

Поиск корней нелинейных уравнений

Поиск корней нелинейных уравнений (1)

Пусть имеется уравнение вида $f(x) = 0$, где $f(x)$ — заданная алгебраическая или трансцендентная функция.

Решить уравнение — значит найти все его корни, то есть те значения x , которые обращают уравнение в тождество.

Если уравнение достаточно сложно, то задача точного определения корней является в некоторых случаях нерешаемой. Поэтому ставится задача найти такое приближенное значение корня $x_{\text{пр}}$, которое отличается от точного значения корня x^* на величину, по модулю не превышающую указанной точности (малой положительной величины) ε , то есть

$$|x^* - x_{\text{пр}}| < \varepsilon$$

Величину ε также называют **допустимой ошибкой**, которую можно задать по своему усмотрению.

Этапы приближенного решения нелинейных уравнений

Приближенное решение уравнения состоит из двух этапов:

- Отделение корней, то есть нахождение интервалов из области определения функции $f(x)$, в каждом из которых содержится только один корень уравнения $f(x) = 0$.
- Уточнение корней до заданной точности.

Поиск корней нелинейных уравнений (2)

Аналитическое отделение корней

Аналитическое отделение корней основано на следующих теоремах.

Теорема 1. Если непрерывная функция $f(x)$ принимает на концах отрезка $[a; b]$ значения разных знаков, т.е.

$$f(a)f(b) < 0$$

то на этом отрезке содержится по крайней мере один корень уравнения.

Теорема 2. Если непрерывная на отрезке $[a; b]$ функция $f(x)$ принимает на концах отрезка значения разных знаков, а производная $f'(x)$ сохраняет знак внутри указанного отрезка, то внутри отрезка существует единственный корень уравнения $f(x) = 0$.

Уточнение корней

Для уточнения корней может использоваться один из следующих методов:

- Метод последовательных приближений (метод итераций)
- Метод Ньютона (метод касательных)
- Метод секущих (метод хорд)
- Метод половинного деления (метод дихотомии)

Метод последовательных приближений (метод итераций)

Метод итерации — численный метод решения математических задач, используемый для приближённого решения алгебраических уравнений и систем. Суть метода заключается в нахождении по приближённому значению величины следующего приближения (являющегося более точным).

Метод позволяет получить решение с заданной точностью в виде предела последовательности итераций. Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения решения.

Функциональное уравнение может быть записано в виде $x = f(x)$

Функцию $f(x)$ называют сжимающим отображением.

Последовательность чисел x_0, x_1, \dots, x_n называется итерационной, если для любого номера $n > 0$ элемент x_n выражается через элемент x_{n-1} по рекуррентной формуле

$$x_n = f(x_{n-1})$$

а в качестве x_0 взято любое число из области задания функции $f(x)$.

Метод простой итерации

Необходимо записать уравнение в виде

$$x = \phi(x), \quad |\phi''(x)| < 1 \text{ на } [a, b]$$

Затем задать начальное приближение x_0 и организовать следующую итерацию:

$$x_{k+1} = \phi(x_k), \quad k = 0, 1, 2, \dots$$

Вычисление прекратить, если

$$|x_{k+1} - x_k| < \epsilon$$

Пусть задана функция $f(x)$, требуется найти корни уравнения

$$f(x) = 0. \quad (2.8)$$

Метод простых итераций (последовательных приближений) является наиболее общим, и многие другие методы можно представить как некоторую вариацию метода простых итераций.

Представим уравнение (2.8) в виде

$$x = \psi(x). \quad (2.9)$$

Это можно сделать, например, прибавив x к обеим частям уравнения (2.9).

Рассмотрим последовательность чисел x_i , которая определяется следующим образом:

$$x_{k+1} = \psi(x_k), \quad x_0 \text{ принадлежит } [a; b].$$

Метод простых итераций имеет следующую наглядную геометрическую интерпретацию (рис. 2.10). Решением уравнения (2.9) будет абсцисса точки пересечения прямой $y = x$ с кривой $y = \psi(x)$. При выполнении итераций значение функции $\psi(x)$ в точке x_i необходимо отложить по оси абсцисс. Это можно сделать, если провести горизонталь до пересечения с прямой $y = x$ и из точки их пересечения опустить перпендикуляр на ось абсцисс. На рис. 2.10 показаны разные ситуации: а) сходимость к корню односторонняя; б) сходимость с разных сторон.

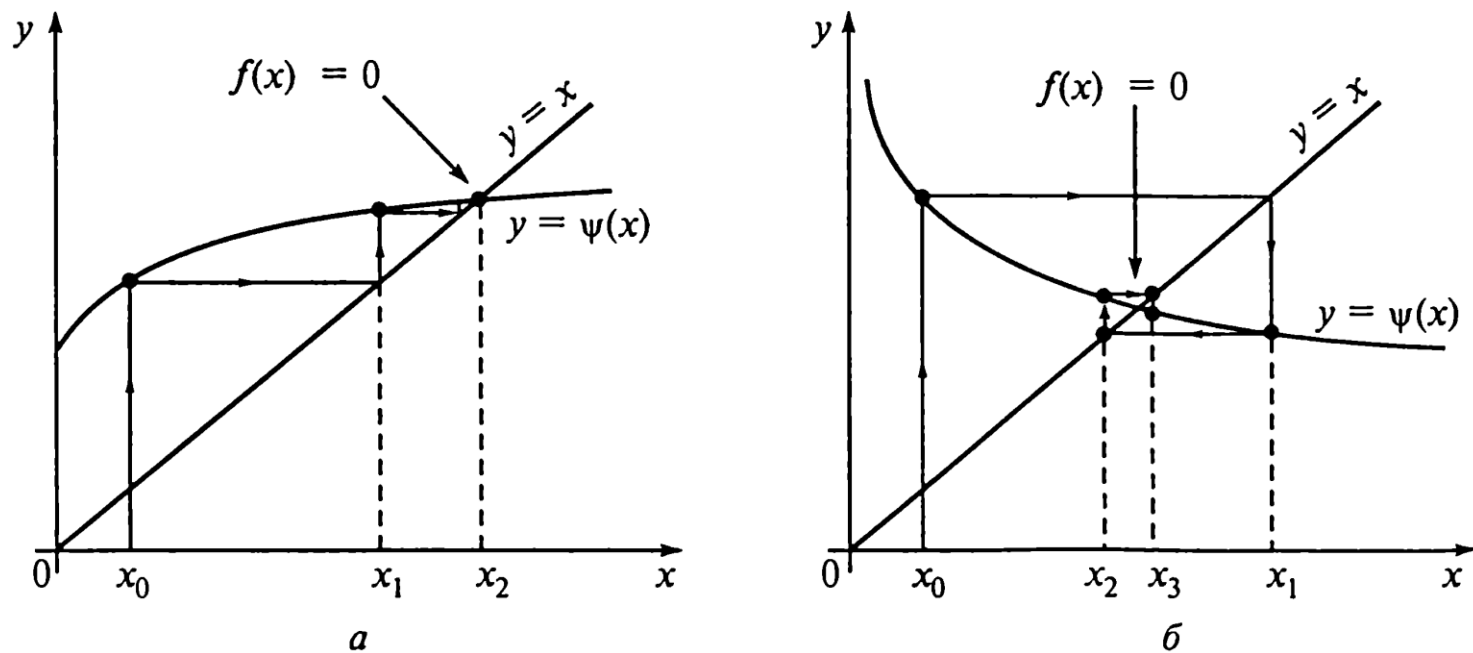


Рис. 2.10. Приближение к корню методом простой итерации

Сходимость процесса приближения к корню в значительной степени определяется видом зависимости $\psi(x)$. На рис. 2.11 показан расходящийся процесс, при котором метод простой итерации не находит решения уравнения.

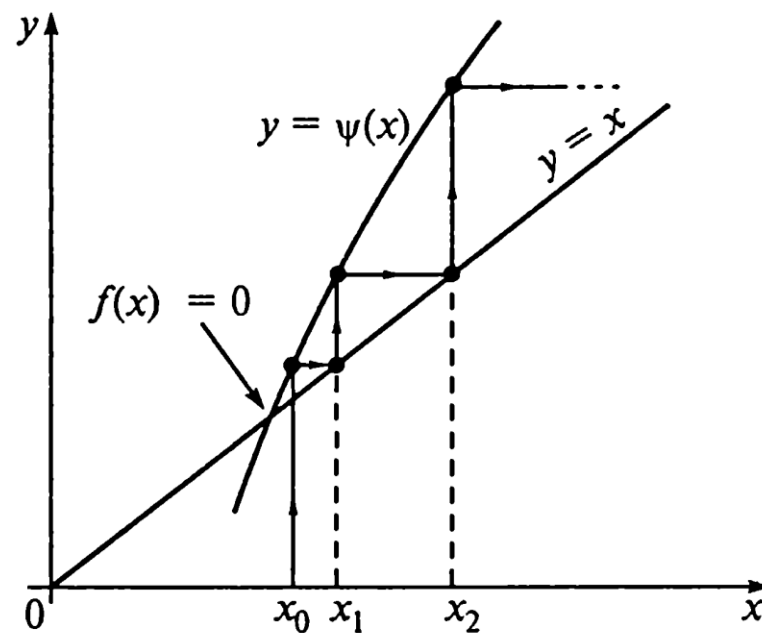


Рис. 2.11. Расходящийся процесс в методе простой итерации

На рис. 2.10 сходимость обеспечивается для медленно изменяющихся функций $\psi(x)$, для которых выполняется условие $|\psi'(x)| < 1$.

На рис. 2.11 расходящийся процесс наблюдается для более быстро меняющейся функции $|\psi'(x)| > 1$.

Можно сделать вывод, что для обеспечения сходимости метода простой итерации необходимо выполнить условие $|\psi'(x)| < 1$.

На практике в качестве рассматриваемой окрестности используют интервал $[a; b]$, а условие сходимости итерационного процесса имеет вид:

$$|\psi'(x)| < 1.$$

Для сходящегося итерационного процесса характерно следующее: при решении задачи переменная последовательно стремится к некоторому искомому пределу. Так как итерационный процесс представляет собой последовательность повторяющихся вычислительных процедур, то он, естественно, описывается циклическими алгоритмами. Особенность итерационного цикла заключается в том, что неизвестен закон изменения рекуррентной величины, выбранной в качестве параметра цикла, и неизвестно число повторений цикла. При этом значение, полученное на n -й итерации, является исходным для следующей $(n + 1)$ -й итерации (рис. 2.12).

Процесс итераций продолжается до тех пор, пока для двух последовательных приближений x_{n+1} и x_n не будет обеспечено выполнение неравенства

$$|x_n - x_{n-1}| \leq \varepsilon,$$

где ε — точность вычислений.

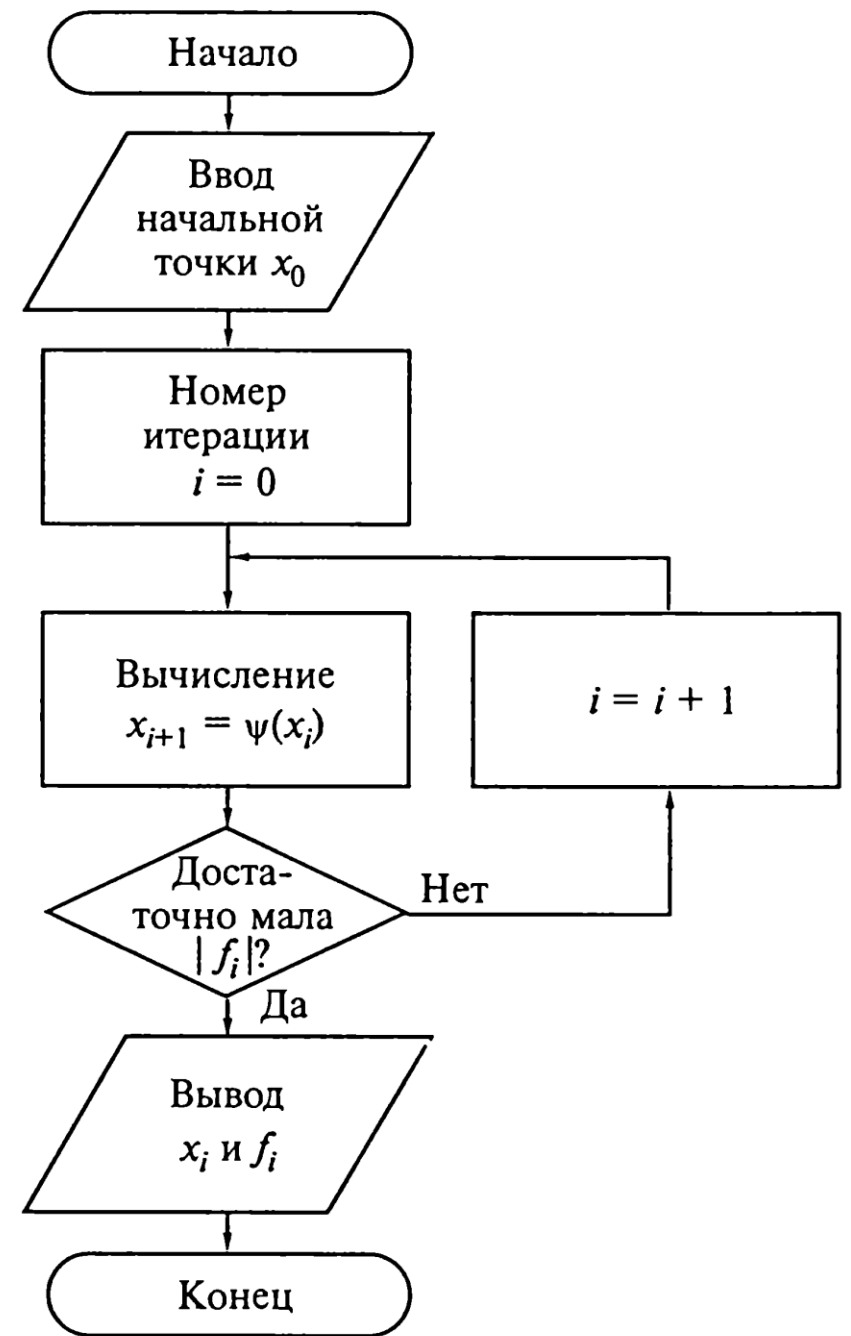
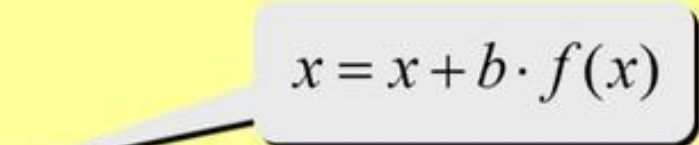



Рис. 2.12. Метод простой итерации

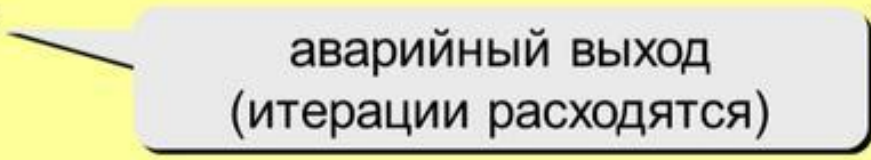
Метод итераций (программа)

```
//-----  
// Iter решение уравнения методом итераций  
// Вход:  x - начальное приближение  
//        b - параметр  
//        eps - точность решения  
// Выход: решение уравнения f(x)=0  
//        n - число шагов  
/////-----  
float Iter ( float x, float b, float eps, int &n)  
{  
    float dx;  
    n = 0;  
    while ( 1 ) {  
        dx = b*f(x);  
        x = x + dx;  
        if ( fabs(dx) < eps ) break;  
        n ++;  
        if ( n > 100 ) break;  
    }  
    return x;  
}
```


$$x = x + b \cdot f(x)$$



нормальный
выход

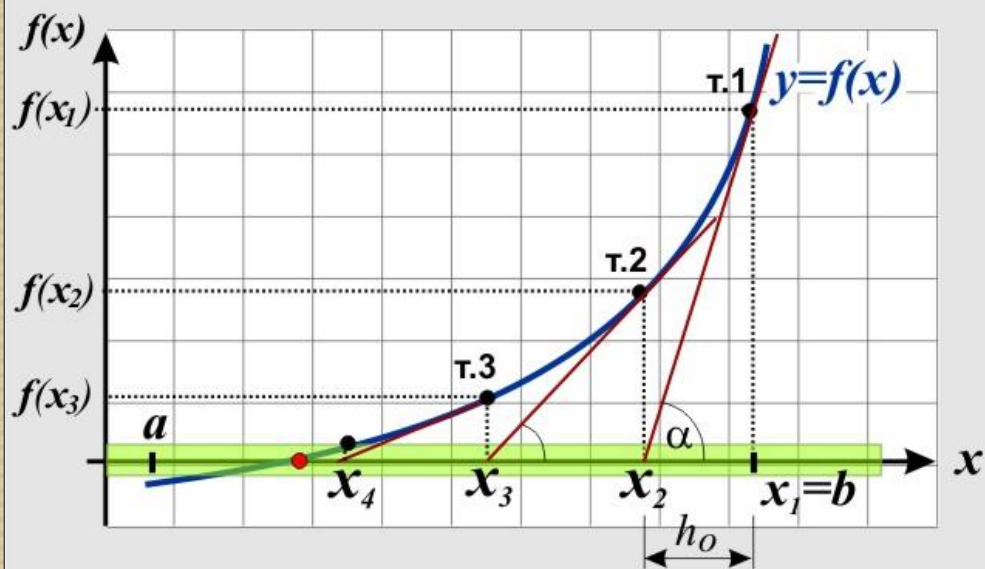
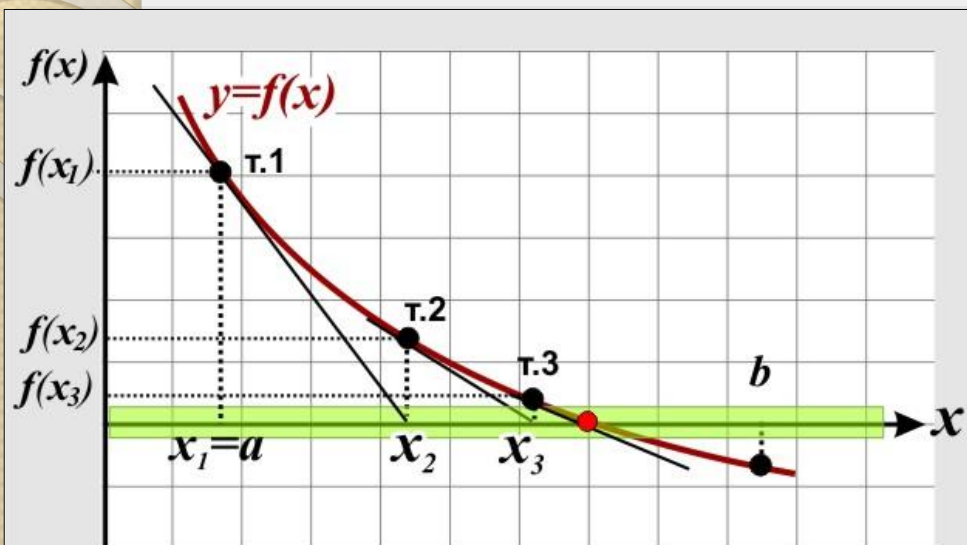


аварийный выход
(итерации расходятся)

Численные методы нахождения корней уравнения

Метод касательных (метод Ньютона), (метод линеаризации)

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



- 1) В точке $f(x_1) = a$ или $f(x_1) = b$ строится касательная
- 2) Следующая точка приближения – это точка пересечения, полученной касательной с осью абсцисс x_2
- 3) Процесс продолжается вплоть до достижения заданной точности ε

Из рис. 3.3 очень легко получить итерационную формулу метода, используя геометрический смысл производной. Если $f(x)$ имеет непрерывную производную, тогда получим:

$$f'(x_1) = \frac{f(x_1)}{x_1 - x_2}; \quad x_2 = x_1 - \frac{f(x_1)}{f'(x_1)};$$

Аналогично можно получить x_2, x_3, \dots Таким образом, можно записать общую формулу:

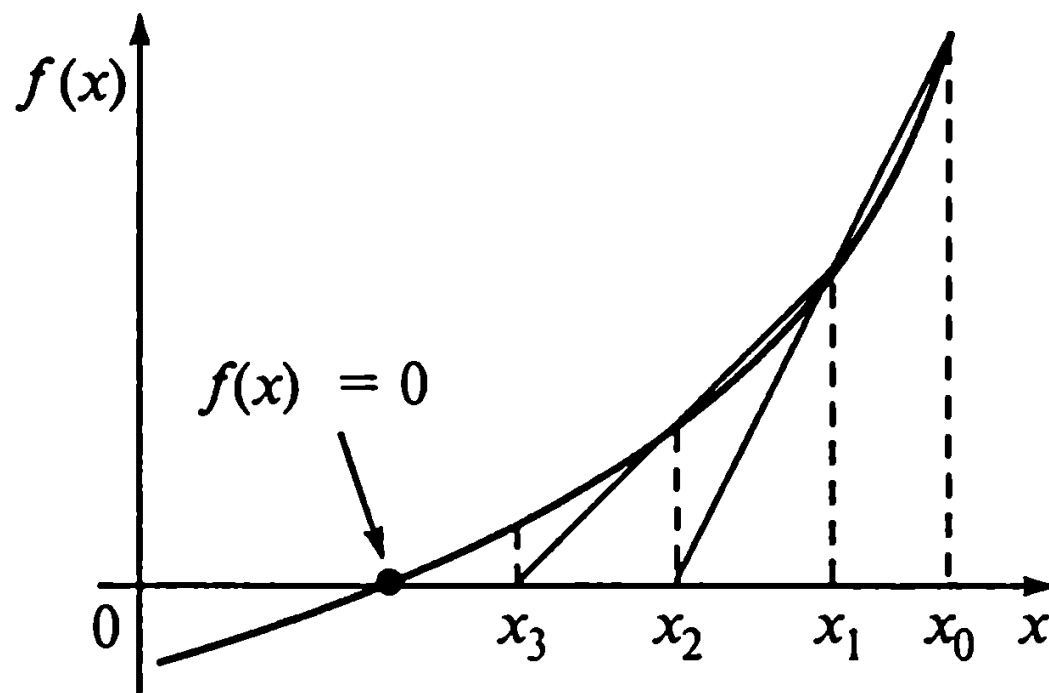
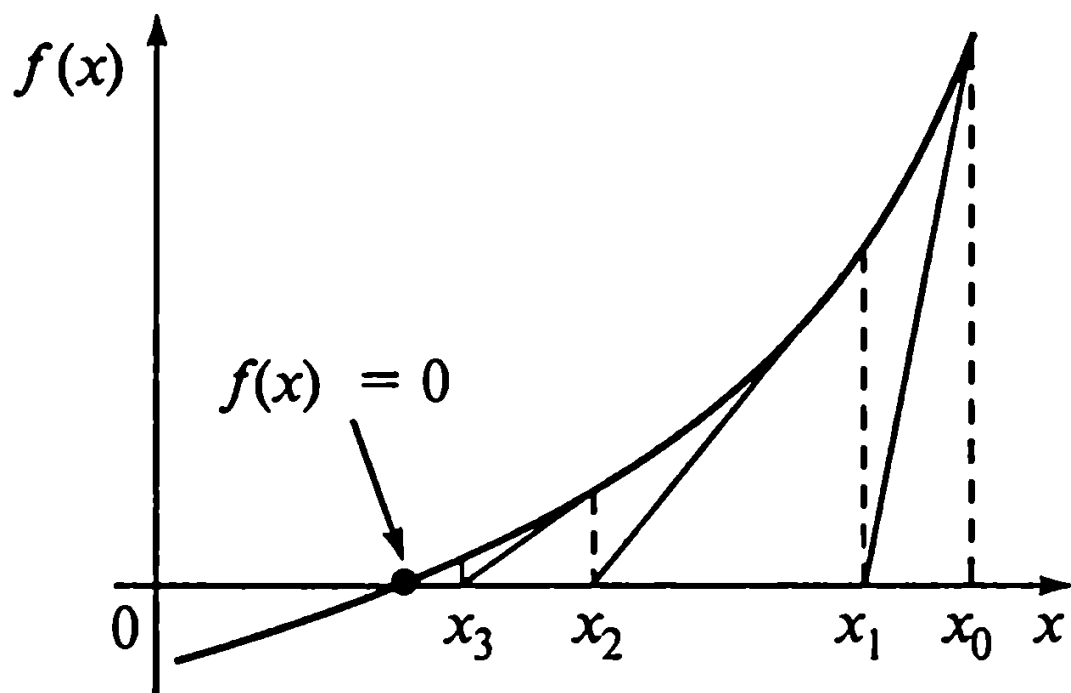
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)};$$

Метод Ньютона (метод касательных)

Если известно начальное приближение x_0 корня уравнения $f(x) = 0$, то последовательные приближения находят по формуле метод Ньютона.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \dots$$

Графическая интерпретация метода касательных имеет вид



Выберем на отрезке $[a; b]$ произвольную точку x_0 — нулевое приближение. Затем найдем

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

далее

$$x_2 = x_1 - \frac{F(x_1)}{F'(x_1)}.$$

Таким образом, процесс нахождения корня уравнения сводится к вычислению чисел x_n по формуле

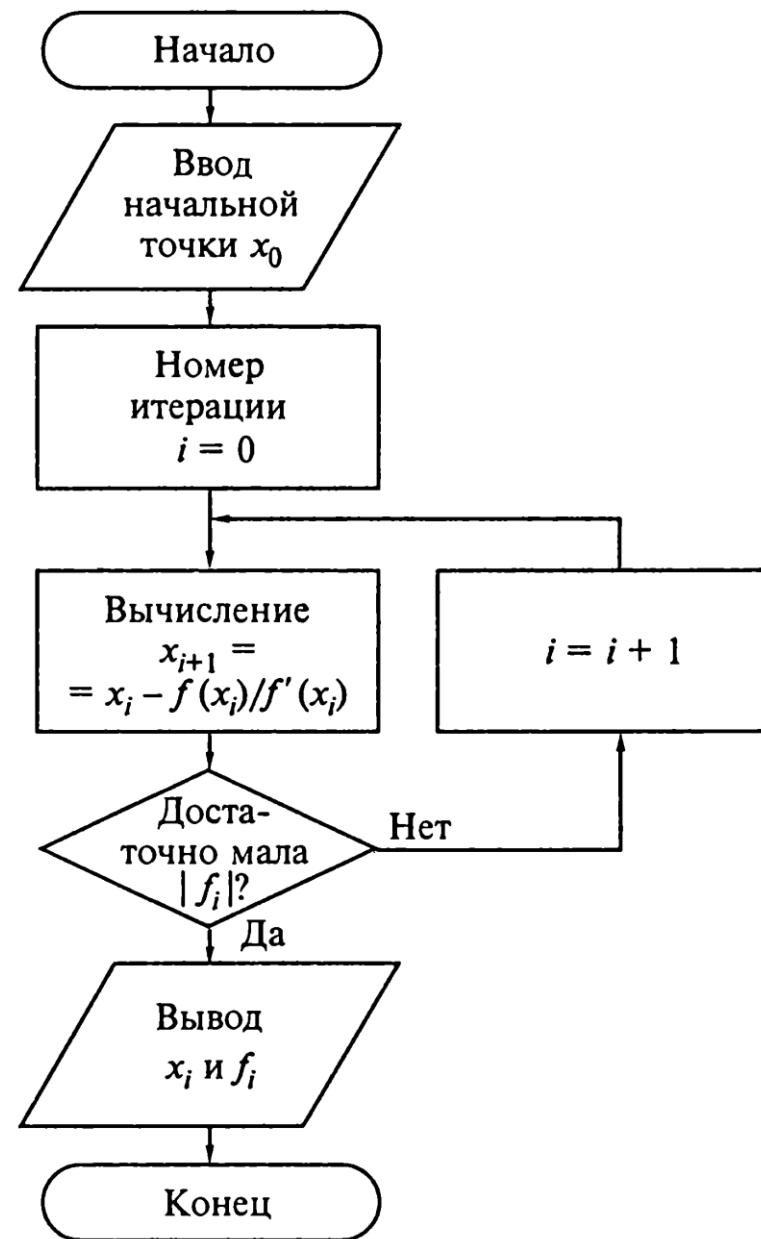
$$x_n = x_{n-1} - \frac{F(x_{n-1})}{F'(x_{n-1})}, \quad n = 1, 2, 3, \dots$$

Процесс вычисления продолжается до тех пор, пока не будет выполнено условие

$$|x_n - x_{n-1}| < \varepsilon.$$

Схема итерационного процесса метода Ньютона приведена на рис. 2.15, из которого понятно, что каждое следующее приближение может быть определено по формуле

$$x^{(n+1)} - x^{(n)} = \frac{f(x^{(n)})}{\operatorname{tg} \alpha} = \frac{f(x^{(n)})}{f'(x^{(n)})}.$$



Метод Ньютона (программа)

```
{-----  
Newton решение уравнения методом Ньютона  
Вход:  x - начальное приближение  
       eps - точность решения  
Выход: решение уравнения  $f(x)=0$ , n - число шагов  
-----}  
function Newton (x, eps: real; var N: integer): real;  
var dx: real;  
    OK: boolean;  
begin  
    N := 0; OK := False;  
    while not OK and (N <  
    begin  
        dx := f(x) / df(x);  
        x := x - dx;  
        N := N + 1;  
        OK := abs(dx) < eps;  
    end;  
    Newton := x;  
end;
```

```
{ функция }  
function f(x:real): real;  
begin  
    f := 3*x*x*x+2*x+5;  
end;  
{ производная }  
function df(x:real): real;  
begin  
    df := 9*x*x + 2;  
end;
```

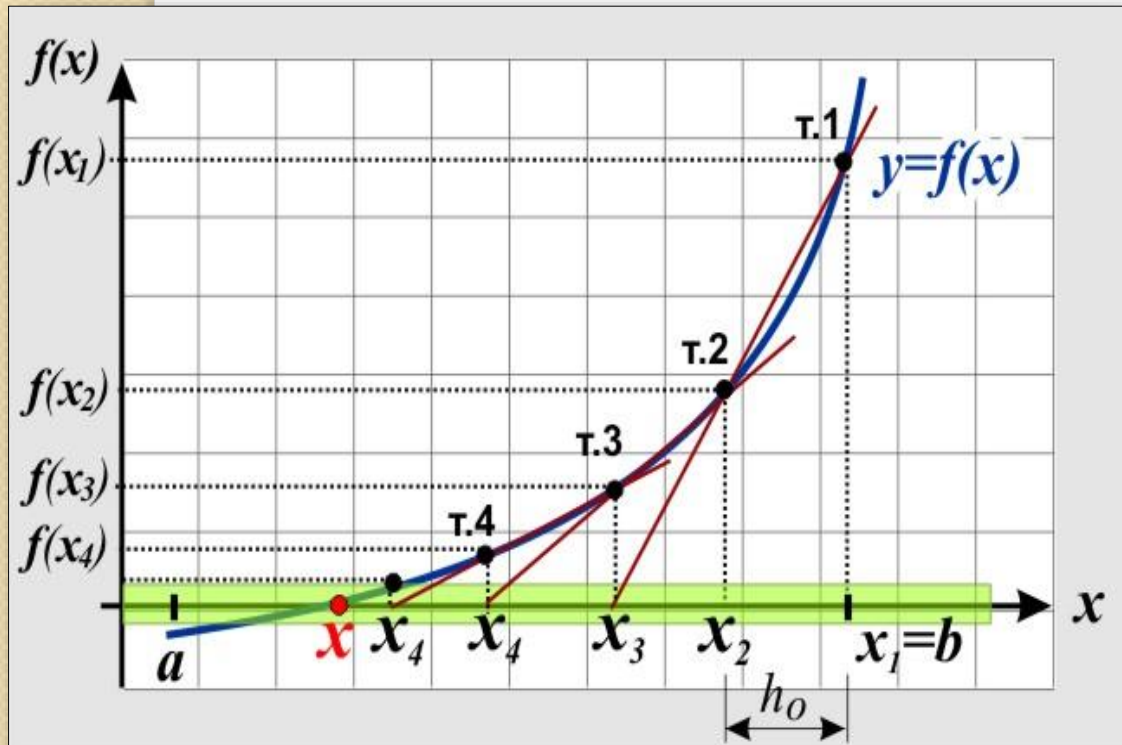

Численные методы нахождения корней уравнения

Метод секущих

В данном методе, в отличие от метода Ньютона, проводятся не касательные, а секущие. Из рисунка легко получить итерационную формулу:

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

- 1) задаться первоначальными значениями x_1 и x_2
- 2) вычислить x_3 по формуле, представленной выше
- 3) вычислить значение функции $f(x_3)$ в этой точке, сравнить с заданной погрешностью ε
- 5) если $f(x_3) > \varepsilon$ заменить x_2 на x_3 , если $f(x_3) \leq \varepsilon$ прекратить расчет
- 6) повторить действия с п.2 по п.3, пока заданная точность не будет достигнута.



В качестве начального приближения необходимо задать не только x_1 , но и x_2 . Метод секущих имеет одно преимущество перед методом Ньютона – здесь не нужно вычислять производную.

Но этот метод имеет также существенные недостатки. Сходимость итераций может быть немонотонной не только вдали, но и в малой окрестности корня. В знаменателе формулы стоит разность значений функции. Вдали от корня это не существенно, но **вблизи корня значения функции малы и очень близки. Возникает проблема деления на очень малые числа или даже ноль**, что приводит к потере значащих цифр, т.е. точности, или, как говорят к «разболтке» счёта.

Метод секущих (метод хорд)

Если x_0, x_1 — приближенные значения корня уравнения $f(x) = 0$, а $f(x_0) f(x_1) < 0$, то последующие приближения находят по формуле

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(x_{n-1})} (x_n - x_{n-1}), \quad n = 1, 2, \dots$$

Методом хорд называют также метод, при котором один из концов отрезка $[a; b]$ закреплен (рис. 2.13), т. е. вычисление приближения корня уравнения $f(x) = 0$ производят по формулам

$$x_0 = b, \quad x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(a)} (x_n - a)$$

либо

$$x_0 = a, \quad x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(b)} (x_n - b).$$

При расчете предполагается, что корень уравнения находится на отрезке $[a; b]$, а $f''(x)$ сохраняет знак на $[a; b]$.

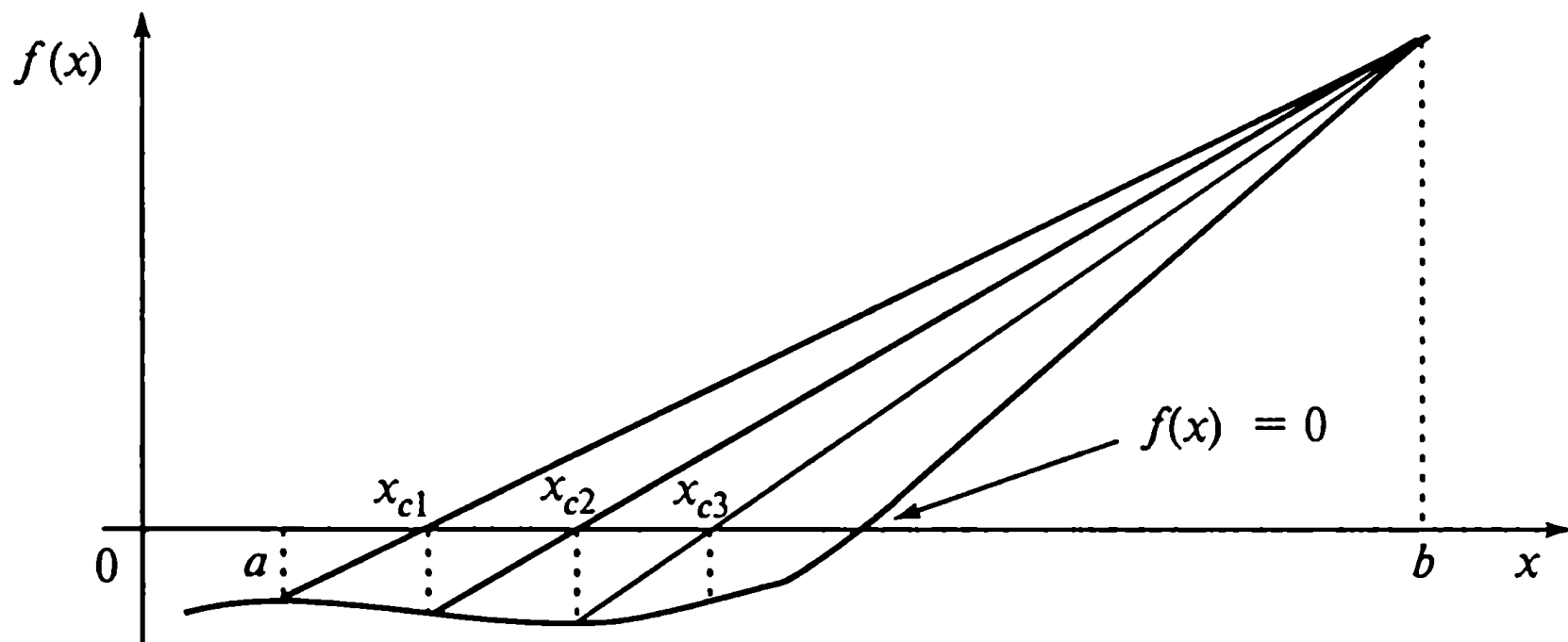


Рис. 2.13. Метод хорд

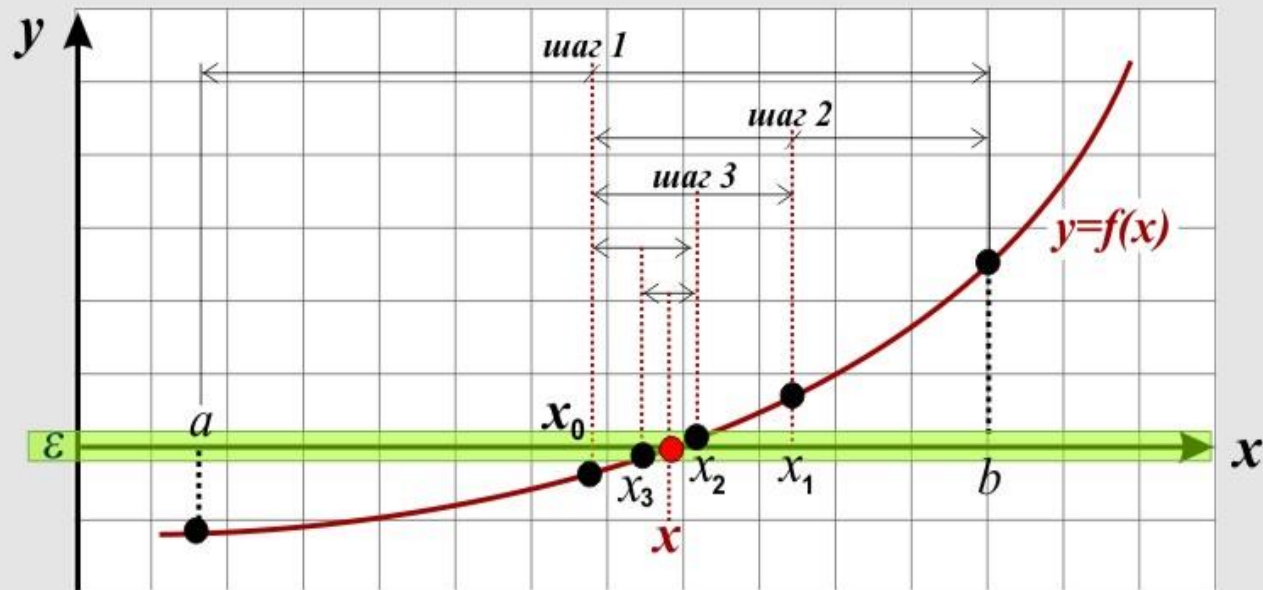
Из рис. 2.13 видно, что получаемые точки x_c постепенно сходятся к корню уравнения. Поскольку в рассмотренном методе очередное приближение x_c определяется с помощью интерполяции, учитывающей наклон кривой $f(x)$, он во многих случаях оказывается более эффективным, чем метод половинного деления.

Численные методы нахождения корней уравнения

Метод дихотомии (деление пополам)

- 1) Пусть мы нашли такие точки a и b что $f(a) \cdot f(b) < 0$, т.е. на отрезке $[a, b]$ лежит не менее одного корня
- 2) Найдем середину отрезка $x_0 = (a+b)/2$ и вычислим значение функции на середине этого отрезка $f(x_0)$
- 3) Из двух получившихся половинок отрезка выберем ту, для которой $f(x_0) \cdot f(a)$ или $f(b) \leq 0$, т.е. отрезок на котором функция меняет знак.
- 4) новый отрезок опять делим пополам и выберем ту половину, на концах которой функция имеет разные знаки, и т. д.

Если требуется найти корень с точностью ε , то продолжаем деление пополам до тех пор, пока значение функции в средней точке не будет меньше ε . **Тогда середина последнего отрезка даст значение корня с требуемой точностью.**



Метод прост и надежен сходится для любых непрерывных функций, в том числе не дифференцируемых.

Для начала расчета надо найти отрезок, на котором функция меняет знак. Если в этом отрезке несколько корней, то заранее неизвестно, к какому из них сойдется процесс (хотя к одному из них сойдется).

Применяется тогда, когда требуется высокая надежность счета, а скорость сходимости малосущественна.

Метод половинного деления

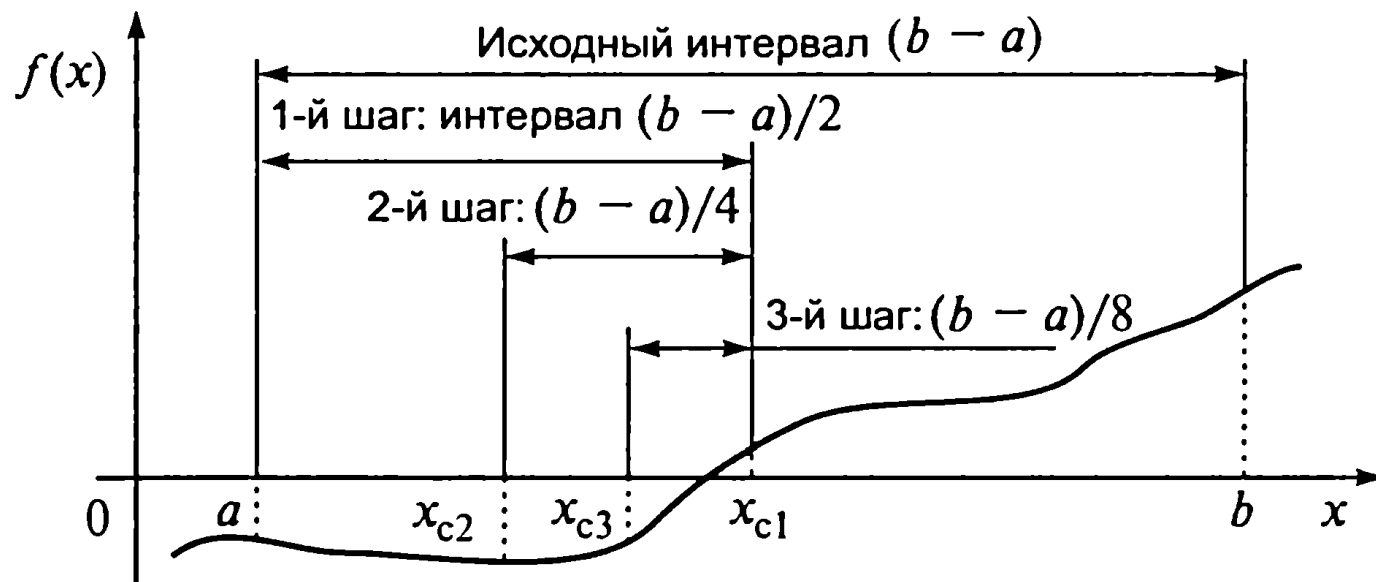


Рис. 2.7. Метод половинного деления (дихотомии)

В методе половинного деления (дихотомии, бисекции) заданный отрезок $[a; b]$ разделим пополам (рис. 2.7) и положим $x_0 = (a + b)/2$. Из двух полученных отрезков $[a; x_0]$ и $[x_0; b]$ выбираем тот, на концах которого функция $f(x)$ имеет противоположные знаки. Полученный отрезок снова делим пополам и приводим те же рассуждения. Процесс продолжаем до тех пор, пока

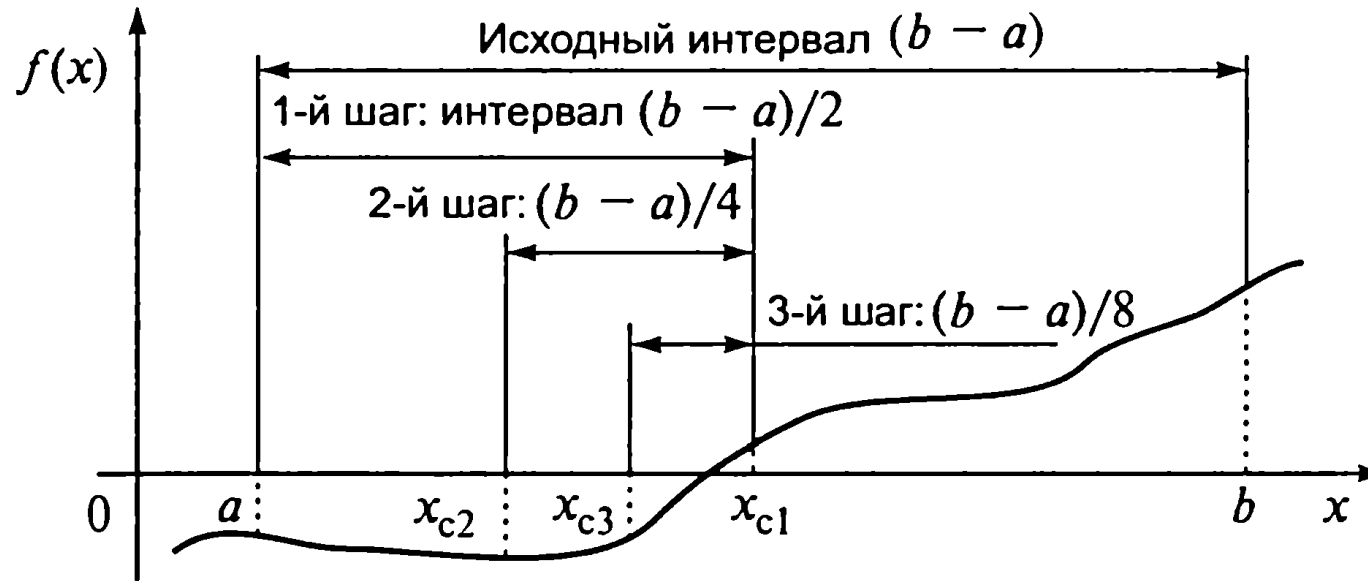
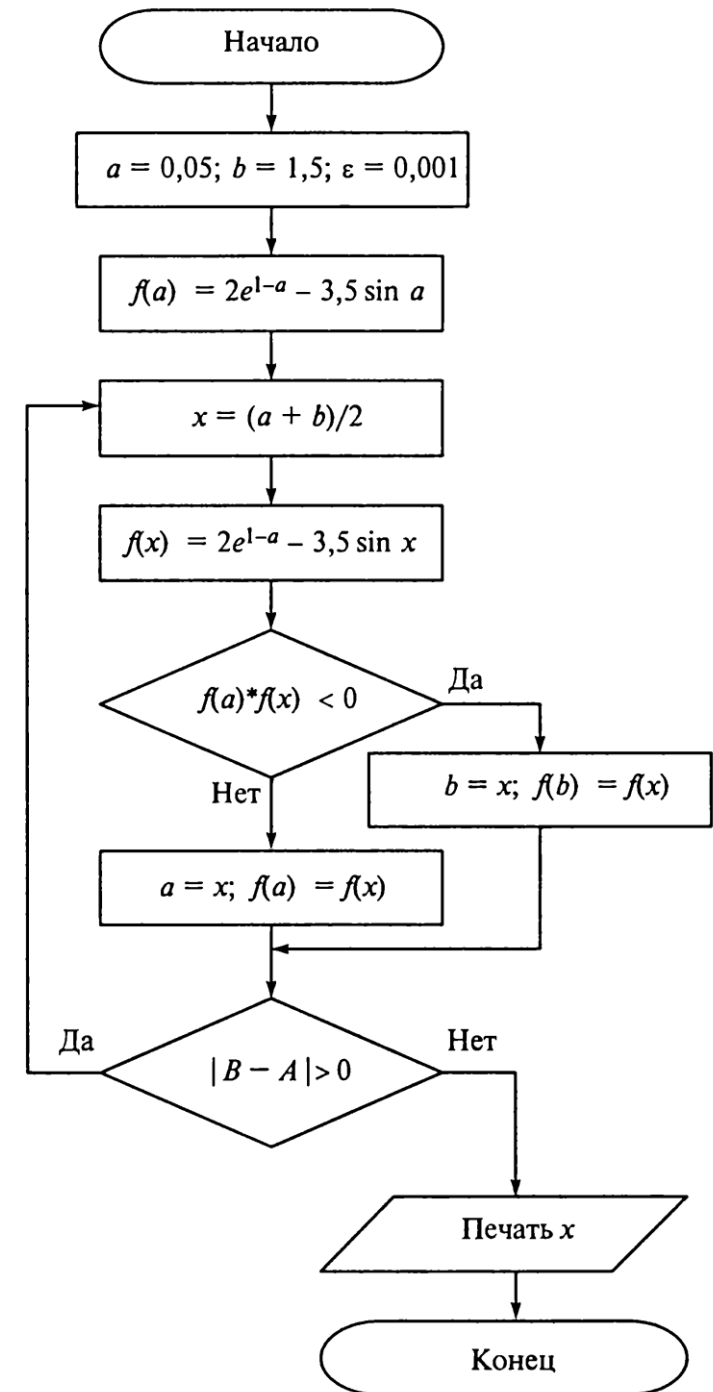


Рис. 2.7. Метод половинного деления (дихотомии)

длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданного ε , любую точку отрезка с точностью ε можно принять за корень уравнения $f(x) = 0$.

Таким образом, если x_0 и x_1 таковы, что $f(x_0) f(x_1) < 0$, то полагаем $x_2 = (x_0 + x_1)/2$ и вычисляем $f(x_2)$. Если $f(x_2) = 0$, то корень найден. В противном случае из отрезков $[x_0; x_2]$ и $[x_2; x_1]$ выбираем тот, на концах которого f принимает значения разных знаков, и проделываем аналогичную операцию. Процесс продолжаем до получения требуемой точности.

Пример 2.21. Методом половинного деления найти корень уравнения $2e^{1-x} - 3,5\sin x = 0$ на отрезке $[0,05; 1,5]$ с точностью $\varepsilon = 0,001$.



Метод деления отрезка пополам

```
//-----  
// BinSolve находит решение на [a,b]  
//          методом деления отрезка пополам  
// Вход:  a, b - границы интервала,  a < b  
//          eps - точность решения  
// Выход: x - решение уравнения f(x)=0  
//-----  
float BinSolve ( float a, float b, float eps )  
{  
    float c;  
    while ( b - a > eps )  
    {  
        c = (a + b) / 2;  
        if ( f(a)*f(c) < 0 )  
            b = c;  
        else a = c;  
    }  
    return (a + b) / 2;  
}  
  
float f ( float x )  
{  
    return x*x - 5;  
}
```