  $20.9 = 20.0 + 0.9$ , и в левой части каждого уравнения оставим большее слагаемое, а меньшее и все остальные члены уравнения перенесем в правую часть. В результате получим:

$$\left. \begin{aligned} 20x_1 &= -0.9x_1 - 1.2x_2 - 2.1x_3 - 0.9x_4 + 21.70, \\ 21x_2 &= -1.2x_1 - 0.2x_2 - 1.5x_3 - 2.5x_4 + 27.46, \\ 19x_3 &= -2.1x_1 - 1.5x_2 - 0.8x_3 - 1.3x_4 + 28.76, \\ 32x_4 &= -0.9x_1 - 2.5x_2 - 1.3x_3 - 0.1x_4 + 49.72. \end{aligned} \right\} \quad (4.18)$$

Теперь необходимо поделить все элементы каждого уравнения системы (4.18) на коэффициент при неизвестном в левой части. Тогда матрица  $C$  будет выглядеть следующим образом:

$$C = \begin{pmatrix} -\frac{0.9}{20} & -\frac{1.2}{20} & -\frac{2.1}{20} & -\frac{0.9}{20} \\ -\frac{1.2}{21} & -\frac{0.2}{21} & -\frac{1.5}{21} & -\frac{2.5}{21} \\ -\frac{2.1}{19} & -\frac{1.5}{19} & -\frac{0.8}{19} & -\frac{1.3}{19} \\ -\frac{0.9}{32} & -\frac{2.5}{32} & -\frac{1.3}{32} & -\frac{0.1}{32} \end{pmatrix} = \begin{pmatrix} -0.045 & -0.060 & -0.105 & -0.45 \\ -0.057 & -0.0095 & -0.071 & -0.119 \\ -0.111 & -0.071 & -0.42 & -0.068 \\ -0.028 & -0.078 & -0.406 & -0.0031 \end{pmatrix}, (4.19)$$

а столбец свободных членов

$$\bar{f} = \begin{pmatrix} 21.70/20 \\ 27.46/20 \\ 28.76/20 \\ 49.72/20 \end{pmatrix} = \begin{pmatrix} 1.085 \\ 1.308 \\ 1.514 \\ 1.554 \end{pmatrix}. \quad (4.20)$$

Решение системы (4.16) с помощью программы, составленной в соответствии с алгоритмом, изображенным на рис. 4.2, при  $\varepsilon = 0.001$  дает следующие результаты:  $k = 6$ ,  $\delta = 3.79 \cdot 10^{-4}$ ,  $x_1 = 0.80$ ,  $x_2 = 1.0$ ,  $x_3 = 1.2$ ,  $x_4 = 1.4$ .

Метод Зейделя [3] является модификацией метода простой итерации. Состоит он в том, что при вычислении  $k + 1$ -го приближения неизвестного  $x_i$  для всех  $i > 1$  используют уже вычисленные ранее  $k + 1$ -е приближения неизвестных  $x_1, x_2, \dots, x_{i-1}$ . Таким образом, для системы (4.13) вычисления ведутся по формулам

$$\left. \begin{aligned} x_1^{(k+1)} &= C_{11}x_1^{(k)} + C_{12}x_2^{(k)} + \dots + C_{1,n-1}x_{n-1}^{(k)} + C_{1n}x_n^{(k)} + f_1, \\ x_2^{(k+1)} &= C_{21}x_1^{(k+1)} + C_{22}x_2^{(k)} + \dots + C_{2,n-1}x_{n-1}^{(k)} + C_{2n}x_n^{(k)} + f_2, \\ &\dots\dots\dots, \\ x_n^{(k+1)} &= C_{n1}x_1^{(k+1)} + C_{n2}x_2^{(k+1)} + \dots + C_{n,n-1}x_{n-1}^{(k+1)} + C_{nn}x_n^{(k)} + f_n. \end{aligned} \right\} \quad (4.21)$$

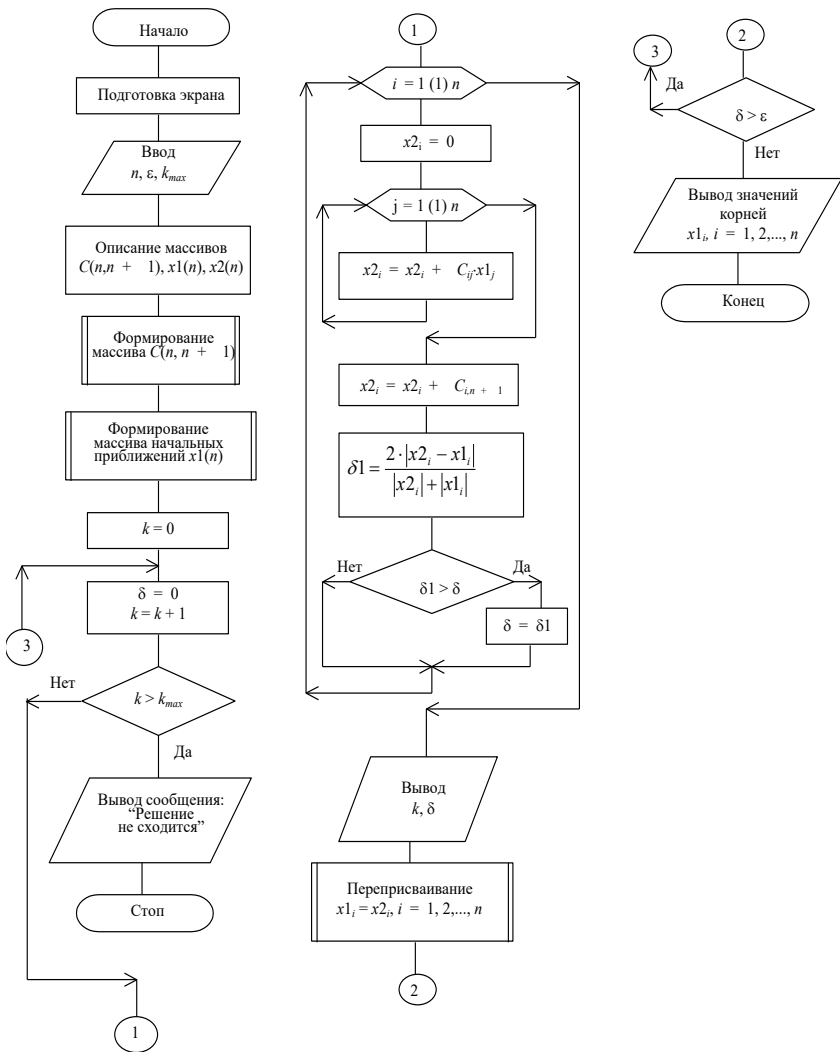


Рис. 4.2. Блок-схема алгоритма решения системы линейных алгебраических уравнений методом простой итерации



Метод Зейделя, как правило, *быстрее* приводит к желаемому результату. Кроме этого, нет необходимости хранить значения всех неизвестных для двух последовательных итераций, достаточно обозначить  $x_i^{(k+1)}$  как  $x_i$  и переприсваивание  $x_i^{(k)} = x_i$  производить сразу после проверки условия  $\delta_1 < \delta$ . Решение системы (4.16) методом Зейделя дает следующие результаты:  $k = 4$ ,  $\delta = 3.114 \cdot 10^{-4}$ ,  $x_1 = 0.8$ ,  $x_2 = 1.0$ ,  $x_3 = 1.2$ ,  $x_4 = 1.4$ .

Итерационные методы (если они сходятся) имеют следующие преимущества по сравнению с прямыми методами:

- если для решения системы порядка  $n$  требуется менее  $n$  итераций, то получается выигрыш во времени (это происходит обычно при  $n > 10$ );
- погрешность округления в итерационных методах сказывается меньше, чем в прямых;
- итерационные методы особенно выгодны при решении систем уравнений, у которых значительное число коэффициентов равно нулю (матрица разрежена);
- процесс итерации состоит из однообразных операций и сравнительно легко программируется для компьютера.

### КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие уравнения называются алгебраическими?
2. Какие алгебраические уравнения являются линейными?
3. Какие уравнения, кроме алгебраических, Вы знаете?
4. Как в матричной форме записывается система линейных алгебраических уравнений?
5. Что понимается под матрицей коэффициентов системы линейных алгебраических уравнений?
6. Что понимается под вектором-столбцом свободных членов системы линейных алгебраических уравнений?
7. Что понимается под вектором-столбцом неизвестных?
8. Какому условию должна удовлетворять система линейных алгебраических уравнений, чтобы иметь единственное решение?
9. Что понимается под главным определителем системы линейных алгебраических уравнений?
10. В чем состоит метод Крамера решения системы линейных алгебраических уравнений?
11. Какие методы решения систем линейных алгебраических уравнений называются *прямыми* и почему?

12. Какие методы решения системы линейных алгебраических уравнений называются *косвенными* и почему?

13. В чем заключается идея метода Гаусса для решения системы линейных алгебраических уравнений?

14. Что происходит при прямом ходе в методе Гаусса?

15. Что происходит при обратном ходе в методе Гаусса?

16. Почему при исключении очередного неизвестного производится поиск наибольшего по абсолютной величине элемента в первом (левом) столбце матрицы коэффициентов?

17. Что понимается под расширенной матрицей системы?

18. Объясните алгоритм, изображенный на рис. 4.1.

19. Как производится проверка матрицы на особенность (вырожденность) при решении системы линейных алгебраических уравнений методом Гаусса?

20. Объясните алгоритм процедуры поиска строки с максимальным элементом в левом столбце.

21. Напишите (составьте) процедуру перестановки (замены местами)  $k$ -й и  $l$ -й строк матрицы.

22. Объясните работу процедуры исключения  $k$ -го неизвестного из всех уравнений, начиная с  $k + 1$ -го.

23. Напишите (составьте) процедуру, реализующую обратный ход.

24. В чем состоит идея метода простой итерации для решения системы линейных алгебраических уравнений?

25. К какому виду необходимо преобразовать исходную систему уравнений для решения ее итерационными методами?

26. Какому условию должна удовлетворять матрица коэффициентов преобразованной системы (матрица  $C$ ), чтобы обеспечивалась сходимость итерационного метода?

27. Какому условию должна удовлетворять матрица коэффициентов исходной системы уравнений (матрица  $A$ ), чтобы обеспечивалась сходимость итерационного метода?

28. Как преобразовать исходную систему  $A \cdot \bar{x} = \bar{b}$  к виду  $\bar{x} = C \cdot \bar{x} + \bar{f}$ ?

29. Как принимаются начальные приближения неизвестных при решении системы линейных алгебраических уравнений итерационными методами?

30. Чем отличается метод Зейделя от метода простой итерации?

31. Как убедиться в правильности решения системы линейных алгебраических уравнений?  
32. Что такое невязка уравнения и как она вычисляется?

## 5. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ЧИСЛЕННЫМИ МЕТОДАМИ

### 5.1. ПОСТАНОВКА ЗАДАЧИ И ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

*Обыкновенным* дифференциальным уравнением (в отличие от дифференциальных уравнений с *частными* производными) называется уравнение вида

$$F(x, y, y', y'', \dots, y^{(n-1)}, y^{(n)}) = 0, \quad (5.1)$$

где  $x$  – независимая переменная (аргумент);  $y = f(x)$  – искомая функция;  $y', y'', \dots, y^{(n)}$  – производные первого, второго,  $n$ -го порядков.

Порядок дифференциального уравнения определяется порядком старшей производной, входящей в уравнение. В инженерной практике чаще всего приходится решать уравнения первого и второго порядков. Функция  $y = f(x)$  называется решением уравнения (5.1), если при подстановке ее в уравнение последнее обращается в тождество.

Если удастся найти математическое выражение для  $y = f(x)$ , то говорят об *аналитическом* решении уравнения (5.1). Например, для уравнения  $y' - \cos x = 0$  аналитическое решение может быть найдено следующим образом:  $dy = \cos x \cdot dx$ ,  $\int dy = \int \cos x \cdot dx$ ,  $y = \sin x + C$ . Постоянная интегрирования  $C$  может быть определена при наличии некоторого дополнительного условия типа  $y_0 = f(x_0) = a$ . Методы аналитического решения различных типов обыкновенных дифференциальных уравнений изучаются в математическом анализе.

Однако во многих случаях найти аналитическое решение дифференциального уравнения не удастся или оно весьма затруднительно. В то же время существуют *численные* методы решения дифференциальных уравнений, применение которых при наличии средств вычислительной техники не составляет большо-

го труда. Ниже рассматриваются два из них: метод *Эйлера* и метод *Рунге-Кутты*.

## 5.2. МЕТОД ЭЙЛЕРА

Для численного решения обыкновенного дифференциального уравнения первого порядка оно должно быть приведено к *нормальному* виду (нормализовано). Это означает, что уравнение должно быть преобразовано таким образом, чтобы в левой части находилась *только производная* искомой функции (с коэффициентом, равным единице), а в правой – все остальное. Итак, имеем дифференциальное уравнение

$$y' = f(x, y) \quad (5.2)$$

и начальное условие

$$y(x_0) = y_0. \quad (5.3)$$

Функцию  $f(x, y)$  называют *правой частью* дифференциального уравнения.

Задача численного решения уравнения (5.2) при начальном условии (5.3) состоит в том, чтобы на некотором отрезке  $[x_0, x_k]$  определить (найти) значения искомой функции  $y = f(x)$ , иначе говоря, составить *таблицу значений функции с некоторым шагом  $h$*  (будем в дальнейшем называть его шагом *табулирования*). Процедуру численного решения дифференциального уравнения часто называют *интегрированием*.

Для решения задачи выбирается малый шаг  $\Delta x \leq h$  (*шаг интегрирования*) и строится система равноотстоящих точек  $x_i = x_0 + i \cdot \Delta x$ ,  $i = 0, 1, 2, \dots, n$ , где  $n = (x_k - x_0)/\Delta x$ . Приближенные значения искомой функции  $y_i \approx f(x_i)$  в этих точках вычисляются по формуле Эйлера

$$y_{i+1} \approx y_i + \Delta x f(x_i, y_i), \quad i = 0, 1, \dots, n-1. \quad (5.4)$$

Искомая интегральная кривая – график функции  $y = f(x)$  – заменяется *ломаной* (ломаной Эйлера). Геометрическая иллюстрация метода Эйлера приведена на рис. 5.1.

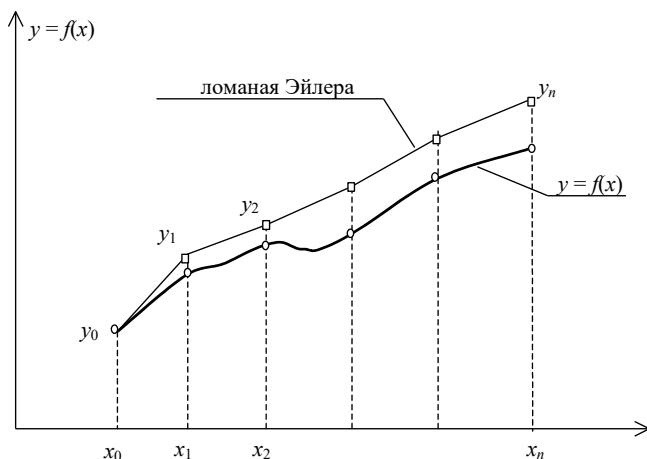


Рис. 5.1. Геометрическая иллюстрация метода Эйлера

Погрешность решения дифференциального уравнения методом Эйлера зависит от шага интегрирования  $\Delta x$ . Ниже будет показано, что она (погрешность) достаточно высока, поэтому этот метод следует применять только для грубой, чисто качественной, оценки характера поведения искомой функции.

Алгоритм решения обыкновенного дифференциального уравнения первого порядка методом Эйлера может быть представлен следующими основными операциями:

- 1) ввод  $x_0, x_k, y_0, h$ ;
- 2) ввод коэффициента кратности  $k$ ;
- 3) вычисление шага интегрирования  $\Delta x = h/k$ ;
- 4) определение количества строк таблицы значений функции  $n = (x_k - x_0)/h$ ;
- 5) организация цикла для  $n$  повторений;
- 6) организация цикла для  $k$  повторений;
- 7) вычисление нового значения искомой функции (повторение  $k$  раз);
- 8) вывод вычисленных значений  $x$  и  $y$  (повторение  $n$  раз);
- 9) прекращение вычислений.

Блок-схема алгоритма численного решения обыкновенного дифференциального уравнения первого порядка методом Эйлера приведена на рис. 5.2.

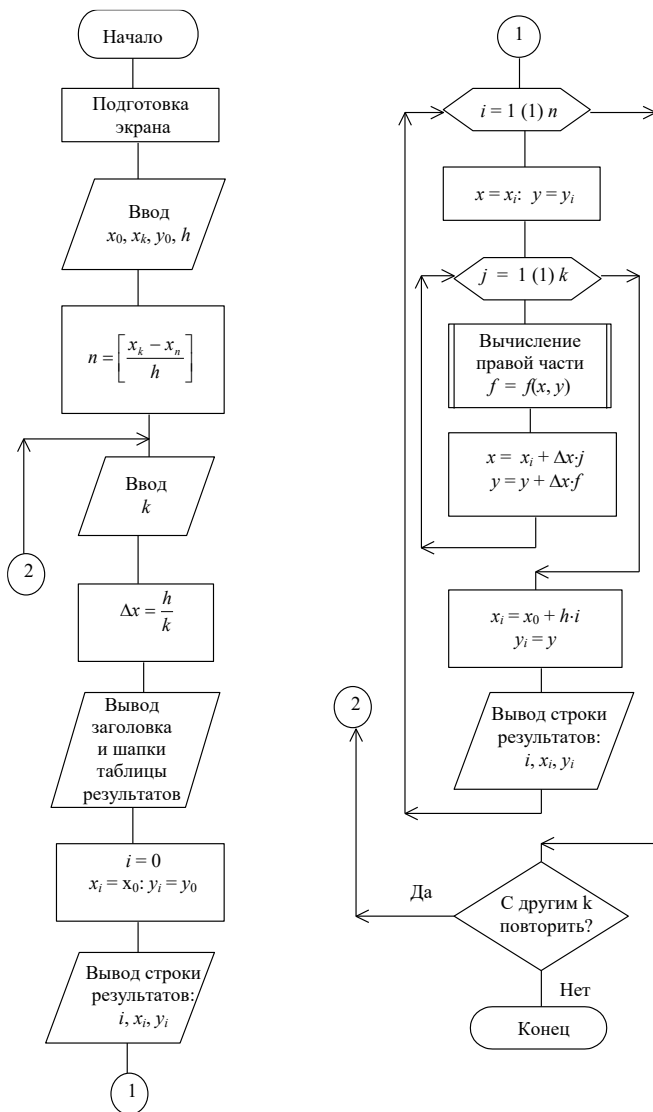


Рис. 5.2. Блок-схема алгоритма решения обыкновенного дифференциального уравнения первого порядка методом Эйлера

### 5.3. МЕТОД РУНГЕ-КУТТА

По методу Рунге-Кутта для уравнения (5.2) с начальными условиями (5.3) вычисление приближенного значения искомой функции  $y_{i+1}$  в точке  $x_{i+1} = x_i + \Delta x$  производится по формулам

$$\left. \begin{aligned} y_{i+1} &= y_i + \Delta y_i, \\ \Delta y_i &= \frac{1}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \right\}, \quad (5.5)$$

где

$$\left. \begin{aligned} k_1 &= \Delta x \cdot f(x_i, y_i), \\ k_2 &= \Delta x \cdot f(x_i + \Delta x/2, y_i + k_1/2), \\ k_3 &= \Delta x \cdot f(x_i + \Delta x/2, y_i + k_2/2), \\ k_4 &= \Delta x \cdot f(x_i + \Delta x, y_i + k_3). \end{aligned} \right\} \quad (5.6)$$

Погрешность решения дифференциального уравнения методом Рунге-Кутта имеет порядок  $(\Delta x)^4$  на всем отрезке  $[x_0, x_k]$ . Грубую оценку погрешности можно получить с помощью двойного пересчета по формуле [3]

$$\varepsilon = \left| y_i^* - y(x_i) \right| \approx \frac{|y_i^* - y_i|}{15},$$

где  $y(x_i)$  – значение точного решения уравнения в точке  $x_i$ ;  $y_i$  – приближенное значение, полученное при численном решении методом Рунге-Кутта с шагом интегрирования  $\Delta x$ ;  $y_i^*$  – приближенное значение, полученное при численном решении методом Рунге-Кутта с *половинным* шагом интегрирования  $\Delta x/2$ .

Алгоритм численного решения обыкновенного дифференциального уравнения первого порядка методом Рунге-Кутта изображен на рис. 5.3. Отличается он от алгоритма решения методом Эйлера процедурой интегрирования. Блок-схема алгоритма этой процедуры изображена на рис. 5.4. Из блок-схемы видно, что для вычисления одного значения искомой функции (одного интегрирования) необходимо *четыре раза* вычислять значение правой части нормированного дифференциального уравнения. Однако это окупается *высокой точностью* (низкой погрешностью) метода Рунге-Кутта.

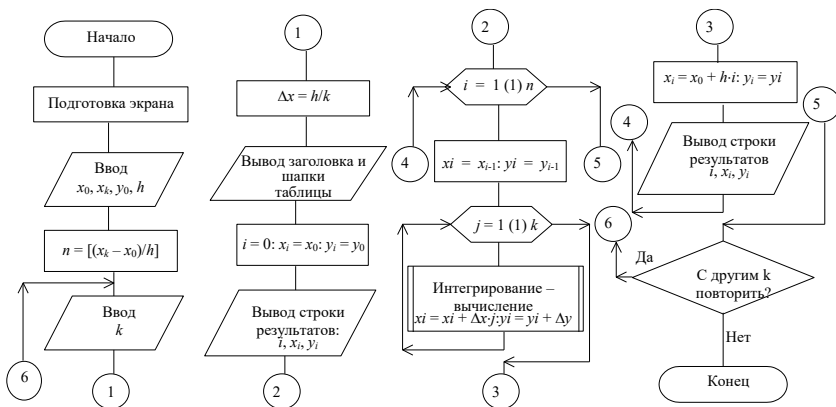


Рис. 5.3. Блок-схема алгоритма решения обыкновенного дифференциального уравнения первого порядка методом Рунге-Кутты

SUB IntgRK(xi, yi, Δx, k)

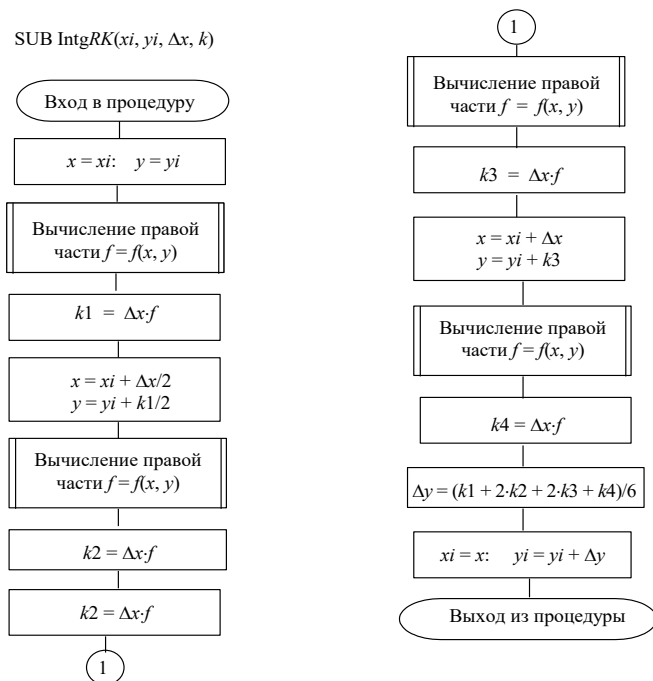


Рис. 5.4. Блок-схема алгоритма процедуры интегрирования методом Рунге-Кутты



В табл. 5.1 приведены результаты численного решения дифференциального уравнения  $y' = 1 + 0.4 \cdot y \cdot \sin x - 2 \cdot y^2$  на отрезке  $[0, 1]$  при начальных условиях  $y(0) = 0$  для  $\Delta x = h = 0.1$  (шаг интегрирования принят равным шагу табулирования) методами Эйлера и Рунге-Кутты. В графе  $\varepsilon_i$  приведены значения  $|y_i - y_i^*|$

Т а б л и ц а 5.1

Результаты решения дифференциального уравнения методами Эйлера и Рунге-Кутты

$i$	$x_i$	Методом Эйлера			Методом Рунге-Кутты		
		$y_i$	$y_i^*$	$\varepsilon_i$	$y_i$	$y_i^*$	$\varepsilon_i$
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.1	0.1000	0.0998	$2.0 \cdot 10^{-4}$	0.0995	0.0995	$2.31 \cdot 10^{-7}$
2	0.2	0.1984	0.1972	$1.2 \cdot 10^{-3}$	0.1958	0.1958	$5.07 \cdot 10^{-7}$
3	0.3	0.2921	$0.2894^4$	$2.7 \cdot 10^{-3}$	0.2864	0.2864	$8.64 \cdot 10^{-7}$
4	0.4	0.3785	0.3739	$4.6 \cdot 10^{-3}$	0.3693	0.3693	$1.34 \cdot 10^{-6}$
5	0.5	0.4557	0.4494	$6.3 \cdot 10^{-3}$	0.4432	0.4432	$1.91 \cdot 10^{-6}$
6	0.6	0.5229	0.5152	$7.7 \cdot 10^{-3}$	0.5077	0.5077	$2.50 \cdot 10^{-6}$
7	0.7	0.5801	0.5714	$8.7 \cdot 10^{-3}$	0.5631	0.5631	$2.98 \cdot 10^{-6}$
8	0.8	0.6277	0.6187	$9.0 \cdot 10^{-3}$	0.6101	0.6101	$3.40 \cdot 10^{-6}$
9	0.9	0.6669	0.6580	$8.9 \cdot 10^{-3}$	0.6496	0.6496	$3.70 \cdot 10^{-6}$
10	1.0	0.6989	0.6905	$8.4 \cdot 10^{-3}$	0.6824	0.6824	$3.81 \cdot 10^{-6}$

#### 5.4. ЧИСЛЕННОЕ РЕШЕНИЕ СИСТЕМЫ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА

Методы Эйлера и Рунге-Кутты легко распространяются на системы обыкновенных дифференциальных уравнений первого порядка. Для примера рассмотрим систему двух уравнений:

$$\left. \begin{aligned} y_1' &= f_1(x, y_1, y_2), \\ y_2' &= f_2(x, y_1, y_2), \end{aligned} \right\} \quad (5.7)$$

с начальными условиями

$$\left. \begin{aligned} y_1(x_0) &= y_1^{(0)}, \\ y_2(x_0) &= y_2^{(0)}. \end{aligned} \right\} \quad (5.8)$$

Для этой системы приближенные значения искомых функций на некотором  $j + 1$ -м шаге интегрирования при  $x^{(j+1)} = x^{(j)} + \Delta x$  вычисляются по формулам Эйлера

$$\left. \begin{aligned} y_1^{(j+1)} &= y_1^{(j)} + \Delta x \cdot f_1 \left( x^{(j)}, y_1^{(j)}, y_2^{(j)} \right), \\ y_2^{(j+1)} &= y_2^{(j)} + \Delta x \cdot f_2 \left( x^{(j)}, y_1^{(j)}, y_2^{(j)} \right), \end{aligned} \right\} \quad (5.9)$$

где  $j = 0, 1, 2, \dots$

Формулы Рунге-Кутта для системы уравнений (5.7) выглядят следующим образом:

$$\left. \begin{aligned} y_1^{(j+1)} &= y_1^{(j)} + \Delta y_1^{(j)}, \\ y_2^{(j+1)} &= y_2^{(j)} + \Delta y_2^{(j)}, \\ \Delta y_1^{(j)} &= \frac{1}{6} \cdot (k_{1y_1} + 2k_{2y_1} + 2k_{3y_1} + k_{4y_1}), \\ \Delta y_2^{(j)} &= \frac{1}{6} \cdot (k_{1y_2} + 2k_{2y_2} + 2k_{3y_2} + k_{4y_2}), \end{aligned} \right\} \quad (5.10)$$

$$\left. \begin{aligned} k_{1y_1} &= \Delta x \cdot f_1(x^{(j)}, y_1^{(j)}, y_2^{(j)}), \\ k_{1y_2} &= \Delta x \cdot f_2(x^{(j)}, y_1^{(j)}, y_2^{(j)}), \\ k_{2y_1} &= \Delta x \cdot f_1 \left( x^{(j)} + \Delta x/2, y_1^{(j)} + k_{1y_1}/2, y_2^{(j)} + k_{1y_2}/2 \right), \\ k_{2y_2} &= \Delta x \cdot f_2 \left( x^{(j)} + \Delta x/2, y_1^{(j)} + k_{1y_1}/2, y_2^{(j)} + k_{1y_2}/2 \right), \\ k_{3y_1} &= \Delta x \cdot f_1 \left( x^{(j)} + \Delta x/2, y_1^{(j)} + k_{2y_1}/2, y_2^{(j)} + k_{2y_2}/2 \right), \\ k_{3y_2} &= \Delta x \cdot f_2 \left( x^{(j)} + \Delta x/2, y_1^{(j)} + k_{2y_1}/2, y_2^{(j)} + k_{2y_2}/2 \right), \\ k_{4y_1} &= \Delta x \cdot f_1 \left( x^{(j)} + \Delta x, y_1^{(j)} + k_{3y_1}, y_2^{(j)} + k_{3y_2} \right), \\ k_{4y_2} &= \Delta x \cdot f_2 \left( x^{(j)} + \Delta x, y_1^{(j)} + k_{3y_1}, y_2^{(j)} + k_{3y_2} \right). \end{aligned} \right\} \quad (5.11)$$

При решении системы  $n$  уравнений используются аналогичные формулы. На рис. 5.5 приведена блок-схема алгоритма процедуры интегрирования системы двух дифференциальных уравнений методом Рунге-Кутта.

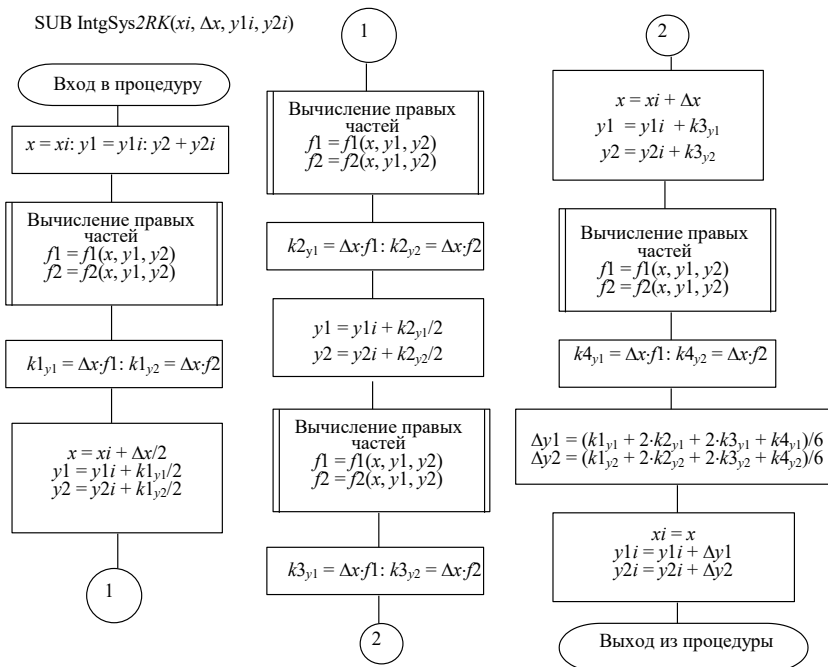


Рис. 5.5. Блок-схема алгоритма процедуры интегрирования системы двух дифференциальных уравнений первого порядка методом Рунге-Кутты

## 5.5. ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ВЫСШИХ ПОРЯДКОВ

Для численного решения обыкновенного дифференциального уравнения высшего порядка оно приводится к системе дифференциальных уравнений первого порядка путем введения вспомогательных переменных (функций). Например, уравнение второго порядка

$$y'' = f(x, y, y') \quad (5.12)$$

с начальными условиями (количество начальных условий определяется порядком дифференциального уравнения)

$$y(x_0) = y_0, \quad y'(x_0) = y'_0 \quad (5.13)$$

приводится к системе двух дифференциальных уравнений первого порядка с помощью двух вспомогательных переменных (функций)  $y_1(x)$  и  $y_2(x)$ :

$$\left. \begin{aligned} y'_1 &= f_1(x, y_1, y_2), \\ y'_2 &= f_2(x, y_1, y_2), \end{aligned} \right\} \quad (5.14)$$

с начальными условиями

$$y_1^{(0)} = y_0, \quad y_2^{(0)} = y'_0. \quad (5.15)$$

В системе уравнений (5.14) правая часть первого уравнения есть не что иное, как  $y_2$ , т. е.  $f_1(x, y_1, y_2) = y_2$ , а правая часть второго  $-f_2(x, y_1, y_2) = f(x, y, y')$  – есть правая часть исходного дифференциального уравнения второго порядка. В табл. 5.2 приведены результаты решения следующего дифференциального уравнения:  $y'' - 3y' + 2y = 2\sin x$ ,  $x \in [0, 1]$ ,  $y(0) = 2.6$ ,  $y'(0) = 3.2$ ,  $h = 0.1$ .

Т а б л и ц а 5.2

**Результаты численного решения обыкновенного  
дифференциального уравнения второго порядка**

$i$	$x_i$	Методом Эйлера; $\Delta x = h/8$		Методом Рунге-Кутты; $\Delta x = h$	
		$y_i$	$y_i$	$y_i$	$y'_i$
0	0.0	2.600	3.200	2.600	3.200
1	0.1	2.640	3.680	2.944	3.687
2	0.2	3.333	4.266	3.341	4.282
3	0.3	3.790	4.978	3.804	5.008
4	0.4	4.324	5.846	4.348	5.893
5	0.5	4.953	6.901	4.989	6.973
6	0.6	5.697	8.183	5.750	8.289
7	0.7	6.581	9.740	6.657	9.890
8	0.8	7.635	11.63	7.740	11.84
9	0.9	8.895	13.93	9.039	14.21
10	1.0	10.40	16.71	10.60	17.10

## 5.6. ПРИМЕР РЕШЕНИЯ ИНЖЕНЕРНОЙ ЗАДАЧИ

Рассмотрим следующую задачу. Необходимо смоделировать переходный процесс в электрической цепи постоянного тока при замыкании ключа (рис. 5.6).

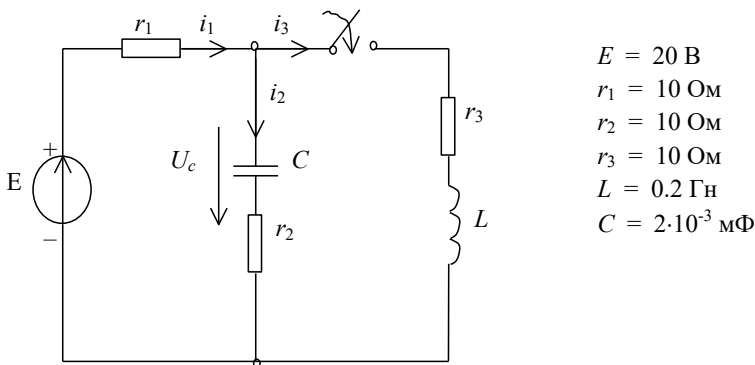


Рис. 5.6. Схема разветвленной электрической цепи постоянного тока с сосредоточенными параметрами

### 5.6.1. Постановка задачи

Решение задачи начинается с ее постановки. Смоделировать переходный процесс – это значит определить законы изменения токов и напряжений в ветвях цепи в течение определенного времени, пока их значения не установятся.

Исходными данными для моделирования являются известные параметры цепи: напряжение источника  $E$ , значения сопротивлений  $r_1$ ,  $r_2$ ,  $r_3$ , индуктивность катушки  $L$ , емкость конденсатора  $C$ . Исходными данными также можно считать начальные условия – значения токов и напряжений в начальный момент после коммутации (в начале моделирования).

Результатами моделирования должны быть значения токов и напряжений в ветвях цепи в функции времени:  $i_1(t)$ ,  $i_2(t)$ ,  $i_3(t)$ ,  $U_c(t)$ . Результаты желательно оформить в виде таблицы и соответствующих графиков.

### 5.6.2. Составление математической модели

Для составления математической модели используем законы Кирхгофа и запишем исходную систему уравнений

$$\left. \begin{aligned} E &= i_1 \cdot r_1 + i_2 \cdot r_2 + U_c, \\ E &= i_1 \cdot r_1 + i_3 \cdot r_3 + L \cdot \frac{di_3}{dt}, \\ i_1 &= i_2 + i_3. \end{aligned} \right\} \quad (5.16)$$

Ток  $i_2$  является током заряда-разряда конденсатора. Для него можно (и нужно) записать следующее выражение:

$$i_2 = i_c = C \cdot \frac{dU_c}{dt}. \quad (5.17)$$

Совершенно очевидно, что в конечном итоге в качестве математической модели мы получим систему двух обыкновенных дифференциальных уравнений первого порядка. Используя (5.17) и третье уравнение системы (5.16), запишем

$$i_1 = C \cdot \frac{dU_c}{dt} + i_3. \quad (5.18)$$

Подставив выражение (5.17) и (5.18) в первое уравнение системы (5.16), получим:  $E = (C \cdot \frac{dU_c}{dt} + i_3) \cdot r_1 + C \cdot \frac{dU_c}{dt} \cdot r_2 + U_c$ , или

$$C \cdot (r_1 + r_2) \cdot \frac{dU_c}{dt} = E - U_c - i_3 \cdot r_1.$$

Окончательно

$$\frac{dU_c}{dt} = \frac{E - U_c - i_3 \cdot r_1}{C \cdot (r_1 + r_2)}. \quad (5.19)$$

Из (5.19) можно выразить  $i_2 = i_c$ :

$$i_2 = \frac{E - U_c - i_3 \cdot r_1}{r_1 + r_2}. \quad (5.20)$$

Используя (5.20), выразим  $i_1$ :

$$i_1 = \frac{E - U_c + i_3 \cdot r_2}{r_1 + r_2}. \quad (5.21)$$

Подставив (5.21) во второе уравнение системы (5.16), разрешив его относительно производной, получим:

$$\frac{di_3}{dt} = \frac{E \cdot r_2 + U_c \cdot r_1 - i_3 \cdot (r_1 \cdot r_2 + r_1 \cdot r_3 + r_2 \cdot r_3)}{L \cdot (r_1 + r_2)}. \quad (5.22)$$

Обозначим  $r_1 + r_2 = r_4$  и  $r_1 \cdot r_2 + r_1 \cdot r_3 + r_2 \cdot r_3 = r_5$  и запишем систему двух дифференциальных уравнений

$$\left. \begin{aligned} \frac{dU_c}{dt} &= \frac{E - U_c - i_3 \cdot r_1}{C \cdot r_4}, \\ \frac{di_3}{dt} &= \frac{E \cdot r_2 + U_c \cdot r_1 - i_3 \cdot r_5}{L \cdot r_4}. \end{aligned} \right\} \quad (5.23)$$

Таким образом, математической моделью для решения задачи моделирования переходного процесса в цепи постоянного тока являются система дифференциальных уравнений (5.23) и выражения (формулы) (5.21) и (5.20) для определения  $i_1$  и  $i_2$ .

Для решения системы дифференциальных уравнений необходимо задать начальные условия. На основе законов коммутации можно записать:

$$U_c^{(0)} = E, i_3^{(0)} = 0. \quad (5.24)$$

### 5.6.3. Метод решения

Для решения системы двух обыкновенных дифференциальных уравнений первого порядка воспользуемся выше изложенным методом Рунге-Кутты. Независимой переменной (аргументом) в данном случае является время  $t$ , а искомыми функциями – напряжение  $U_c$  и ток  $i_3$ , которые определяются в результате решения системы (5.23). Значения токов  $i_1$  и  $i_2$  определяются по формулам (5.21) и (5.20) после нахождения  $U_c$  и  $i_3$ .

В табл. 5.3 приведены результаты моделирования, а на рис. 5.7 соответствующие графики.

Т а б л и ц а 5.3

Результаты моделирования переходного процесса в разветвленной электрической цепи постоянного тока с сосредоточенными параметрами

Исходные данные

$$E = 20 \text{ В} \quad r_1 = r_2 = r_3 = 10 \text{ Ом} \quad L = 0.2 \text{ Гн} \quad C = 2 \cdot 10^{-3} \text{ мФ}$$

$$t_{\text{кон}} = 0.2 \text{ с} \quad t_{pr} = h = 0.02 \text{ с} \quad \Delta t = 0.001 \text{ с}$$

N п/п	$t$	$U_c$	$i_1$	$i_2$	$i_3$
0	0.00	20.00	0.000	0.000	0.000
1	0.02	17.63	0.580	-0.343	0.924
2	0.04	14.41	0.835	-0.277	1.112
3	0.06	12.23	0.940	-0.163	1.104
4	0.08	11.03	0.981	-0.083	1.064
5	0.10	10.44	0.995	-0.039	1.034
6	0.12	10.18	0.999	-0.017	1.016
7	0.14	10.07	1.000	-0.007	1.007
8	0.16	10.02	1.000	-0.003	1.003
9	0.18	10.01	1.000	-0.001	1.001
10	0.20	10.00	1.000	-0.000	1.000

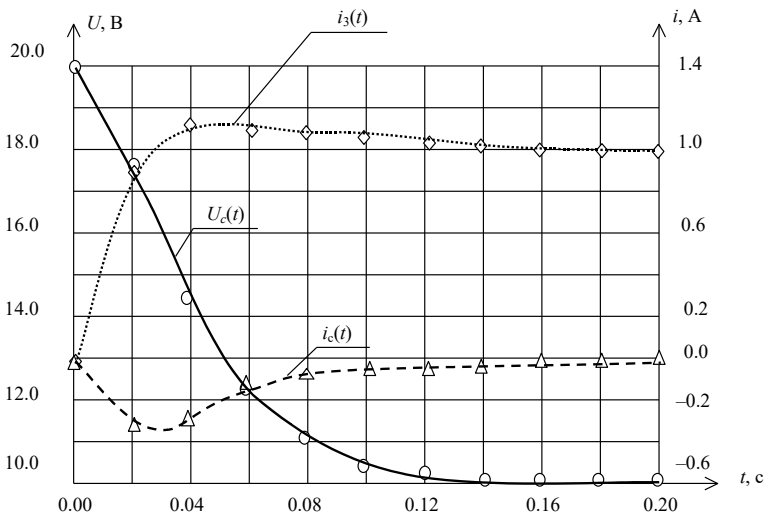


Рис. 5.7. Графики переходного процесса в цепи постоянного тока



## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каких случаях возникает необходимость численного решения дифференциального уравнения?
2. В чем принципиальная разница между аналитическим и численным решениями дифференциального уравнения?
3. К какому виду должно быть приведено обыкновенное дифференциальное уравнение первого порядка для численного решения?
4. В чем заключается метод Эйлера решения дифференциального уравнения первого порядка?
5. Объясните метод Рунге-Кутты решения обыкновенного дифференциального уравнения первого порядка.
6. Какова зависимость погрешности численного решения дифференциального уравнения от шага интегрирования?
7. Как практически оценивается погрешность численного решения дифференциального уравнения?
8. Какой из методов – Эйлера или Рунге-Кутта – является более точным?
9. Объясните блок-схему алгоритма процедуры интегрирования обыкновенного дифференциального уравнения первого порядка методом Рунге-Кутты.
10. Как распространяются численные методы решения одного дифференциального уравнения первого порядка на системы дифференциальных уравнений?
11. Как решаются дифференциальные уравнения высших порядков численными методами?
12. Объясните блок-схему алгоритма процедуры интегрирования системы двух дифференциальных уравнений первого порядка методом Рунге-Кутты.

## СПИСОК ЛИТЕРАТУРЫ

1. Демидович Б. П., Марон И. А. Основы вычислительной математики. – М.: Наука, 1970. – 667 с.
2. Светозарова Г. И. и др. Практикум по программированию на алгоритмических языках. – М.: Наука, 1980. – 317 с.
3. Копченова Н. В., Марон И. А. Вычислительная математика в примерах и задачах. – М.: Наука, 1972. – 367 с.
4. Вычислительная техника и программирование: Метод. указания к лабораторным работам / Р. П. Герман, Б. С. Сафонов // Новосиб. электротехн. ин-т, 1991. – 38 с.
5. Герман Р. П. Программирование и применение ЭВМ. Примеры расчетов: Учеб. пособие / Новосиб. электротехн. ин-т. – Новосибирск; 1985. – 68 с.

## ТЕКСТЫ ПРОГРАММ

```
'LBR31a - лабораторная работа N 1 по численным методам
'Уточнение корня нелинейного уравнения методом
'половинного деления
DECLARE FUNCTION F (x)
COLOR 7, 1: CLS
'ввод исходных данных
READ a, b, Eps
DATA 0,1,1e-4
'обнуление счетчика
i = 0
'подготовка к вычислению Fa
x = a
'вычисление Fa - значения функции на левом конце отрезка
Fa = F(x)
'организация цикла с предусловием
DO UNTIL b - a < Eps
    'увеличение счетчика на единицу
    i = i + 1
    'вычисление координаты середины отрезка (деление от-
резка пополам)
    x = (a + b) / 2
    'вычисление Fx - значения функции в середине отрезка
    Fx = F(x)
    'проверка условия выхода из цикла
    IF ABS(Fx) < Eps / 10 THEN EXIT DO
    'проверка совпадения знаков у Fa и Fx
    IF Fa * Fx > 0 THEN
        'перенос левой границы отрезка
        a = x
        Fa = Fx
    ELSE
        'перенос правой границы отрезка
        b = x
    END IF
END IF
```

```

LOOP
'проверка правильности исходных данных
IF i = 0 THEN
    LOCATE 5, 20
    PRINT "Исходные данные неверны"
ELSE
    LOCATE 5, 15
    PRINT "Количество делений отрезка i = ";
    PRINT USING "#####"; i
    LOCATE 7, 15
    PRINT "Уточненное значение корня x = ";
    PRINT USING "##.###^"; x
    LOCATE 9, 15
    PRINT "Невязка уравнения f(x) = ";
    PRINT USING "##.###^"; Fx
END IF
END
FUNCTION F (x)
    F = x ^ 4 + 2 * x ^ 3 - x - 1
END FUNCTION

```

```

'intrpNW,QBasic - интерполирование формулами Ньютона
DECLARE SUB FormIR (n)
DECLARE SUB PrntIR (Str, n)
DECLARE SUB Shapk (Str)
DECLARE SUB FrmTab (m, k)
DECLARE SUB NWT1 (x, j0, m, dx, Y)
DECLARE SUB NWT2 (n, x, j0, m, dx, Y)
DECLARE SUB Searchj0 (lzd, j0)
DECLARE SUB Prnt ()
COLOR 7, 1: CLS
READ n
DATA 11
DIM SHARED l(n), R(n), x(3), Y(3), dY(3, 3)
CALL FormIR(n)'формирование исходных массивов
Str = 2: CALL PrntIR(Str, n)'вывод табл. заданн. функции
Str = 4: CALL Shapk(Str)'вывод шапки таблицы результатов
dx = l(2) - l(1)'вычисление шага по аргументу

```

```

DO'начало цикла
  LOCATE 2, 40: PRINT " ";
  LOCATE 2, 40
  INPUT "Жду I-заданное"; lsd'ввод lз
  IF lsd < l(1) OR lsd > l(n) THEN EXIT DO
  x = lsd: Str = Str + 1: LOCATE Str, 35
  PRINT USING "##.### " ; lsd;
  CALL Searchj0(lsd, j0)'поиск j0
  FOR m = 1 TO 3
    IF j0 <= n - m THEN
      CALL NWT1(x, j0, m, dx, Y)
    ELSE
      CALL NWT2(n, x, j0, m, dx, Y)
    END IF
    PRINT USING "##.#### " ; Y;
  NEXT m
LOOP'повторение цикла
LOCATE 23, 20
PRINT "Конец интерполирования"
'данные для таблично заданной функции
DATA 1,.5752,1.1,.6475,1.2,.7247,1.3,.8073,1.4,.8961
DATA 1.5,.9917,1.6,1.095,1.7,1.206,1.8,1.327
DATA 1.9,1.458,2,1.601
END

SUB FormlR (n)
  FOR j = 1 TO n
    READ l(j), R(j)
  NEXT j
END SUB

SUB FrmTab (m, k)
  FOR l = 0 TO m
    dY(l, 0) = R(k + l)
  NEXT l
  k1 = m
  FOR j = 1 TO m
    k1 = k1 - 1
    FOR l = 0 TO k1

```

```

        dY(l, j) = dY(l + 1, j - 1) - dY(l, j - 1)
    NEXT l
NEXT j
END SUB

```

```

SUB NWT1 (x, j0, m, dx, Y)
    Xo = l(j0)
    k = j0
    CALL FrmTab(m, k)
    q = (x - Xo) / dx
    Y = 0: q1 = 1
    FOR j = 0 TO m
        Y = Y + dY(0, j) * q1
        q1 = q1 * (q - j) / (j + 1)
    NEXT j
END SUB

```

```

SUB NWT2 (n, x, j0, m, dx, Y)
    Xm = l(j0 + 1)
    k = j0 + 1 - m
    CALL FrmTab(m, k)
    q = (x - Xm) / dx
    Y = 0: q1 = 1: l = 0
    FOR l = m TO 0 STEP -1
        Y = Y + dY(l, l) * q1
        l = l + 1
        q1 = q1 * (q + l - 1) / l
    NEXT l
END SUB

```

```

SUB PrntIR (Str, n)
    LOCATE Str, 8
    PRINT "Исходная таблица";
    Str = Str + 1
    LOCATE Str, 5
    PRINT "-----";
    LOCATE Str + 1, 5
    PRINT "N n/n l R";
    LOCATE Str + 2, 5
    PRINT "-----";

```

```

Str = Str + 3
FOR j = 1 TO n
    LOCATE Str, 5
    PRINT USING "### "; j;
    PRINT USING "##.# "; I(j);
    PRINT USING "##.####"; R(j);
    Str = Str + 1
NEXT j
END SUB

SUB Searchj0 (Isd, j0)
    j = 1: j0 = j
    DO UNTIL Isd > = I(j0) AND Isd < = I(j + 1)
        j = j + 1: j0 = j
    LOOP
END SUB

SUB Shapk (Str)
    LOCATE Str, 40
    PRINT "Результаты интерполирования";
    Str = Str + 1: LOCATE Str, 35
    FOR j = 1 TO 8: PRINT "-----"; : NEXT j
    Str = Str + 1: LOCATE Str, 53
    PRINT "R(Iз)";
    Str = Str + 1: LOCATE Str, 36: PRINT "Iз ";
    FOR j = 1 TO 7: PRINT "-----"; : NEXT j
    Str = Str + 1: LOCATE Str, 40
    PRINT TAB(45); "m = 1"; TAB(55); "m = 2"; TAB(65); "m = 3";
    Str = Str + 1: LOCATE Str, 35
    FOR j = 1 TO 8: PRINT "-----"; : NEXT j
    Str = Str + 1
END SUB

```

'intrplgr - интерполирование многочленом Лагранжа

```

DECLARE SUB FormUI (n)
DECLARE SUB PrntUI (Str, n)
DECLARE SUB Shapk (Str)
DECLARE SUB Searchj0 (Usd, j0)
DECLARE SUB Prnt ()

```

```

DECLARE SUB Lagrn (m, U, j0, L)
COLOR 7, 1: CLS
READ n
DATA 10
DIM SHARED U(n), I(n), x(3), y(3)
FormUI n'формирование исходных массивов
Str = 2: PrntUI Str, n'вывод табл. задан. функции
Str = 4: Shapk Str'вывод шапки таблицы результатов
DO'начало цикла
    LOCATE 2, 40: PRINT " ";
    LOCATE 2, 40
    INPUT "Жду U-заданное"; Usd'ввод Uз
    IF Usd < U(1) OR Usd > U(n) THEN EXIT DO
    x = Usd: Str = Str + 1: LOCATE Str, 35
    PRINT USING "###.## "; Usd;
    Searchj0 Usd, j0'поиск j0
    FOR m = 1 TO 3
        IF j0 > n - m THEN j0 = n - m
        Lagrn m, Usd, j0, L
        PRINT USING "###.## "; L;
    NEXT m
LOOP'повторение цикла
LOCATE 23, 20
PRINT "Конец интерполирования"
'данные для таблично заданной функции
DATA -3.05,-75.0,-2.05,-74.5,-0.90,-70.3
DATA 0.35,-66.7,1.10,-55.0,2.00,-20.0
DATA 3.00,5.0,4.10,17.5,5.00,35.8,6.00,38.0
END

SUB FormUI (n)
    FOR j = 1 TO n
        READ U(j), I(j)
    NEXT j
END SUB

SUB Lagrn (m, Usd, j0, L)
    L = 0
    FOR j = 0 TO m

```

```

    L1 = I(j0 + j)
    FOR k = 0 TO m
        IF k <> j THEN
            L1 = L1 * (Usd - U(j0 + k)) / (U(j0 + j) - U(j0 + k))
        END IF
    NEXT k
    L = L + L1
NEXT j
END SUB

SUB PrntUI (Str, n)
    LOCATE Str, 8
    PRINT "Исходная таблица";
    Str = Str + 1
    LOCATE Str, 5
    PRINT "-----";
    LOCATE Str + 1, 5
    PRINT "N n/n U I";
    LOCATE Str + 2, 5
    PRINT "-----";
    Str = Str + 3
    FOR j = 1 TO n
        LOCATE Str, 5
        PRINT USING "### "; j;
        PRINT USING "##.## "; U(j);
        PRINT USING "###.##"; I(j);
        Str = Str + 1
    NEXT j
END SUB

SUB Searchj0 (Usd, j0)
    j = 1: j0 = j
    DO UNTIL Usd > = U(j) AND Usd < = U(j + 1)
        j = j + 1: j0 = j
    LOOP
END SUB

SUB Shapk (Str)
    LOCATE Str, 40
    PRINT "Результаты интерполирования";

```



```

Str = Str + 1: LOCATE Str, 35
FOR j = 1 TO 8: PRINT "-----"; : NEXT j
Str = Str + 1: LOCATE Str, 53
PRINT "I(U3)";
Str = Str + 1: LOCATE Str, 36: PRINT "U3 ";
FOR j = 1 TO 7: PRINT "-----"; : NEXT j
Str = Str + 1: LOCATE Str, 40
PRINT TAB(45); "m = 1"; TAB(55); "m = 2"; TAB(65); "m = 3";
Str = Str + 1: LOCATE Str, 35
FOR j = 1 TO 8: PRINT "-----"; : NEXT j
Str = Str + 1
END SUB

```

'INTRAP - программа вычисления определенного интеграла  
'методом трапеций

DECLARE FUNCTION f (x) 'объявление процедуры-функции  
COLOR 7, 1: CLS 'подготовка экрана

LOCATE 5, 10

INPUT ; "Жду пределы интегрирования a, b"; a, b

k = 1

DO

LOCATE 7, 10: PRINT STRING\$(40, " ")

LOCATE 7, 10

INPUT ; "Жду шаг интегрирования h"; h

n = CINT((b - a) / h)

x = a: s = f(x)

x = b: s = s + f(x)

FOR j = 1 TO n - 1

x = a + h \* j: s = s + 2 \* f(x)

NEXT j

l = h / 2 \* s

LOCATE 8 + k, 10

PRINT "h = "; : PRINT USING " ##.##### "; h;

PRINT "l = "; : PRINT USING "##.#####"; l

LOCATE 9 + k, 10: PRINT STRING\$(50, " ")

LOCATE 10 + k, 10

INPUT "С другим шагом h повторить?(да - 1, нет - 0)"; reg

k = k + 1

```

LOOP WHILE reg = 1
END

FUNCTION f (x)
    f = EXP(.84 * SIN(1.24 * x))
END FUNCTION

```

```

'GaussQ - решение системы линейных алгебраических
уравнений
'методом Гаусса (последовательного исключения неиз-
вестных)
DECLARE SUB FormA (n)
DECLARE SUB PrntA (Str, n)
DECLARE SUB SearchI (k, n, I)
DECLARE SUB SwapI (k, n, I)
DECLARE SUB IskIXk (k, n)
DECLARE SUB Obratn (n)
DECLARE SUB PrntX (n)
DECLARE SUB wart ()
'подготовка экрана
COLOR 7, 1: CLS
'ввод размерности системы
READ n
DATA 6
'описание глобальных (доступных всем процедурам) массивов
DIM SHARED a(n, n + 1), x(n)
'формирование расширенной матрицы системы
CALL FormA(n)
'вывод исходной матрицы на экран
Str = 2
LOCATE Str, 10
PRINT "Исходная расширенная матрица системы";
Str = Str + 2
CALL PrntA(Str, n)
'ожидание
wart
'основной алгоритм
FOR k = 1 TO n - 1
    'поиск максимального элемента в k-м столбце

```

```

CALL SearchI(k, n, I)
'замена строк матрицы
CALL SwapI(k, n, I)
'проверка матрицы на особенность
IF ABS(a(k, k) / a(1, 1)) < 1E-08 THEN PRINT "Матр.
особенн.": STOP
'исключение k-го неизвестного
CALL IskIXk(k, n)
'вывод матрицы на экран
CLS : Str = 2: LOCATE Str, 10
PRINT "исключение "; : PRINT USING "#####"; k;
PRINT "-го неизвестного";
Str = Str + 2
CALL PrntA(Str, n)
wart
NEXT k
IF ABS(a(n, n)) < 1E-08 THEN PRINT "Матр. особенн.": STOP
'обратный ход (вычисление значений неизвестных)
CALL Obratn(n)
'вывод значений неизвестных на экран
CALL PrntX(n)
'блоки данных для контрольного примера
DATA 6,-3, 2, 1, -1, 1, 11
DATA -3,-7, 0, 4, -2, 1, -5
DATA 4,-3, 6,-1, 2, 1, 28
DATA 2, 4, 5,-7, -3, 2, -6
DATA -1, 5,-4, 0, 8,-2, 25
DATA 3, 0, 4,-2, 5,-6, -4
END

SUB FormA (n)
  FOR i = 1 TO n
    FOR j = 1 TO n + 1
      READ a(i, j)
    NEXT j
  NEXT i
END SUB

SUB IskIXk (k, n)

```

```

    FOR i = k + 1 TO n
        s = a(i, k) / a(k, k)
        a(i, k) = 0
        FOR j = k + 1 TO n + 1
            a(i, j) = a(i, j) - s * a(k, j)
        NEXT j
    NEXT i
END SUB

SUB Obratn (n)
    x(n) = a(n, n + 1) / a(n, n)
    FOR i = n - 1 TO 1 STEP -1
        s = a(i, n + 1)
        FOR j = i + 1 TO n
            s = s - a(i, j) * x(j)
        NEXT j
        x(i) = s / a(i, i)
    NEXT i
END SUB

SUB PrntA (Str, n)
    LOCATE Str, 5
    FOR i = 1 TO n
        LOCATE Str + i, 5
        FOR j = 1 TO n + 1
            PRINT USING "###.## "; a(i, j);
        NEXT j
    NEXT i
END SUB

SUB PrntX (n)
    CLS
    FOR i = 1 TO n
        LOCATE 5 + i, 20
        PRINT "x(";
        PRINT USING "##"; i;
        PRINT ") = ";
        PRINT USING "##.##"; x(i)
    NEXT i
END SUB

```

```

SUB Searchl (k, n, l)
  s = 0
  FOR i = k TO n
    t = ABS(a(i, k))
    IF s < t THEN s = t: l = i
  NEXT i
END SUB

SUB Swapkl (k, n, l)
  FOR j = k TO n + 1
    SWAP a(k, j), a(l, j)
  NEXT j
END SUB

SUB wart
  LOCATE 25, 30
  PRINT "Для продолжения нажмите любую клавишу";
  k$ = "": WHILE k$ = ""
    k$ = INKEY$: WEND
END SUB

```

'Численное решение обыкновенного дифференциального  
'уравнения первого порядка методом Эйлера

```

DECLARE SUB saglv (x0!, xk!, y0!, h!)
DECLARE SUB shapk ( )
DECLARE SUB prav (x!, y!, f!)
DECLARE SUB prntabl (str!, ny!, x!(), y!(), y1!(), eps!())
DECLARE SUB cherta (str!, psz!, dln!)
COLOR 7, 1: CLS
READ x0, xk, y0, h
DATA 0,1,0,.1
ny = CINT((xk - x0) / h)'определение кол-ва строк таблицы
DIM x(ny), y(ny), y1(ny), eps(ny)
saglv x0, xk, y0, h'вывод заголовка
shapk'вывод шапки таблицы
str = 9
i = 0
x(i) = x0: y(i) = y0: y1(i) = y0: eps(i) = 0
FOR k = 1 TO 2'организация цикла для двойного просчета

```

```

dx = h / k'определение шага интегрирования
FOR i = 1 TO ny'организация цикла для решения
    x = x(i - 1)
    IF k = 1 THEN
        y = y(i - 1)
    ELSE
        y = y1(i - 1)
    END IF
    FOR j = 1 TO k
        prav x, y, f
        x = x + dx: y = y + dx * f
    NEXT j
    x(i) = x0 + h * i
    IF k = 1 THEN
        y(i) = y
    ELSE
        y1(i) = y
    END IF
NEXT i
NEXT k
FOR i = 1 TO ny
    eps(i) = y(i) - y1(i)
NEXT i
prntabl str, ny, x(), y(), y1(), eps()
END

SUB cherta (str, psz, dln)
    LOCATE str, psz
    PRINT STRING$(dln, "-")
END SUB

SUB prav (x, y, f)
    f = 1 + .4 * y * SIN(x) - 2 * y ^ 2
END SUB

SUB prntabl (str, ny, x(), y(), y1(), eps())
    FOR i = 0 TO ny
        LOCATE str + i, 20
        PRINT USING "### "; i;
        PRINT USING "##.# "; x(i);
    
```

```

        PRINT USING " ##.#### "; y(i); y1(i);
        PRINT USING " ##.##^ ^ ^ ^"; eps(i);
    NEXT i
END SUB

SUB saglv (x0, xk, y0, h)
    LOCATE 2, 20
    PRINT "РЕЗУЛЬТАТЫ ЧИСЛЕННОГО РЕШЕНИЯ
ОБЫКНОВЕННОГО"
    PRINT TAB(20); "ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ
МЕТОДОМ ЭЙЛЕРА"
    PRINT
    PRINT TAB(25); "x0 = "; x0; " xk = "; xk; " y0 = "; y0; " h = "; h
END SUB

SUB shapk
    cherta 6, 20, 50
    LOCATE 7, 20
    PRINT TAB(20); " N п/п x y y* eps"
    cherta 8, 20, 50
END SUB

```

'Численное решение обыкновенного дифференциального  
'уравнения первого порядка методом Рунге-Кутта

```

DECLARE SUB saglv (x0!, xk!, y0!, h!)
DECLARE SUB shapk ()
DECLARE SUB prav (x!, y!, f!)
DECLARE SUB prntabl (str!, ny!, x!(), y!(), y1!(), eps!())
DECLARE SUB cherta (str!, psz!, dln!)
COLOR 7, 1: CLS
READ x0, xk, y0, h
DATA 0,1,0,.1
ny = CINT((xk - x0) / h)'определение кол-ва строк таблицы
DIM x(ny), y(ny), y1(ny), eps(ny)
saglv x0, xk, y0, h'вывод заголовка
shapk'вывод шапки таблицы
str = 9
i = 0
x(i) = x0: y(i) = y0: y1(i) = y0: eps(i) = 0

```

'далее должен быть реализован алгоритм формирования  
'массивов с результатами решения дифференциального  
'уравнения методом Рунге-Кутты

```
FOR k = 1 TO 2
  dx = h / k
  FOR i = 1 TO ny
    xi = x(i - 1)
    IF k = 1 THEN
      yi = y(i - 1)
    ELSE
      yi = y1(i - 1)
    END IF
    FOR j = 1 TO k
      x = xi: y = yi
      prav x, y, f
      k1 = dx * f
      x = xi + dx / 2: y = yi + k1 / 2
      prav x, y, f
      k2 = dx * f
      y = yi + k2 / 2
      prav x, y, f
      k3 = dx * f
      x = xi + dx: y = yi + k3
      prav x, y, f
      k4 = dx * f
      dy = (k1 + 2 * (k2 + k3) + k4) / 6
      xi = x: yi = yi + dy
    NEXT j
    x(i) = x0 + h * i
    IF k = 1 THEN
      y(i) = yi
    ELSE
      y1(i) = yi
    END IF
  NEXT i
NEXT k
FOR i = 1 TO ny
  eps(i) = ABS(y(i) - y1(i))
```



```

NEXT i
prntabl str, ny, x(), y(), y1(), eps()
END

SUB cherta (str, psz, dln)
    LOCATE str, psz
    PRINT STRING$(dln, "-")
END SUB

SUB prav (x, y, f)
    
$$f = 1 + .4 * y * \sin(x) - 2 * y^2$$

END SUB

SUB prntabl (str, ny, x(), y(), y1(), eps())
    FOR i = 0 TO ny
        LOCATE str + i, 20
        PRINT USING "### "; i;
        PRINT USING "##.# "; x(i);
        PRINT USING " ##.##### "; y(i); y1(i);
        PRINT USING " ##.##^ ^ ^ ^"; eps(i);
    NEXT i
END SUB

SUB saglv (x0, xk, y0, h)
    LOCATE 2, 20
    PRINT "РЕЗУЛЬТАТЫ ЧИСЛЕННОГО РЕШЕНИЯ  
ОБЫКНОВЕННОГО"  
PRINT TAB(17); "ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ  
МЕТОДОМ ";  
PRINT "РУНГЕ-КУТТА"  
PRINT  
PRINT TAB(25); "x0 = "; x0; " xk = "; xk; " y0 = "; y0; " h = "; h
END SUB

SUB shapk
    cherta 6, 20, 50
    LOCATE 7, 20
    PRINT TAB(20); " N п/п x y y* eps"
    cherta 8, 20, 50
END SUB

```

```

'Численное решение обыкновенного дифференциального
'уравнения второго порядка методом Эйлера
DECLARE SUB saglv (x0, xk, y0, y10, h, k)
DECLARE SUB shapk ()
DECLARE SUB prav (x, y, y1, f1, f2)
DECLARE SUB prnstr (i, x, y, y1)
DECLARE SUB cherta (str, psz, dln)
COLOR 7, 1: CLS
READ x0, xk, y0, y10, h, k
DATA 0,1,2.6,3.2,.1,8
ny = CINT((xk - x0) / h)'определение кол-ва строк таблицы
saglv x0, xk, y0, y10, h, k'вывод заголовка
shapk'вывод шапки таблицы
str = 9
i = 0
x = x0: y = y0: y1 = y10
prnstr i, x, y, y1
dx = h / k'определение шага интегрирования
FOR i = 1 TO ny'организация цикла для решения
    xi = x: yi = y: y1i = y1
    FOR j = 1 TO k
        prav x, y, y1, f1, f2
        x = xi + dx * j: y = y + dx * f1: y1 = y1 + dx * f2
    NEXT j
    xi = x0 + h * i: x = xi
    prnstr i, x, y, y1
NEXT i
END

SUB cherta (str, psz, dln)
    LOCATE str, psz
    PRINT STRING$(dln, "-")
END SUB

SUB prav (x, y, y1, f1, f2)
    f1 = y1
    f2 = 2 * SIN(x) + 3 * y1 - 2 * y
END SUB

SUB prnstr (i, x, y, y1)

```

```

LOCATE i + 10, 20
PRINT USING "#### "; i;
PRINT USING "###.## "; x;
PRINT USING "###.### "; y; y1
END SUB

SUB saglv (x0, xk, y0, y10, h, k)
LOCATE 2, 20
PRINT "РЕЗУЛЬТАТЫ ЧИСЛЕННОГО РЕШЕНИЯ
ОБЫКНОВЕННОГО"
PRINT TAB(25); "ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ
ВТОРОГО"
PRINT TAB(30); "ПОРЯДКА МЕТОДОМ ЭЙЛЕРА"
PRINT
PRINT TAB(25); "x0 = "; x0; " xk = "; xk; " y0 = "; y0;
PRINT "y10 = "; y10; " h = "; h
END SUB

SUB shapk
cherta 6, 20, 50
LOCATE 7, 20
PRINT TAB(20); " N п/п x y y* eps"
cherta 8, 20, 50
END SUB

```

'Численное решение обыкновенного дифференциального  
'уравнения второго порядка методом Рунге-Кутта

```

DECLARE SUB saglv (x0!, xk!, y0!, h!)
DECLARE SUB shapk ()
DECLARE SUB prav (x, y, y1, f, f1)
DECLARE SUB prnstr (str, i, xi, yi, y1i)
DECLARE SUB cherta (str!, psz!, dln!)
COLOR 7, 1: CLS
READ x0, xk, y0, y10, h
DATA 0,1,2.6,3.2,.1
ny = CINT((xk - x0) / h)'определение кол-ва строк таблицы
saglv x0, xk, y0, h'вывод заголовка
shapk'вывод шапки таблицы
str = 9: dx = h

```

```

i = 0
xi = x0: yi = y0: y1i = y10
prnstr str, i, xi, yi, y1i
FOR i = 1 TO ny
    x = xi: y = yi: y1 = y1i
    prav x, y, y1, f, f1
    k1y = dx * f: k1y1 = dx * f1
    x = xi + dx / 2: y = yi + k1y / 2
    y1 = y1i + k1y1 / 2
    prav x, y, y1, f, f1
    k2y = dx * f: k2y1 = dx * f1
    y = yi + k2y / 2: y1 = y1i + k2y1 / 2
    prav x, y, y1, f, f1
    k3y = dx * f: k3y1 = dx * f1
    x = xi + dx: y = yi + k3y
    y1 = y1i + k3y1
    prav x, y, y1, f, f1
    k4y = dx * f: k4y1 = dx * f1
    dy = (k1y + 2 * (k2y + k3y) + k4y) / 6
    dy1 = (k1y1 + 2 * (k2y1 + k3y1) + k4y1) / 6
    xi = x0 + h * i: yi = yi + dy: y1i = y1i + dy1
    prnstr str, i, xi, yi, y1i
NEXT i
END

SUB cherta (str, psz, dln)
    LOCATE str, psz
    PRINT STRING$(dln, "-")
END SUB

SUB prav (x, y, y1, f, f1)
    f = y1
    f1 = 2 * SIN(x) - 2 * y + 3 * y1
END SUB

SUB prnstr (str, i, xi, yi, y1i)
    LOCATE str + i, 20
    PRINT USING "#####" ; i;
    PRINT USING "##.##" ; xi;
    PRINT USING "###.###" ; yi; y1i

```

END SUB

SUB saglv (x0, xk, y0, h)

LOCATE 2, 20

PRINT "РЕЗУЛЬТАТЫ ЧИСЛЕННОГО РЕШЕНИЯ  
ОБЫКНОВЕННОГО"

PRINT TAB(22); "ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ  
ВТОРОГО"

PRINT TAB(25); "ПОРЯДКА МЕТОДОМ РУНГЕ-КУТТА "

PRINT

PRINT TAB(25); "x0 = "; x0; " xk = "; xk; " y0 = "; y0; " h  
= "; h

END SUB

SUB shapk

cherta 6, 20, 50

LOCATE 7, 20

PRINT TAB(20); " N п/п x y y\* eps"

cherta 8, 20, 50

END SUB

'Obchk - программа расчета заготовки для конической обечайки

DECLARE SUB Sagl ()

DECLARE SUB Prnisch (dmal, dbol, hbol, delt)

DECLARE SUB Prnres (rmal, rbol, alfgrd, L, hmal, k)

CONST pi = 3.14159

COLOR 7, 1: CLS

'ввод (формирование) исходных данных

READ dmal, dbol, hbol, delt, eps

DATA 100,200,75,10,.001

'определение начального приближения корня уравнения

beta = ATN((dbol - dmal) / 2 / hbol)

k = 0

DO

k = k + 1

'вычисление очередного приближения корня

arg = ((dbol - dmal) / 2 - delt \* COS(beta))

arg = arg / (hbol - delt \* SIN(beta))

beta1 = ATN(arg)

```

    dltbeta = ABS(beta - beta1)
    beta = beta1
LOOP WHILE dltbeta > eps
'определение размеров заготовки
rma1 = (dma1 + delt * COS(beta)) / 2 / SIN(beta)
rbo1 = (dbol - delt * COS(beta)) / 2 / SIN(beta)
alf = 2 * pi * SIN(beta)
IF alf > = pi THEN
    L = 2 * rbo1: hma1 = rbo1 * (1 - COS(alf / 2))
ELSE
    L = 2 * rbo1 * SIN(alf / 2): hma1 = rbo1 - rma1 * COS(alf / 2)
END IF
alfgrd = 180 / pi * alf
'вывод заголовка и шапки таблицы
Sagl
'вывод исходных данных
Prnisch dma1, dbol, hbo1, delt
'вывод результатов
Prnres rma1, rbo1, alfgrd, L, hma1, k
END

SUB Prnisch (dma1, dbol, hbo1, delt)
    LOCATE 7, 2
    PRINT USING "#####.# "; dma1; dbol; hbo1; delt;
END SUB

SUB Prnres (rma1, rbo1, alfgrd, L, hma1, k)
    LOCATE 7, 34
    PRINT USING "#####.# "; rma1; rbo1; alfgrd; L; hma1;
    PRINT USING "#####"; k
END SUB

SUB Sagl
LOCATE 3, 15
PRINT "Расчет заготовки для конической обечайки"
FOR i = 1 TO 16: PRINT "_____"; : NEXT i
PRINT TAB(5); "d"; TAB(13); "D"; TAB(21); "H"; TAB(29);
PRINT "delt"; TAB(37); "r"; TAB(45); "R"; TAB(53);
PRINT "alf"; TAB(61); "L"; TAB(69); "h"; TAB(77); "k"
FOR i = 1 TO 16: PRINT "_____"; : NEXT i

```

END SUB

'RGR42 - расчетно-графическая работа N 2

'по спец. главам высшей математики

'Моделирование переходного процесса

'в электрической цепи постоянного тока

DECLARE SUB Shapk (E, R1, R2, R3, L, C, T0, T9, Tpr, Dt)

DECLARE SUB FrmNach (T0, E, R1, R2, R3, I1, I2, R4, R5)

DECLARE SUB PrnRes (jres, I1, I2)

DECLARE SUB Integr (E, R1, R2, R3, L, C, Dt, R4, R5)

DECLARE SUB Prav (E, R1, R2, R3, L, C, Uc, I3, R4, R5, F1, F2)

DECLARE SUB Setka (x0, y0, xk, yk, tmin, tmax, imin, imax, Umin, Umax)

DECLARE SUB Graf (n, x0, y0, mx, my, x(), y(), ymin, col)

DECLARE SUB wart (str)

COLOR 7, 1: CLS

'ввод исходных данных

READ E, R1, R2, R3, L, C

DATA 20,10,10,10,.2,2000e-06

READ T0, T9, Tpr, Dt, dtgrf

DATA 0,.20,.02,.0001,.001

'вывод исходных данных и шапки таблицы результатов

Shapk E, R1, R2, R3, L, C, T0, T9, Tpr, Dt

'определение числа результатов для графиков

ngr = CINT(T9 / dtgrf)

DIM SHARED t(2), Uc(2), I3(2)'описание массивов

'описание массивов для построения графиков

DIM tgrf(ngr), Ucgrf(ngr), ic(ngr), iL(ngr)

'формир. нач. значен.

FrmNach T0, E, R1, R2, R3, I1, I2, R4, R5

'вывод строки результатов

jres = 0: jgrf = 0

PrnRes jres, I1, I2: TSH = 0

'запоминание значений в массивах для графиков

tgrf(jgrf) = t(1): Ucgrf(jgrf) = Uc(1)

ic(jgrf) = I2: iL(jgrf) = I3(1): tshgrf = 0

DO

    'интегрирование

    Integr E, R1, R2, R3, L, C, Dt, R4, R5

```

'переприсваивание
t(1) = t(2): Uc(1) = Uc(2): I3(1) = I3(2)
TSH = TSH + Dt: tshgrf = tshgrf + Dt
IF tshgrf + Dt * .1 > dtgrf THEN
    'запоминание значений в массивах для графиков
    I2 = (E - I3(1) * R1 - Uc(1)) / R5
    I1 = I2 + I3(1)
    jgrf = jgrf + 1: tgrf(jgrf) = t(1): Ucgrf(jgrf) = Uc(1)
    ic(jgrf) = I2: iL(jgrf) = I3(1): tshgrf = 0
END IF
IF TSH + .1 * Dt > Tpr THEN
    I2 = (E - I3(1) * R1 - Uc(1)) / R5
    I1 = I2 + I3(1)
    jres = jres + 1
    'вывод строки результатов
    PrnRes jres, I1, I2: TSH = 0
END IF
LOOP WHILE t(1) + .1 * Dt < T9
wart 24
WIDTH 80, 43
SCREEN 9: COLOR 7, 0: CLS
x0 = 104: xk = 504: y0 = 80: yk = 280
tmin = 0: tmax = T9
Umin = 10: Umax = 20
imin = -.4: imax = 1.2
Setka x0, y0, xk, yk, tmin, tmax, imin, imax, Umin, Umax
wart 42'END
'подготовка к построению графика Uc(t)
'определение масштабных коэффициентов
mt = (xk - x0) / T9
mUc = (yk - y0) / (Umax - Umin)
mi = (yk - y0) / (imax - imin)
'построение графика Uc(t)
Graf ngr, x0, yk, mt, mUc, tgrf(), Ucgrf(), Umin, 13
wart 42
'построение графика ic(t)
Graf ngr, x0, yk, mt, mi, tgrf(), ic(), imin, 14
wart 42

```



```

'построение графика iL(t)
Graf ngr, x0, yk, mt, mi, tgrf(), iL(), imin, 12
'изображение условных обозначений
FOR j = 0 TO 2
    LINE (280, 11 + j)-(320, 11 + j), 13
NEXT j
LOCATE 2, 44
PRINT "Uc(t)"
FOR j = 0 TO 2
    LINE (280, 27 + j)-(320, 27 + j), 14
NEXT j
LOCATE 4, 44
PRINT "ic(t)"
FOR j = 0 TO 2
    LINE (280, 43 + j)-(320, 43 + j), 12
NEXT j
LOCATE 6, 44
PRINT "iL(t)"
wart 42
LOCATE 42, 25
PRINT "К О Н Е Ц  Р А Б О Т Ы !! "
END

SUB FrmNach (T0, E, R1, R2, R3, I1, I2, R4, R5)
    t(1) = T0: Uc(1) = E: I3(1) = 0
    R4 = R1 * R2 + R1 * R3 + R2 * R3: R5 = R1 + R3
    I2 = (E - I3(1) * R1 - Uc(1)) / R5
    I1 = I2 + I3(1)
END SUB

SUB Graf (n, x0, y0, mx, my, x(), y(), ymin, col)
    x1 = x0: y1 = y0 - CINT(my * (y(0) - ymin))
    FOR j = 1 TO n
        x2 = x0 + CINT(mx * x(j))
        y2 = y0 - CINT(my * (y(j) - ymin))
        LINE (x1, y1)-(x2, y2), col
        IF y1 <> y2 THEN
            LINE (x1 + 1, y1)-(x2 + 1, y2), col
        ELSE

```

```

        LINE (x1, y1 - 1)-(x2, y2 - 1), col
    END IF
    x1 = x2: y1 = y2
NEXT j
END SUB

SUB Integr (E, R1, R2, R3, L, C, Dt, R4, R5)
    t = t(1): Uc = Uc(1): I3 = I3(1)
    CALL Prav(E, R1, R2, R3, L, C, Uc, I3, R4, R5, F1, F2)
    K1UC = Dt * F1: K1I3 = Dt * F2
    t = t(1) + Dt / 2: Uc = Uc(1) + K1UC / 2: I3 = I3(1) + K1I3 / 2
    CALL Prav(E, R1, R2, R3, L, C, Uc, I3, R4, R5, F1, F2)
    K2UC = Dt * F1: K2I3 = Dt * F2
    Uc = Uc(1) + K2UC / 2: I3 = I3(1) + K2I3 / 2
    CALL Prav(E, R1, R2, R3, L, C, Uc, I3, R4, R5, F1, F2)
    K3UC = Dt * F1: K3I3 = Dt * F2
    t = t + Dt: Uc = Uc(1) + K3UC: I3 = I3(1) + K3I3
    CALL Prav(E, R1, R2, R3, L, C, Uc, I3, R4, R5, F1, F2)
    K4UC = Dt * F1: K4I3 = Dt * F2
    DUC = (K1UC + 2 * (K2UC + K3UC) + K4UC) / 6
    DI3 = (K1I3 + 2 * (K2I3 + K3I3) + K4I3) / 6
    Uc(2) = Uc(1) + DUC: I3(2) = I3(1) + DI3: t(2) = t(1) + Dt
END SUB

SUB Prav (E, R1, R2, R3, L, C, Uc, I3, R4, R5, F1, F2)
    F1 = (E - I3 * R1 - Uc) / C / R5
    F2 = (E * R2 + Uc * R1 - I3 * R4) / L / R5
END SUB

SUB PrnRes (jres, I1, I2)
    PRINT TAB(3); USING " ### "; jres;
    PRINT TAB(10); USING " ##.### "; t(1);
    PRINT USING " ###.## "; Uc(1);
    PRINT USING " ##.### "; I1; I2; I3(1)
END SUB

SUB Setka (x0, y0, xk, yk, tmin, tmax, imin, imax, Umin, Umax)
    'выделение прямоугольника для графика
    LINE (x0, y0)-(xk, yk), 7, BF
    'построение горизонтальных линий сетки

```

```

FOR j = 0 TO 10
    y = yk - 20 * j
    LINE (x0, y)-(xk, y), 2
NEXT j
'построение вертикальных линий сетки
FOR j = 0 TO 10
    x = x0 + 40 * j
    LINE (x, y0)-(x, yk), 2
NEXT j
'выделение масштабной линейки для Uc
x = x0 - 1
LINE (x, y0 - 4)-(x, yk + 4), 5
LINE (x + 1, y0 - 4)-(x + 1, yk + 4), 5
LINE (x + 2, y0 - 4)-(x + 2, yk + 4), 5
'построение выступов
x1 = x0 - 5
x2 = x0
FOR i = 0 TO 10
    y = yk - 20 * i
    FOR k = 0 TO 1
        LINE (x1, y + k)-(x2, y + k), 5
    NEXT k
NEXT i
'выделение масштабной линейки для i
x = xk - 1
LINE (x, y0 - 4)-(x, yk + 4), 4
LINE (x + 1, y0 - 4)-(x + 1, yk + 4), 4
LINE (x + 2, y0 - 4)-(x + 2, yk + 4), 4
'построение выступов
x1 = xk - 5
x2 = xk + 5
FOR i = 0 TO 10
    y = yk - 20 * i
    FOR k = 0 TO 1
        LINE (x1, y + k)-(x2, y + k), 4
    NEXT k
NEXT i
str = y0 \ 8 - 1: psz = x0 \ 8 + 1

```

```

LOCATE str, psz: PRINT "Uc, B";
str = y0 \ 8 - 1: psz = xk \ 8 - 1
LOCATE str, psz: PRINT "i, A";
'обозначение масштаба для линейки Uc
dUgr = (Umax - Umin) / 5
FOR j = 0 TO 5
    str = yk \ 8 - j * 5
    U = Umin + dUgr * j
    IF ABS(U) < .00001 THEN U = 0
    LOCATE str, 5
    PRINT USING "###.# "; U;
NEXT j
'обозначение масштаба для линейки i
digr = (imax - imin) / 5
FOR j = 0 TO 5
    str = yk \ 8 - j * 5
    i = imin + digr * j
    IF ABS(i) < .00001 THEN i = 0
    LOCATE str, 66
    PRINT USING "###.###"; i;
NEXT j
'изображение масштаба для линейки t
dtgr = (tmax - tmin) / 5
FOR j = 0 TO 5
    psz = x0 \ 8 - 2 + 10 * j
    LOCATE yk \ 8 + 3, psz
    t = j * dtgr
    PRINT USING "##.###"; t;
NEXT j
'выделение и обозначение масштабной линейки для t
FOR i = 1 TO 2
    LINE (x0, yk - 1 + i)-(xk + 15, yk - 1 + i), 2
    NEXT i
    LOCATE yk \ 8 + 3, xk \ 8 + 5: PRINT "t, c"
END SUB

SUB Shapk (E, R1, R2, R3, L, C, T0, T9, Tpr, Dt)
    LOCATE 2, 25
    PRINT "ИСХОДНЫЕ ДАННЫЕ": PRINT

```

```

PRINT TAB(10); " E = "; E; " R1 = "; R1; " R2 = ";
PRINT R2; " R3 = "; R3; " L = "; L; " C = "; C
PRINT TAB(14); " T0 = "; T0; " T9 = "; T9;
PRINT " TPR = "; Tpr; " DT = "; Dt
PRINT
FOR j = 1 TO 16: PRINT "-----"; : NEXT j
PRINT TAB(5); "N п/п";
PRINT TAB(15); "T"; TAB(27); "UC"; TAB(39); "I1"; TAB(51);
PRINT "I2"; TAB(63); "I3"
FOR j = 1 TO 16: PRINT "-----"; : NEXT j
END SUB

SUB wart (str)
LOCATE str, 30
PRINT "Для продолжения нажмите любую клавишу";
k$ = "": WHILE k$ = ""
k$ = INKEY$: WEND
END SUB

```

## ОГЛАВЛЕНИЕ

1. Численное решение нелинейных уравнений .....	3
1.1. Постановка задачи и основные определения .....	3
1.2. Отделение корней уравнения.....	4
1.3. Уточнение корня уравнения .....	5
1.4. Пример решения инженерной задачи .....	13
2. Интерполирование таблично заданных функций .....	17
2.1. Постановка задачи и основные определения .....	17
2.2. Интерполирование функций с равноотстоящими узлами .....	19
2.3. Пример использования формул Ньютона.....	21
2.4. Интерполирование таблично-заданных функций с неравноотстоящими узлами .....	26
3. Вычисление определенных интегралов .....	31
3.1. Постановка задачи и основные определения .....	31
3.2. Вычисление определенного интеграла методом трапеций .....	32
3.3.Вычисление определенного интеграла методом Симпсона .....	36
3.4. Обеспечение заданной точности .....	38
4. Решение системы линейных алгебраических уравнений .....	39
4.1. Постановка задачи и основные определения .....	39
4.2. Метод Гаусса (последовательных исключений) .....	41
4.3. Косвенные методы решения систем линейных алгебраических уравнений .....	45
5. Решение обыкновенных дифференциальных уравнений численными методами.....	52
5.1. Постановка задачи и основные определения .....	52
5.2. Метод Эйлера.....	53
5.3. Метод Рунге-Кутты .....	56
5.4. Численное решение системы обыкновенных дифференциальных уравнений первого порядка.....	58
5.5. Численное решение обыкновенных дифференциальных уравнений высших порядков.....	60
5.6. Пример решения инженерной задачи .....	62
Список литературы .....	66
Приложение. Тексты программ .....	67