Решение нелинейных уравнений

Разделяют методы решения отдельных уравнений и систем уравнений.

В общем случае уравнение с одним неизвестным имеет вид:

$$f(u)=0 \qquad (4.0)$$

или

$$f_1(u)= f_2(u) \qquad (4.1)$$

где $f(u)$, $f_1(u)$, $f_2(u)$ - заданные функции действительного или комплексного аргумента, определённые на данном отрезке [*a,b*]. *Алгебраическими* уравнениями называются уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные). В частности, многочлен является целой алгебраической функцией. Уравнения, содержащие другие функции (тригонометрические, показательные, логарифмические и др.), называются *трансцендентными*.

Методы решения нелинейных уравнений делятся на **прямые** и **итерационные**.

Прямые методы позволяют записать корни в виде некоторого конечного соотношения (формулы). Известны такие методы для решения тригонометрических, логарифмических, показательных, а также простейших алгебраических уравнений. Для решения же большинства встречающихся на практике уравнений используются методы последовательных приближений, т.е. итерационные методы.

Они предполагают, что известны достаточно малые окрестности, в каждой из которых имеется только один корень уравнения. Принимая за начальное приближение одну из точек этой окрестности, можно вычислить искомый корень с заданной точностью.

Таким образом, задача приближенного вычисления корней уравнения распадается на две:

- **отделение корней уравнения**, то есть отыскание достаточно тесных промежутков, в каждом из которых содержится только один корень уравнения;
- **уточнение корней,** то есть вычисление корня с заданной точностью, если известно некоторое начальное его приближение в области, не содержащей других корней.

Локализация (отделение) корней

Не существует достаточно эффективных методов решения задачи отделения корней в общем случае, особенно при переходе от одного уравнения к системе.

Графический метод решения нелинейных уравнений – один из самых простых. Применять его можно лишь для грубой оценки корней или использовать для отделения корней.

Он заключается в построении графика функции ***f(u)*** либо графиков двух функций $f_1(u)$ и $f_2(u)$ на отрезке [*a,b*].

В первом случае точка пересечения графика функции с осью абсцисс, а во втором – абсцисса точки пересечения двух функций дают приближённое значение корня уравнения (4.0) или соответственно (4.1).

Найденные таким образом приближенные значения корней позволяют выделить отрезки, в которых при необходимости можно выполнить уточнение корней.

Рис. 1. Графический способ отделения корней функции ***f(u)***

Иногда бывает удобнее представить уравнение (4.0) в виде (4.1), тогда приближенные значения корней можно найти, определив абсциссы точек пересечения графиков функций $f_1(u)$, $f_2(u)$.

Пример: $(\sqrt[3]{(x-1)^2} - \sqrt[3]{x^2} = 0) \Rightarrow \sqrt[3]{(x-1)^2} = \sqrt[3]{x^2}$

Несложно заметить, что корень уравнения находится в интервале от нуля до единицы.

При отделении действительных корней расчётным путём для непрерывных функций ***f(u)*** можно руководствоваться следующим:

- если на концах отрезка *[a,b]* функция имеет разные знаки *f(a)f(b)<0,* то между точками *a* и *b* на оси абсцисс имеется хотя бы один корень или нечётное количество корней, при этом отрезок *[a,b]* называют *интервалом изоляции корня*, а значения *a* и *b* называют *пределами интервала изоляции*;
- если же *f(a)f(b)>0* , то между точками *a* и *b* имеется чётное число корней или их нет совсем;
- если *f(a)f(b)<0* и при этом функция ***f(u)*** имеет первую или вторую производную, не меняющую знака на этом отрезке, то корень единственный.

При отделении корней всегда нужно руководствоваться физическими соображениями. Это позволяет значительно сузить диапазон поиска корней, а часто и локализовать их физически допустимые значения. Для априорной оценки исходного приближения можно также использовать решения аналогичной задачи при других исходных данных.

Для алгебраических и трансцендентных нелинейных уравнений пригодны одни и те же методы уточнения приближённых значений действительных корней.

Условие окончания итерационного процесса (нахождения значения корня с точностью ε имеет вид:

$$|u_{(k+1)} - u_{(k)}| \le \varepsilon, \quad k = 0, 1, 2, 3, \ldots$$

или $|f(u_{(k+1)})| \le \varepsilon$ при решении уравнения в виде ***f(u)=0.***

Итерационные методы являются **самоисправляющимися**, так как отдельные ошибки расчётов не приводят к искажению конечных результатов. Это обусловлено тем, что каждое очередное приближение корня при последовательных итерациях можно рассматривать как новое начальное приближение.

Метод перебора – простейший из методов.

При решении нелинейного уравнения методом перебора задаются начальное значение аргумента $x=a$ и шаг h, который при этом определяет и точность нахождения корней нелинейного уравнения.

Пока выполняется условие $f(x)*f(x+h)>0$ аргумент x увеличиваем на шаг h $(x=x+h)$.

Если произведение $f(x)*f(x+h)$ становится отрицательным, то на интервале $[x,x+h]$ существует решение уравнения.

Этот метод можно использовать для отделения корней уравнения, а их уточнение осуществлять другим методом.

Метод половинного деления (дихотомии)

Предположим, что корень существует на отрезке $[a;b]$ и знаки $f(a)$ и $f(b)$ различны (функция меняет знак при переходе через корень x^*).

Положим $a_0 = a$ $b_0 = b$ и вычислим значения функции в левом конце отрезка $f(a_0)$ и $f(b_0)$, и в его середине $c_0 = \frac{a_0 + b_0}{2}$, $f(c_0)$. Сравним знаки чисел $f(a_0)$ и $f(c_0)$. Если эти знаки различны, то корень x^* лежит в интервале $(a_0; c_0)$; если же одинаковы, то тогда различны знаки $f(b_0)$ и $f(c_0)$, и корень лежит в интервале $(c_0; b_0)$.

Возможен ещё случай $f(c_0) = 0$; тогда корень $x^* = c_0$ уже найден.

В обоих случаях смены знака корень оказывается отделён на отрезке $[a_0; c_0]$ или $[c_0; b_0]$, длина которого ровно в два раза меньше длины исходного отрезка $[a_0; b_0] = [a; b]$. Обозначим этот отрезок половинной длины через $[a_1; b_1]$ (то есть положим $a_1 = a_0; b_1 = c_0$ в случае, когда $f(a_0)$ и $f(c_0)$ разных знаков, и $a_1 = c_0; b_1 = b_0$ в случае, когда $f(a_0)$ и $f(c_0)$ одного знака).

Далее повторим процесс для отрезка $[a_1; b_1]$: снова отыщем его середину c_1, найдём значение функции $f(c_1)$ и сравним знак этого числа со знаком $f(a_1)$; если знаки разные, то корень отделён на $[a_2; b_2] = [a_1; c_1]$, если одинаковые, то на $[a_2; b_2] = [c_1; b_1]$ (или же оказывается, что $f(c_1)=0$; тогда корень найден). Длина отрезка, на котором отделён корень, уменьшилась ещё в два раза.

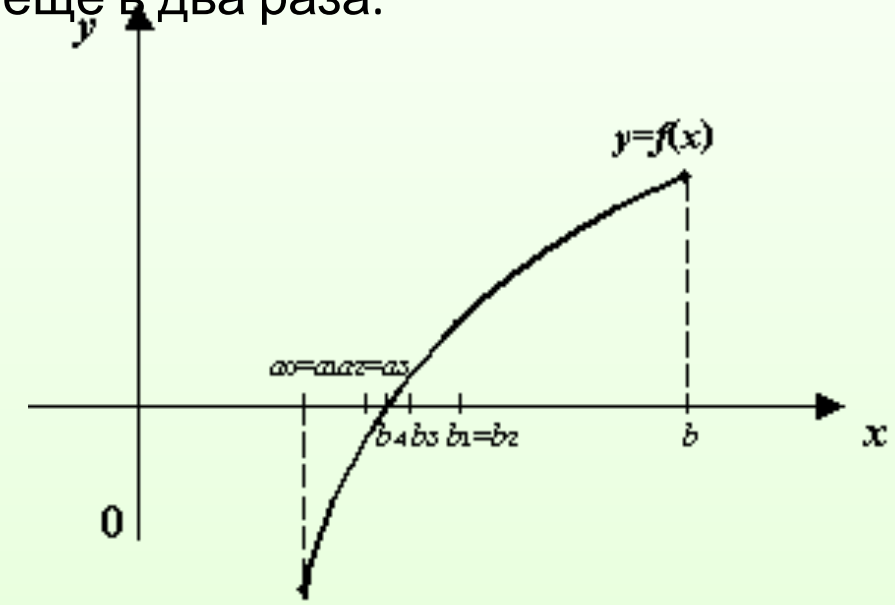


Рис.2.Последовательное деление отрезка пополам и приближение к корню x^*

Поступая тем же образом и далее, получаем, что после k делений длина отрезка, на котором лежит корень, сокращается в 2^k раз и становится $\delta_k = \frac{b-a}{2^k}$ равной c_i (если корень не был точно определён на каком-то предыдущем этапе, то есть не совпал с при некотором i).

Пусть ε -- заданная точность, с которой требуется отыскать корень. Процесс деления отрезков следует остановить, как только станет верным неравенство $\delta_k \le 2\varepsilon$. Очевидно, что если при этом положить

$$\widetilde{x} = c_k = \frac{a_k + b_k}{2},$$

то расстояние от корня x^* , лежащего где-то в интервале (a_k, b_k) , до середины этого интервала $\widetilde{x}$ будет не больше ε , то есть приближённое равенство $x^* \approx \widetilde{x}$ будет выполнено с нужной точностью.

Подставив значение δ_k в неравенство $\delta_k \le 2\varepsilon$, прологарифмировав его и выразив из него k, получим:

$$k \le \lg((b-a)/\varepsilon)-1,$$

т.е. число итераций в этом методе зависит от предварительно задаваемой точности ε и длины отрезка $[a,b]$ и не зависит от вида функции $f(x)$.

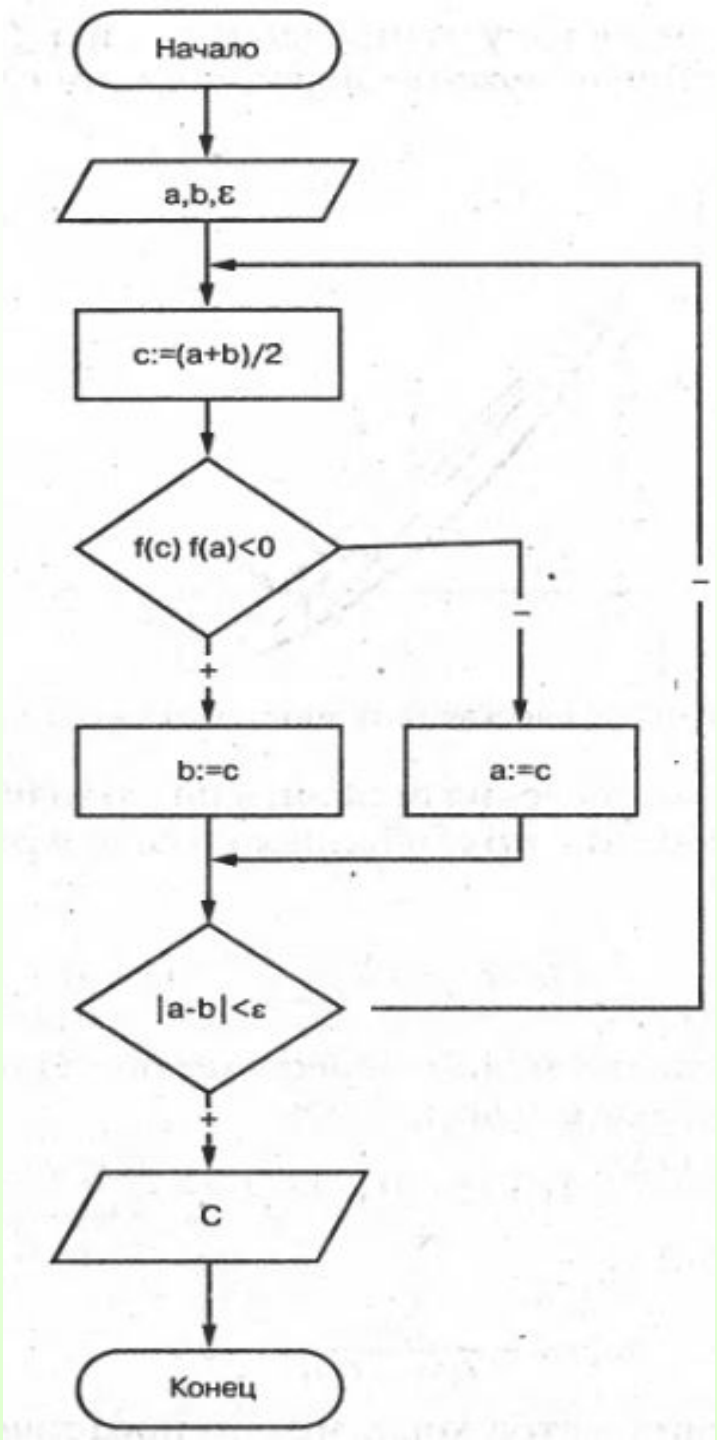
Этот метод прост и надёжен, сходится к любому простому корню любой непрерывной и необязательно дифференцируемой функции.

Недостаток метода: поиск исходного отрезка, на котором функция меняет знак. Если корней на исходном отрезке несколько, то неизвестно к какому из них процесс сойдётся. Метод не работает для поиска корней чётной кратности. Для поиска корней нечётной кратности метод сходится. Однако, сходимость очень медленная (для уточнения трёх цифр требуется 10 итераций), но точность ответа гарантированна.

Метод плохо устойчив к ошибкам округления и не обобщается для случая системы уравнений.

Его можно использовать как один из методов отделения корней.

Блок-схема алгоритма уточнения корней методом дихотомии



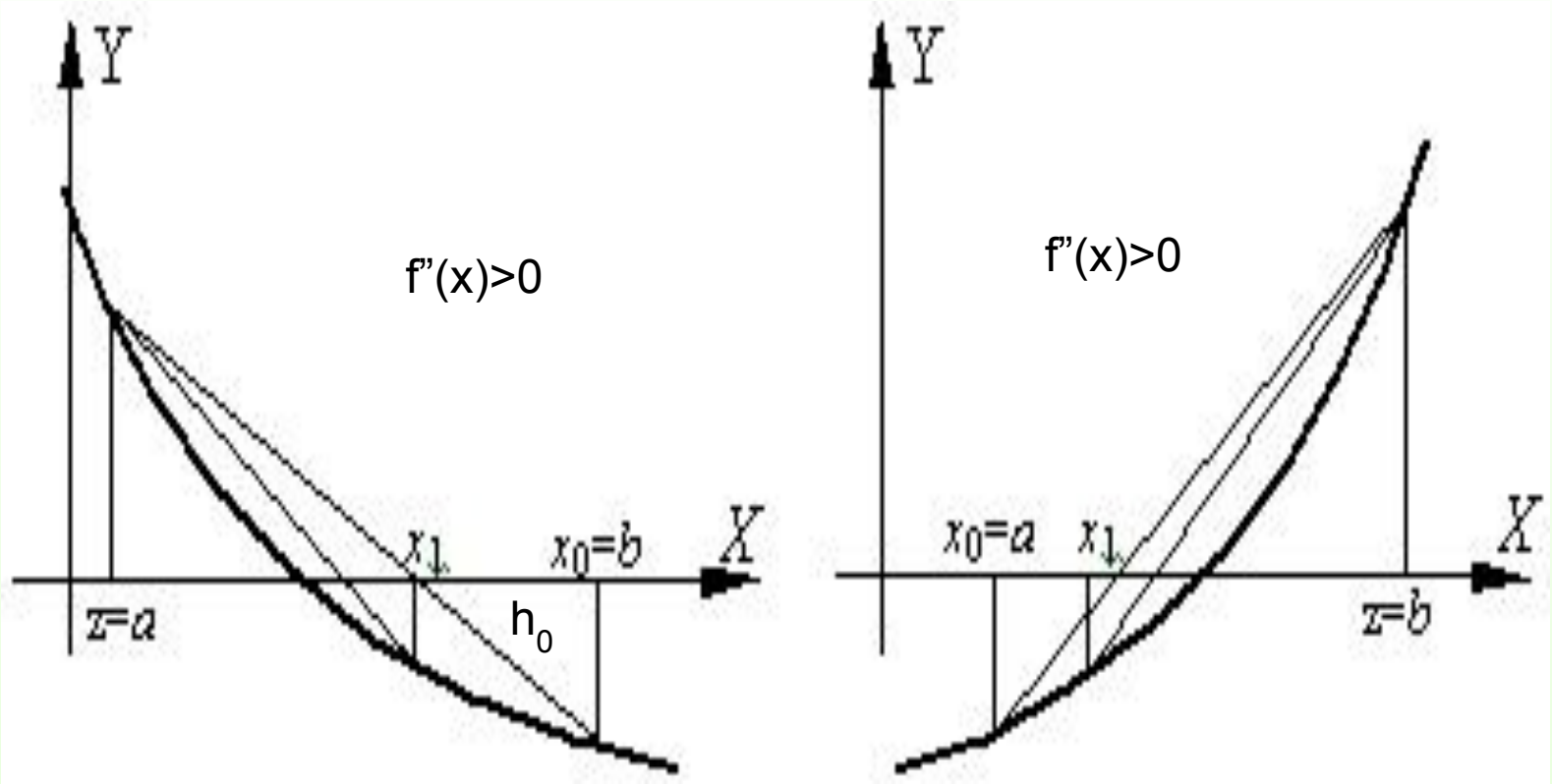
Метод хорд

Метод основан на замене функции *f(x)* на каждом шаге поиска хордой, стягивающей значения функции на концах отрезка *[a,b].* Её пересечение с осью *X* дает приближение корня.

При этом в процессе поиска семейство хорд может строиться:

а) при фиксированном левом конце хорд, т.е. *z=a*, тогда начальная точка $x_0=b$;

б) при фиксированном правом конце хорд, т.е. *z=b*, тогда начальная точка $x_0=a$.



а) б)

Рис. 3. Геометрическая иллюстрация метода хорд

В результате итерационный процесс схождения к корню реализуется рекуррентной формулой, получаемой из подобия треугольников:

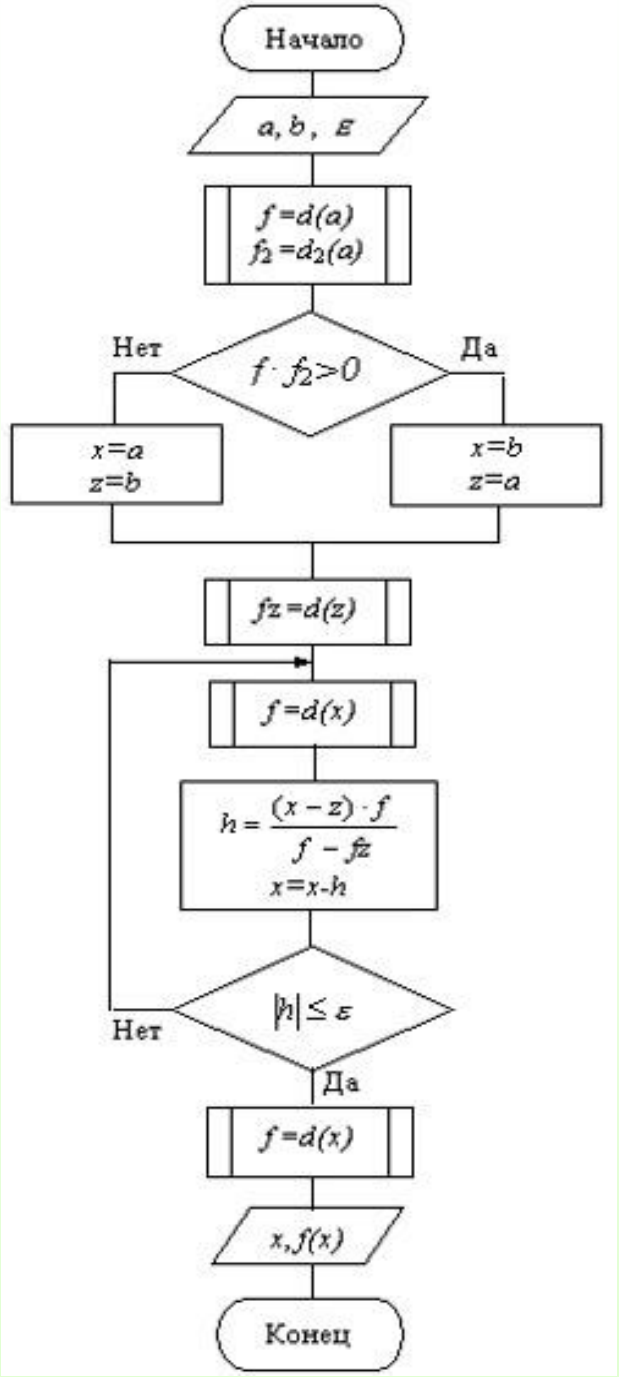
для случая а) $$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(a)}(x_n - a);$$ (4.2)

для случая б) $$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(b)}(x_n - b);$$ (4.3)

Процесс поиска продолжается до тех пор, пока не выполнится условие $|x_{n+1} - x_n| \le$ или $|h| \le \varepsilon.$ (4.4)

Метод обеспечивает быструю **сходимость**, если ***f(z)f"(z) > 0***, т.е. хорды фиксируются в том конце интервала *[a,b],* где знаки функции *f(z)* и её кривизны *f"(z)* совпадают. Метод хорд в большинстве случаев работает быстрее, чем метод дихотомии. Недостатки метода те же, что и в предыдущем случае.

Блок-схема алгоритма уточнения корня методом хорд



Метод Ньютона (метод касательных)

Рассмотренные ранее методы решения нелинейных уравнений являются методами *прямого* поиска. В них для нахождения корня используется нахождение значения *функции* в различных точках интервала *[a,b].*

Метод Ньютона относится к *градиентным* методам, в которых для нахождения корня используется значение *производной*.

Он основан на замене исходной функции *f(x),* на каждом шаге поиска касательной, проведенной к этой функции. Пересечение касательной с осью *X* дает приближение корня.

Выберем начальную точку $x_0=b$ (конец интервала изоляции). Находим значение функции в этой точке и проводим к ней касательную, пересечение которой с осью *X* дает нам первое приближение корня x_1.

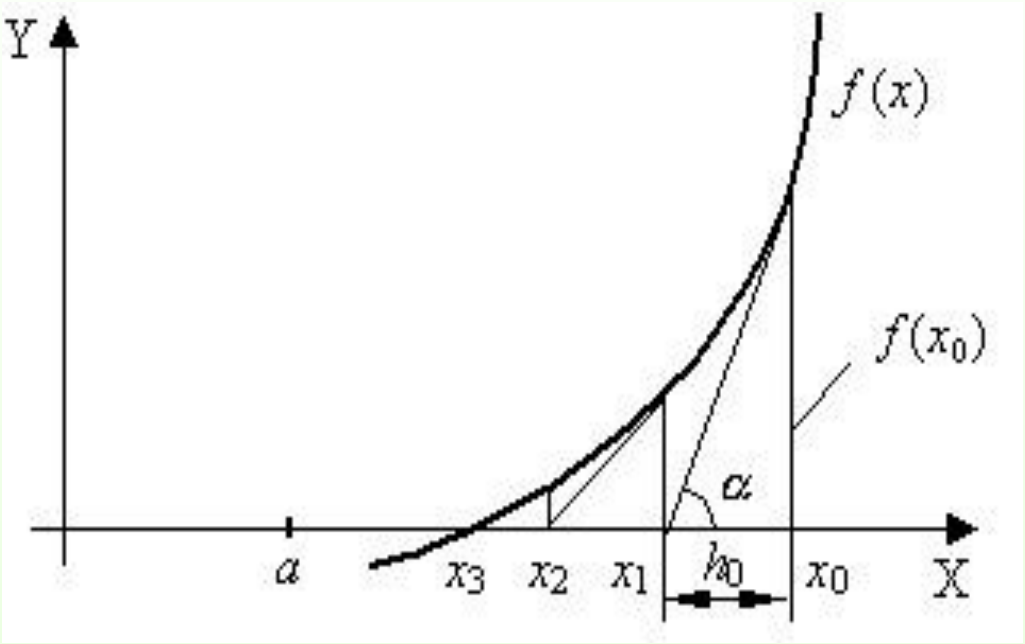


Рис. 4. а) Геометрическая иллюстрация сходимости метода Ньютона(метода касательных)

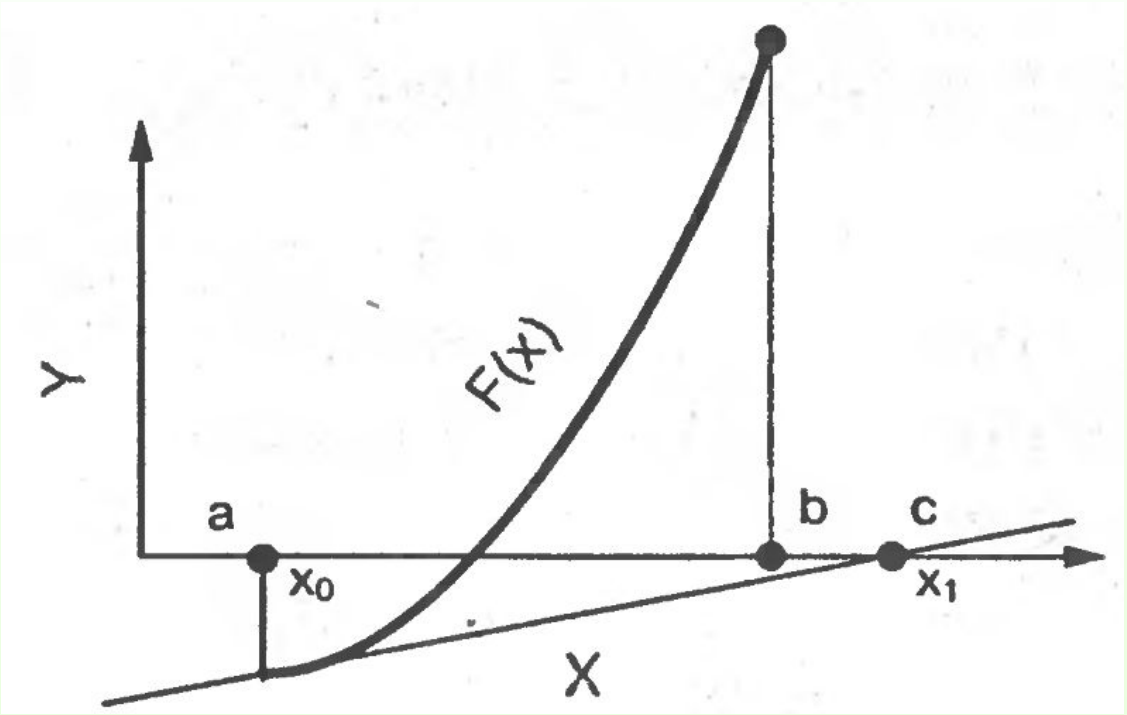


Рис. 4. б) Случай неверного выбора начального приближения в методе Ньютона (методе касательных): $f(x_0) \cdot f''(x_0)<0$

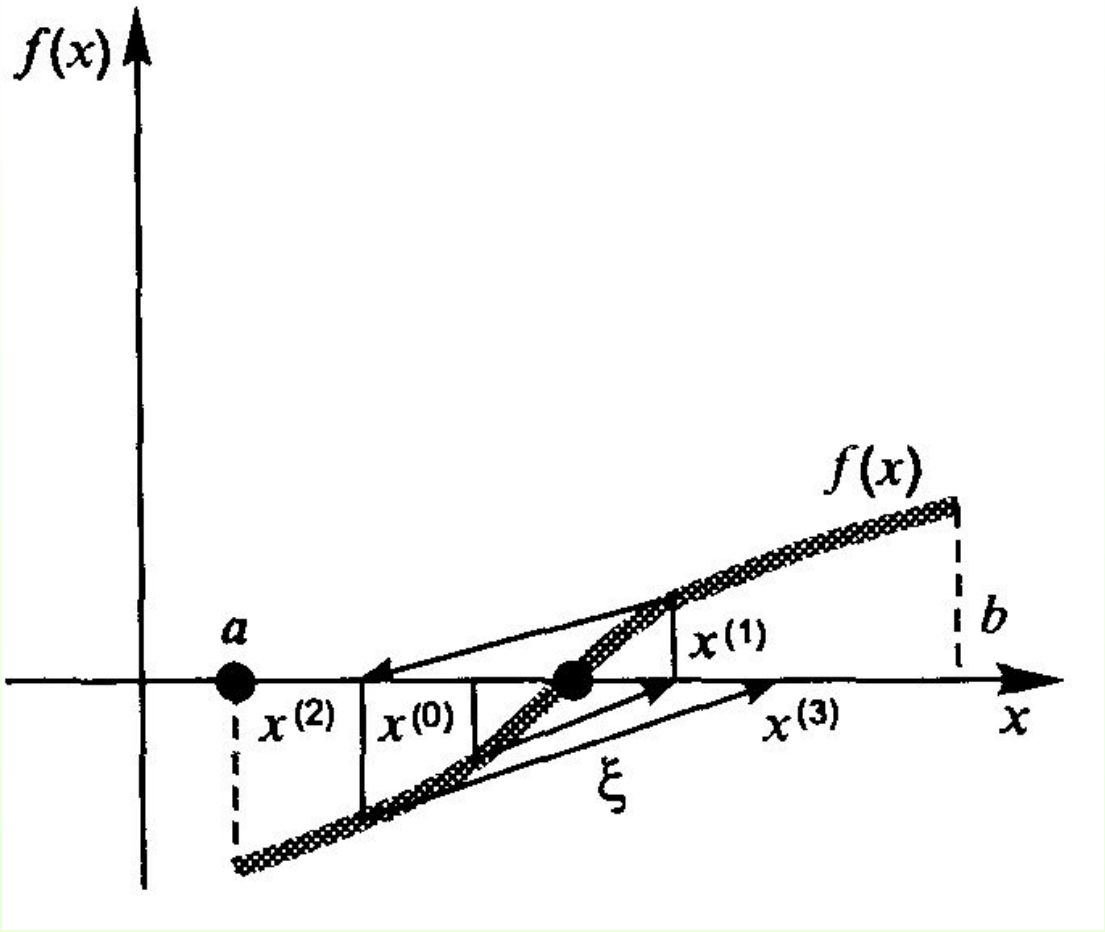


Рис. 4. в) Геометрическая иллюстрация расходимости метода Ньютона(метода касательных)

$x_1 = x_0 - h_0$, $f'(x_0) = \mathrm{tg}(\alpha)$, тогда $h_0 = \dfrac{f(x_0)}{\mathrm{tg}(\alpha)} = \dfrac{f(x_0)}{f'(x_0)}$.

Поэтому $x_1 = x_0 - \dfrac{f(x_0)}{f'(x_0)}$.

В результате итерационный процесс схождения к корню реализуется рекуррентной формулой

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (4.5)$$

Процесс поиска продолжаем до тех пор, пока не выполнится условие:

$|x_{n+1} - x_n| \le \varepsilon$ (4.6)

Подставим в условие (4.6) уравнение (4.5). Получим:

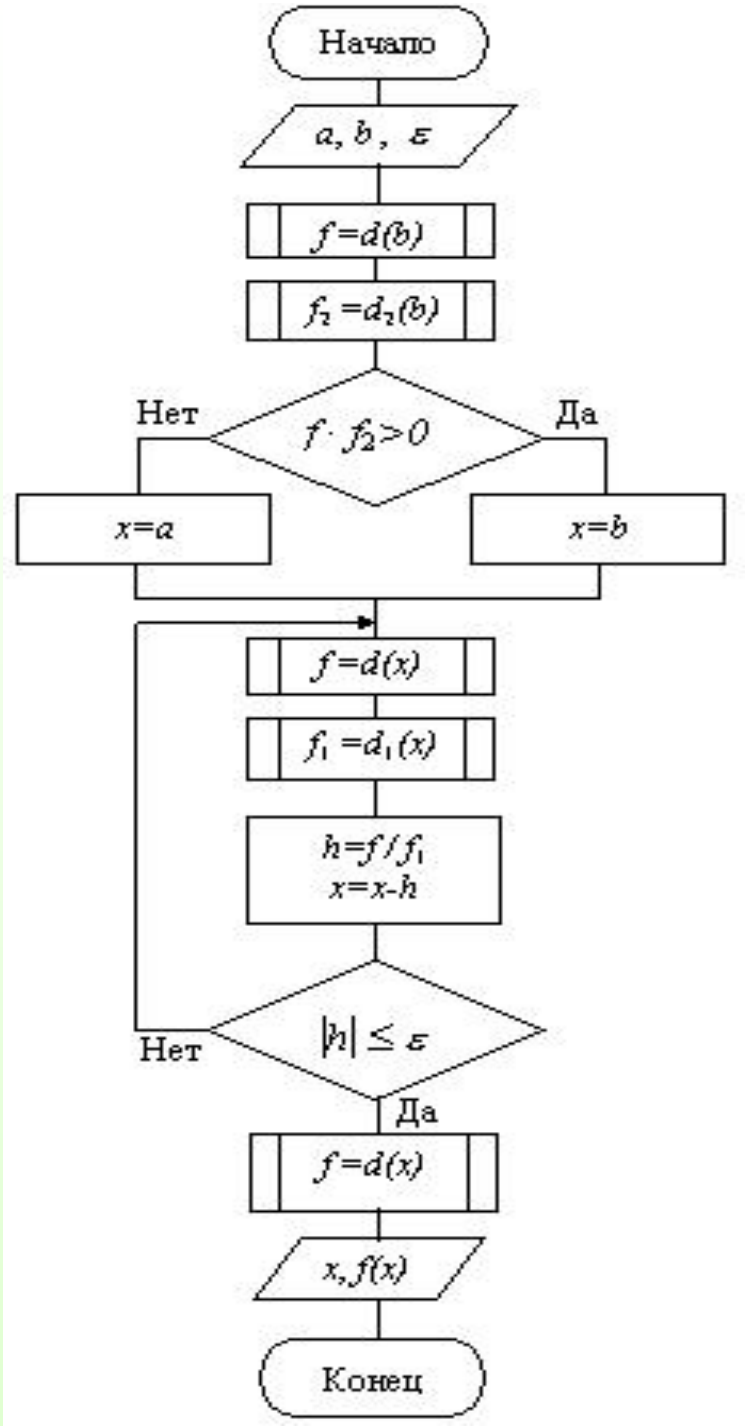
$$\left|\frac{f(x_n)}{f'(x_n)}\right| \le \varepsilon. \qquad (4.7)$$

Метод обеспечивает быструю сходимость, если выполняется условие:

$$f(x_0)\,f''(x_0) > 0, \quad (4.8)$$

т.е. первую касательную рекомендуется проводить в той точке интервала [a,b], где знаки функции $f(x_0)$ и ее кривизны $f''(x_0)$ совпадают.

Блок-схема алгоритма уточнения корня методом Ньютона



Модифицированный метод Ньютона (метод секущих)

В этом методе производная на каждом шаге поиска вычисляется приближённо по результатам двух предыдущих итераций:

$$f'(x) = \frac{\Delta f(x)}{\Delta x}$$

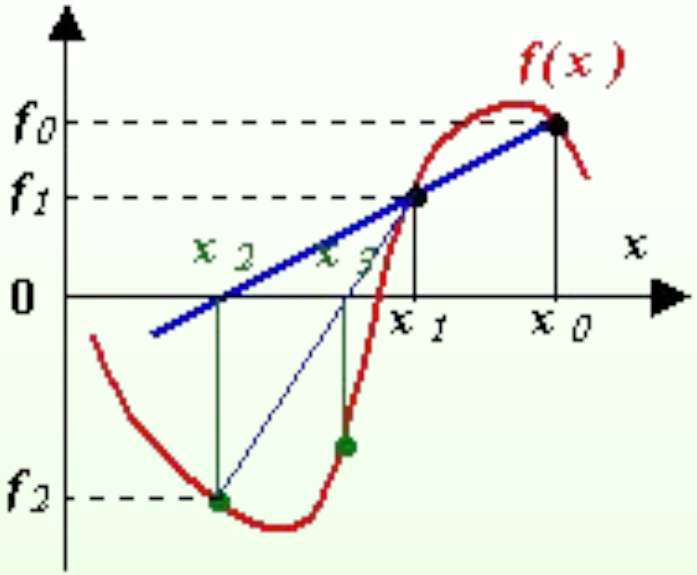
и рекуррентная формула (4.5) имеет вид:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{f(x_n)\Delta x}{\Delta f(x_n)} = \tag{4.9}$$
$$= x_n - \frac{f(x_n)\Delta x}{f(x_n+\Delta x)-f(x_n)},$$

где $\Delta x \approx \varepsilon$. Выход из итерационного процесса по условию (4.6):

$$\left|x_{n+1} - x_n\right| \leq \varepsilon$$

Рис.5. Геометрическая иллюстрация метода секущих



Сходимость может быть немонотонной даже вблизи корня. При этом вблизи корня может происходить потеря точности, т.н. «разболтка решения», особенно значительная в случае кратных корней. От разболтки страхуются *приёмом Гарвика*: выбирают некоторое ε_x и ведут итерации до выполнения условия $|x_{n+1}-x_n| \le \varepsilon_x$. Затем продолжают расчет, пока убывает $|x_{n+1}-x_n|$. Первое же возрастание может свидетельствовать о начале разболтки, а значит, расчет следует прекратить, а последнюю итерацию не использовать.

Метод одной касательной

Заметим, что в методе секущих удобно было бы фиксировать наиболее подходящее для первого шага значение $\lambda_0 = \frac{1}{f'(x_0)}$,

при котором все секущие параллельны касательной, проведённой к графику $y = f(x)$ при $x = x_0$. При таком выборе метод секущих называется методом одной касательной.

Итерационная формула этого метода имеет вид:

$$x_{i+1} = x_i - \frac{1}{f'(x_0)} f(x_i).$$

Как видно из этой формулы, производную придётся вычислить только один раз, а затем на каждом шаге использовать значение $\lambda_0 = \frac{1}{f'(x_0)}$ или, что то же, $k = f'(x_0)$.

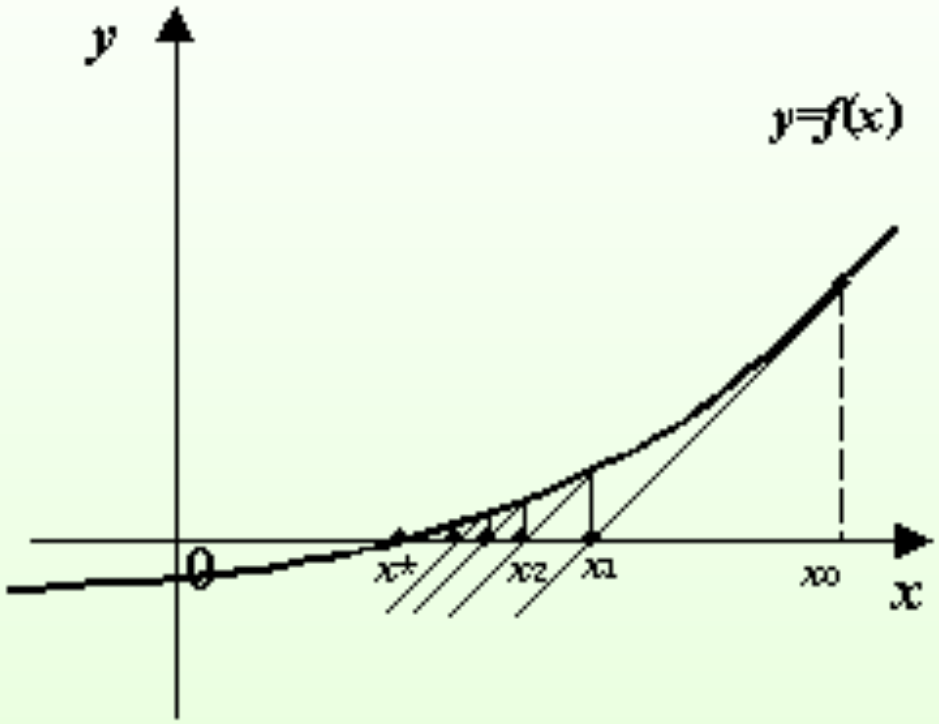


Рис. 6. Геометрическая иллюстрация сходимости метода метода одной касательной

При таком выборе λ_0 в точке выполнено равенство

$$\varphi'(x_0) = 1 - \lambda_0 f'(x_0) = 0,$$

и если отрезок, на котором отделён корень и выбрано начальное приближение x_0 достаточно мал, а производная не[illegible]ывна, то значение буд[illegible]льно отличаться от следовательно, гра[illegible] будет пересекать прямую , идя почти горизонтально. А это будет давать нам быстрое приближение итераций к корню.

Метод простых итераций

Пусть с точностью ε необходимо найти корень уравнения $f(x)=0$, принадлежащий интервалу изоляции $[a,b]$. Функция $f(x)$ и ее первая производная непрерывны на этом отрезке.

Для применения этого метода исх равнение $f(x)=0$ должно быть приведено к виду (4.10)

Это преобразование можно сделать по разному и получить различного вида функции, в зависимости от вида $f(x)$. Например, можно положить $\varphi(x) = x + \tau f(x)$ (4.11)

Где $\tau = const$, при этом корни исходного уравнения не изменятся. В качестве начального приближения 0 выбираем любую точку интервала $[a,b]$. Далее итерационный процесс поиска корня строится по схеме:

$x_{\dots} = \varphi(x_{\dots})$ (4.12)

Процесс поиска прекращается, как только выполняется условие
(4.13)

или число итераций превысит заданное число N.

Для того, чтобы последовательность $x_1, x_2, \ldots, x_n$ приближалась к искомо, необходимо, чтобы выполнялось условие сходимости:
(4.14)

При этом принято различать три вида сходимости:

$|\varphi'(x)| < 0{,}1$ – хорошая сходимость;

$0{,}1 < |\varphi'(x)| < 0{,}5$ – удовлетворительная сходимость;

$0{,}5 < |\varphi'(x)| < 1{,}0$ – плохая сходимость.

Исследуем выбор константы τ в уравнении (4.11):

$\varphi(x) = x + \tau f(x)$ с точки зрения обеспечения максимальной скорости сходимости. В соответствии с критерием сходимости наибольшая скорость сходимости обеспечивается при $|\varphi'(x)| = 0$. При этом, исходя из (4.11), $\tau = -1/f'(x)$, и итерационная формула (4.12): $x_n = \varphi(x_{n-1})$ переходит в

т.е. в формулу метода Ньютона (4.5). Таким образом, метод Ньютона является частным случаем метода простых итераций, обеспечивающим самую высокую скорость сходимости из всех возможных вариантов выбора функции $\varphi(x)$.

Рис. 7. Геометрический смысл метода простых итераций, сходящиеся итерационные процессы,

Рис. 8. В случае немонотонной функции $\varphi(x)$ сходящиеся итерации могут вести себя нерегулярно.

Рис. 9. Расходящиеся итерационные процес

Если не выполнено ни условие , ни условие , то итерации могут зацикливаться.

Рис.10. Пример зацикливания итераций

Блок-схема алгоритма уточнения корней методом простых итераций

Точность приближений $x_{n,}$ т.е. их близость к истинному значению корня x^*, оценивается по формуле:

$$|x_n - x^*| \geq (g/(1-g))\ |x_n - x_{n-1}|,$$

где g определяется как наибольшее значение производной $\varphi'(x)$, удовлетворяющее неравенству на отрезке $[a,b]$: $|\varphi'(x)| \leq g < 1$.

Из этого условия следует, что если g близко к 1, то x_n далеко от x^* даже тогда, когда x_{n-1} близко к x_n, так как

$g/(1-g)$ велико. Поэтому получаемое по условию приближённое значение корня в этом случае будет неточным. Это простое условие окончания итерационного процесса можно использовать, если .

Достоинство метода простых итераций состоит в том, что при удачном выборе функции $\varphi(x)$ он обеспечивает быструю и надёжную сходимость. Однако, для сложных уравнений получение такого вида функции связано с большой предварительной работой, а иногда и вообще не представляется возможным.

Если итерационный процесс расходится, то как можно обеспечить его сходимость?

Есть несколько способов.

Пример

Решение нелинейных уравнений в MATLAB

(уравнений вида f(x)=0)

fzero ('name',x0) – где name – имя файл-функции, вычисляющей левую часть уравнения, x0 – начальное приближение к корню; **fzero ('name',[a,b])** - [a,b] – интервал изоляции корня.

Пример: Найти решение уравнения f(x)= x-sin(x)-0.25.

```
function y=f(x)
y=x-sin(x)-0.25
end
>>fzero('f',1)
```

- Важная особенность указанной команды – возможность выдавать не только значение корня, но и значение функции в этой точке.

Пример.

Найти корни уравнения $(e^x/5)-2(x-1)^2=0$. Определено, что имеется три корня уравнения и интервалы их изоляции.

В M-файле с именем ff.m пишем:

```
function y=ff(x)
y=exp(x)/5-2*(x-1).^2;
end
```

Потом в командном окне пишем:

%Построение графика

```
x=-0.5:0.1:5.5;
y=exp(x)/5-2*(x-1).^2;
plot(x,y,'-k'), grid
```

или

```
ezplot('exp(x)/5-2*(x-1)^2')
grid on
```

```
[x(1),y(1)]=fzero('ff',[0 1]);
[x(2),y(2)]=fzero('ff',[1 2]);
[x(3),y(3)]=fzero('ff',[5 6]);
x
y
```

Пример.

Найти корень уравнения $10^x+2x-100=0$ на интервале

$[1,0;2,0]$

- Для поиска корней полинома используется функция **roots(c)**,

  c – вектор коэффициентов полинома в порядке убывания степеней и наличие нулевых коэффициентов обязательно;
- Наоборот, построить вектор p по заданному вектору его корней можно с помощью функции **poly(x)**.

Пример: Вычислить корни полинома $x^3+0.4x^2+0.6x-1=0$

```
>>p=[1 0.4 0.6 -1];
>>x= roots(p)
>>x
```

Функции, предназначенные для действий над полиномами:

conv(p1, p2) - формирует вектор, соответствующий коэффициентам полинома степени n+m, полученного в результате умножения вектора p1 степени n на вектор p2 степени m, то есть вычисляет произведение двух полиномов;

deconv(pl, p2) -осуществляет деление полинома p1 на полином p2, то есть формирует вектор коэффициентов полинома, который является частным от деления p1 на p2; вызов функции в общем виде [q, r]=deconv(pl, p2) приводит к получению двух полиномов: q - частного от деления и r -остатка от деления;

polyval(p1, x) - вычисляет значение полинома с коэффициентами p1 в точке x;

polyder (p1 [, p2]) - формирует вектор коэффициентов полинома, являющегося производной от полинома с коэффициентами p1, то есть вычисляет производную от полинома; если в качестве аргументов указаны два вектора polyder(p1, p2), то производная вычисляется от произведения этих векторов, вызов функции в общем виде [q, r]= polyder (p1, p2) приведет к вычислению производной от частного pl/p2 и выдаст результат в виде

Пример(использование *conv* и *deconv*): pl=[2 0 1]; p2=[1 0 0 -1];

%Умножение полиномов: (2x^2+1) (x^3-1) = 2x^5+x^3-2x^2-1

p=conv(p2,pl)

Результат: p =

2 0 1 -2 0 -1

%Деление полиномов: (2x^5+x^3-2x^2-1) / (x^3-1) = (2x^2+1)

deconv(p,p2)

Результат: ans =

2 0 1

Пример(использование *polival*): p1=[2 1 0 -1 0 -3];

%Значение полинома (2x^5+x^4-x^2-3) в точке x=0

polyval(p1,0)

Результат: ans =

-3

Пример(использование *polider*): p1=[2 1 0 -1 0 -3];

polyder(p1); %Производная от p1

p2=[1 0 0 -1];

polyder(p1,p2); %Производная от (p1*p2)

[q, r]=polyder(p1,p2) %Производная от частного и остатка (p1/p2)

Результат: q =

4 1 0 -9 -4 9 2 0

r =

1 0 0 -2 0 0 1

Пример(использование *roots*):Найти корни полинома $2x^4 - 8x^3 + 8x^2 - 1 = 0$.

p=[2 -8 8 0 -1];

roots(p)

%Найдем графическое решение

x=-1:0.1:3;

y=polyval(p,x);

plot(x,y,'-k'),grid

Задание.

Решить нелинейные уравнения с точностью 0,0001 указанными в таблице методами, предварительно определив интервалы, на которых существуют решения уравнений. Фиксировать число требуемых итераций. Сделать проверку решения, подставив найденные корни в уравнения, а также решить уравнения, используя стандартные операторы MATLAB. Сравнить результаты.

Варианты уравнений и методов их решения приведены в таблице.

Вар.	Уравнение	Методы решения
1	$x=\exp(-x)$	Перебора и простых итераций
2	$x=\cos(x)$	Перебора и хорд
3	$x=x^2-1$	Перебора и касательных
4	$x=2\exp(-x)$	Перебора и секущих
5	$x=\exp(-3x)$	Перебора и половинного деления