19 Lecture #13: Tuesday, March 31st, 2026

19.1 Finding roots of equations numerically

In elementary courses, many equations can be solved exactly: linear equations, some quadratic equations and a few additional special cases. In practice, however, most nonlinear equations do *not* admit a simple closed-form expression for their solutions. For instance, the equation $\cos x = x$ is easy to state (see Figure 81), but its solution cannot be written in terms of elementary algebraic operations. In such situations, numerical methods are employed to compute an approximation of the actual root. To establish precise terminology and ensure a common framework, we introduce the following definition.

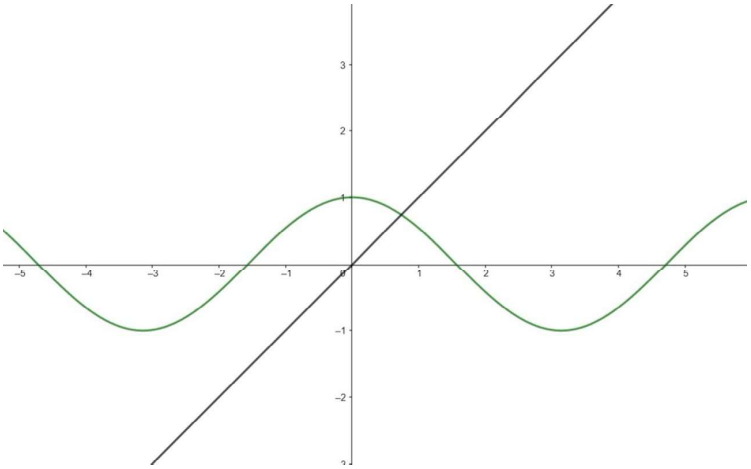

Figure 81: Graphs of the functions $y = \cos(x)$ and $y = x$.

Definition 19.1. By a **root** of a function f we mean any number r such that $f(r) = 0$.

Our goal is to present several numerical methods for approximating a root. The numerical objective is typically the following: given a tolerance $\varepsilon > 0$, find an approximation $\hat{r}$ such that the error is small, ideally satisfying $|\hat{r} - r| < \varepsilon$, where r denotes the exact root. However, there is no universal procedure that guarantees the existence of a root for an arbitrary function. Existence must instead be established from the properties of the function. One of the most useful results from Calculus for this purpose is the Intermediate Value Theorem.

Theorem 19.2 (Intermediate Value Theorem)**.** Let f be continuous on an interval $[a, b]$. If $f(a)$ and $f(b)$ have opposite signs, that is, if $f(a) \cdot f(b) < 0$, then there exists at least one $r \in (a, b)$ such that $f(r) = 0$.

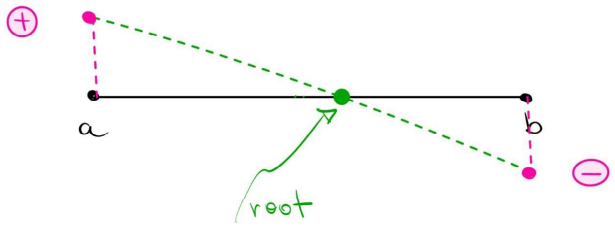

Figure 82: The Intermediate Value Theorem.

As we will see in the next section, the Intermediate Value Theorem provides the foundation for the bisection method. It also explains why one often begins by searching for two points a and b at which the function changes sign (see Figure 82). Nevertheless, a root may exist even when no sign change occurs. For example, one of the simplest functions exhibiting this behavior is $f(x) = x^2$. This function has a root at $x = 0$, but since $f(x) \geqslant 0$ for all x, the Intermediate Value Theorem cannot detect it (see Figure 83). This is a typical phenomenon when dealing with roots of higher multiplicity.

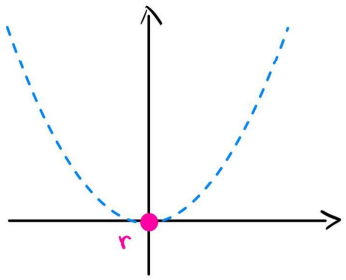

Figure 83: A root that the Intermediate Value Theorem cannot identify where it is.

19.2 The bisection method

Assume that f is continuous on $[a, b]$ and that $f(a) \cdot f(b) < 0$. Then there is at least one root in (a, b) by the Intermediate Value Theorem. The bisection method repeatedly cuts the interval in half while looking for where the root is by once again applying the Intermediate Value Theorem. Indeed, let $m = \frac{a+b}{2}$ be the midpoint of the interval. We have the following.

- If $f(m) = 0$, then m is a root.
- If $f(a) \cdot f(m) < 0$, then the root lies in $[a, m]$.
- If $f(m) \cdot f(b) < 0$, then the root lies in $[m, b]$.

Thus we replace $[a, b]$ by the half-interval that still contains a root and repeat the procedure over and over.

Algorithm 19.3 (bisection method of finding root of f)**.** Let f be a continuous function on interval $[a, b]$ and consider a tolerance ε. Assume that $f(a)$ and $f(b)$ have opposite signs.

0. Set $a_0 = a$, $b_0 = b$.

1. Assumption: $f(a_k)$ and $f(b_k)$ have opposite signs. Let $m_k = \frac{1}{2}(a_k + b_k) = a_k + \frac{1}{2}(b_k - a_k)$.

2. If $f(m_k) = 0$ or $|b_k - a_k| < \varepsilon$ then algorithm stops, output is m_k. Otherwise:

If $f(a_k)$ and $f(m_k)$ have opposite signs, set $a_{k+1} = a_k$, $b_{k+1} = m_k$, increase k by one and go back to step 1.

If $f(m_k)$ and $f(b_k)$ have opposite signs, set $a_{k+1} = m_k$, $b_{k+1} = b_k$, increase k by one and go back to step 1.

Notice that by the previous algorithm, the approximation after k steps is usually taken to be m_k.

Example 19.4. Consider $f(x) = x^3 - x - 10$. We compute $f(0) = -10 < 0$ and $f(5) = 110 > 0$. Hence there is at least one root in $(0, 5)$. The first midpoint is

$$m_0 = \frac{0+5}{2} = 2.5.$$

Since $f(2.5) = 2.5^3 - 2.5 - 10 = 3.125 > 0$, the root lies in $[0, 2.5]$. Next,

$$m_1 = \frac{0+2.5}{2} = 1.25$$

and $f(1.25) = 1.25^3 - 1.25 - 10 < 0$, so the root lies in $[1.25, 2.5]$. Continuing in this way, we obtain a sequence of intervals whose lengths tend to zero. See in Figure 84 that the algorithm stops after reaching the tolerance as is set to be 0.001.

```
> myf:=x^3-x-10;
```

$$myf := x^3 - x - 10$$

```
> Root(myf,xinit=[0.,5.],method=bisection,tolerance=0.001,digits=7);
k=00  a=  0.0000000  b=  5.0000000  x=  2.5000000  f(x)=  3.1250000  test= 2.5000000
k=01  a=  0.0000000  b=  2.5000000  x=  1.2500000  f(x)= -9.2968750  test= 1.2500000
k=02  a=  1.2500000  b=  2.5000000  x=  1.8750000  f(x)= -5.2832031  test= 0.6250000
k=03  a=  1.8750000  b=  2.5000000  x=  2.1875000  f(x)= -1.7199707  test= 0.3125000
k=04  a=  2.1875000  b=  2.5000000  x=  2.3437500  f(x)=  0.5308533  test= 0.1562500
k=05  a=  2.1875000  b=  2.3437500  x=  2.2656250  f(x)= -0.6360435  test= 0.0781250
k=06  a=  2.2656250  b=  2.3437500  x=  2.3046875  f(x)= -0.0631452  test= 0.0390625
k=07  a=  2.3046875  b=  2.3437500  x=  2.3242188  f(x)=  0.2311942  test= 0.0195312
k=08  a=  2.3046875  b=  2.3242188  x=  2.3144531  f(x)=  0.0833624  test= 0.0097656
k=09  a=  2.3046875  b=  2.3144531  x=  2.3095703  f(x)=  0.0099434  test= 0.0048828
k=10  a=  2.3046875  b=  2.3095703  x=  2.3071289  f(x)= -0.0266421  test= 0.0024414
k=11  a=  2.3071289  b=  2.3095703  x=  2.3083496  f(x)= -0.0083597  test= 0.0012207
k=12  a=  2.3083496  b=  2.3095703  x=  2.3089600  f(x)=  0.0007893  test= 0.0006104

                                                    2.308959960
```

Figure 84: The approximation of the root of $f(x) = x^3 - x - 10$ after 12 iterations using bisection.

The bisection method is a typical example of a bracketing method[‡‡]. That is why that one of the main strengths of this method is that **the error can be controlled** explicitly. Indeed, let r be the exact root contained in the interval $[a_k, b_k]$. Since m_k is its midpoint, we have that

$$|E_k| = |r - m_k| \leqslant \frac{b_k - a_k}{2}.$$

But each iteration halves the interval length. This means that in every step we get a multiplication by $1/2$, that is, we can continue the above inequality and then get

$$|E_k| \leqslant \frac{b_k - a_k}{2} \leqslant \frac{1}{2} \cdot \frac{b_{k-1} - a_{k-1}}{2} \leqslant \ldots \leqslant \frac{1}{2^{k+1}} |b_0 - a_0|$$

where $[a_0, b_0] = [a, b]$ is the original interval that we have started the algorithm with. This shows that the error is nicely controlled by a data we know. In fact, it is a **predictable** method.

[‡‡]A **bracketing method** is a **root-finding method** that begins with an interval $[a, b]$ such that the root is trapped, or *bracketed*, inside it. Typically, one assumes that f is continuous on $[a, b]$ and that $f(a)$ and $f(b)$ have opposite signs. By the Intermediate Value Theorem, this guarantees the existence of at least one root $r \in (a, b)$. The method then repeatedly replaces the original interval by a smaller subinterval that still contains a sign change, and therefore still brackets a root.

Notice, however, that although the bisection method is reliable, it is not particularly aggressive. In other words, its rate of convergence is rather **slow**: the method typically gains only 1 decimal digit of accuracy every 3 or 4 additional iterations (see the "test" part in Figure 85 and Figure 86). For instance, when the tolerance is 0.0001, one observes that the method requires approximately 3-4-3 iterations to gain each additional correct digit. Similarly, for a tolerance of 0.0000001, the pattern is approximately 3-4-3-3-4-3 iterations per additional reliable digit.

```
> Root(myf,xinit=[0.,5.],method=bisection,tolerance=0.0001,digits=7);
k=00  a=  0.0000000  b=  5.0000000  x=  2.5000000  f(x)=  3.1250000  test= 2.5000000
k=01  a=  0.0000000  b=  2.5000000  x=  1.2500000  f(x)= -9.2968750  test= 1.2500000
k=02  a=  1.2500000  b=  2.5000000  x=  1.8750000  f(x)= -5.2832031  test= 0.6250000
k=03  a=  1.8750000  b=  2.5000000  x=  2.1875000  f(x)= -1.7199707  test= 0.3125000
k=04  a=  2.1875000  b=  2.5000000  x=  2.3437500  f(x)=  0.5308533  test= 0.1562500
k=05  a=  2.1875000  b=  2.3437500  x=  2.2656250  f(x)= -0.6360435  test= 0.0781250
k=06  a=  2.2656250  b=  2.3437500  x=  2.3046875  f(x)= -0.0631452  test= 0.0390625
k=07  a=  2.3046875  b=  2.3437500  x=  2.3242188  f(x)=  0.2311942  test= 0.0195312
k=08  a=  2.3046875  b=  2.3242188  x=  2.3144531  f(x)=  0.0833624  test= 0.0097656
k=09  a=  2.3046875  b=  2.3144531  x=  2.3095703  f(x)=  0.0099434  test= 0.0048828
k=10  a=  2.3046875  b=  2.3095703  x=  2.3071289  f(x)= -0.0266421  test= 0.0024414
k=11  a=  2.3071289  b=  2.3095703  x=  2.3083496  f(x)= -0.0083597  test= 0.0012207
k=12  a=  2.3083496  b=  2.3095703  x=  2.3089600  f(x)=  0.0007893  test= 0.0006104
k=13  a=  2.3083496  b=  2.3089600  x=  2.3086548  f(x)= -0.0037859  test= 0.0003052
k=14  a=  2.3086548  b=  2.3089600  x=  2.3088074  f(x)= -0.0014985  test= 0.0001526
k=15  a=  2.3088074  b=  2.3089600  x=  2.3088837  f(x)= -0.0003546  test= 0.0000763

                                                    2.308883666
```

Figure 85: The approximation of the root of $f(x) = x^3 - x - 10$ with tolerance of 0.0001.

```
> Root(myf,xinit=[0.,5.],method=bisection,tolerance=0.0000001,digits=7);
k=00  a=  0.0000000  b=  5.0000000  x=  2.5000000  f(x)=  3.1250000  test= 2.5000000
k=01  a=  0.0000000  b=  2.5000000  x=  1.2500000  f(x)= -9.2968750  test= 1.2500000
k=02  a=  1.2500000  b=  2.5000000  x=  1.8750000  f(x)= -5.2832031  test= 0.6250000
k=03  a=  1.8750000  b=  2.5000000  x=  2.1875000  f(x)= -1.7199707  test= 0.3125000
k=04  a=  2.1875000  b=  2.5000000  x=  2.3437500  f(x)=  0.5308533  test= 0.1562500
k=05  a=  2.1875000  b=  2.3437500  x=  2.2656250  f(x)= -0.6360435  test= 0.0781250
k=06  a=  2.2656250  b=  2.3437500  x=  2.3046875  f(x)= -0.0631452  test= 0.0390625
k=07  a=  2.3046875  b=  2.3437500  x=  2.3242188  f(x)=  0.2311942  test= 0.0195312
k=08  a=  2.3046875  b=  2.3242188  x=  2.3144531  f(x)=  0.0833624  test= 0.0097656
k=09  a=  2.3046875  b=  2.3144531  x=  2.3095703  f(x)=  0.0099434  test= 0.0048828
k=10  a=  2.3046875  b=  2.3095703  x=  2.3071289  f(x)= -0.0266421  test= 0.0024414
k=11  a=  2.3071289  b=  2.3095703  x=  2.3083496  f(x)= -0.0083597  test= 0.0012207
k=12  a=  2.3083496  b=  2.3095703  x=  2.3089600  f(x)=  0.0007893  test= 0.0006104
k=13  a=  2.3083496  b=  2.3089600  x=  2.3086548  f(x)= -0.0037859  test= 0.0003052
k=14  a=  2.3086548  b=  2.3089600  x=  2.3088074  f(x)= -0.0014985  test= 0.0001526
k=15  a=  2.3088074  b=  2.3089600  x=  2.3088837  f(x)= -0.0003546  test= 0.0000763
k=16  a=  2.3088837  b=  2.3089600  x=  2.3089218  f(x)=  0.0002173  test= 0.0000381
k=17  a=  2.3088837  b=  2.3089218  x=  2.3089027  f(x)= -0.0000687  test= 0.0000191
k=18  a=  2.3089027  b=  2.3089218  x=  2.3089123  f(x)=  0.0000743  test= 0.0000095
k=19  a=  2.3089027  b=  2.3089123  x=  2.3089075  f(x)=  0.0000028  test= 0.0000048
k=20  a=  2.3089027  b=  2.3089075  x=  2.3089051  f(x)= -0.0000329  test= 0.0000024
k=21  a=  2.3089051  b=  2.3089075  x=  2.3089063  f(x)= -0.0000150  test= 0.0000012
k=22  a=  2.3089063  b=  2.3089075  x=  2.3089069  f(x)= -0.0000061  test= 0.0000006
k=23  a=  2.3089069  b=  2.3089075  x=  2.3089072  f(x)= -0.0000016  test= 0.0000003
k=24  a=  2.3089072  b=  2.3089075  x=  2.3089074  f(x)=  0.0000006  test= 0.0000001
k=25  a=  2.3089072  b=  2.3089074  x=  2.3089073  f(x)= -0.0000005  test= 0.0000001

                                                    2.308907285
```

Figure 86: The approximation of the root of $f(x) = x^3 - x - 10$ with tolerance of 0.0000001.

Mathematically, the pattern 3-4-3-3-... comes from the following fact. As shown earlier, we have

$$|E_k| = |r - m_k| \leqslant \frac{1}{2}|a_k - b_k| =: e_k.$$

At each step, the quantity e_k is reduced by a factor of $\frac{1}{2}$, that is,

$$e_{k+1} \leqslant \frac{1}{2}e_k.$$

Therefore, after 3 and 4 additional iterations, we obtain, respectively,

$$e_{k+3} \leqslant \frac{1}{8}e_k \qquad \text{and} \qquad e_{k+4} \leqslant \frac{1}{16}e_k.$$

This shows that one new decimal digit of accuracy is gained between the third and the fourth iteration. Moreover, after 10 iterations, we have

$$e_{k+10} \leqslant \frac{1}{2^{10}}e_k = \frac{1}{1024}e_k \approx \frac{1}{1000}e_k.$$

Hence, after 10 iterations, the error is reduced by approximately three decimal orders of magnitude, which means that we gain about 3 new reliable digits. This means that this method is slow. One of the reasons why it has this behavior it is because it ignores the shape of the graph and only uses sign information. For this reason, bisection is often used either as a safe standalone method or as a first step to obtain a good initial guess for a faster method.

19.3 The Newton method

Let us now turn to a second method for approximating roots of functions: the Newton method. Unlike the bisection method, which improves an approximation by repeatedly shrinking an interval, Newton's method uses the local linear behavior of the function. More precisely, it replaces the function near a given point by its tangent line and takes the intersection of that tangent line with the x-axis as a new approximation to the root. The main idea behind Newton's method is both simple and powerful. If a current approximation is already reasonably close to a root, then the graph of the tangent line at that point often provides a much better estimate than the previous one. In this way, the method uses not only the value of the function, but also its derivative, to move rapidly toward the solution. As we will see, this geometric idea leads to one of the most important and efficient methods in numerical analysis.

Consider x_0 to be a point close to the root r as in the Figure 87.

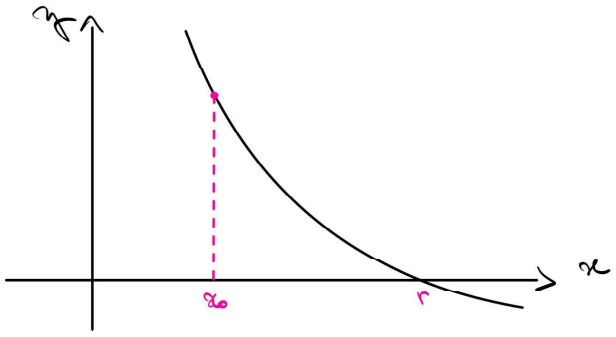

Figure 87: First step of the Newton method. Pick a point x_0 relatively close to the root.

Now, take the tangent line at $(x_0, f(x_0))$ and prolong it until it touches the x-axes and this is going to be our new approximation to the room x_1. Repeat the procedure (see Figure 88).

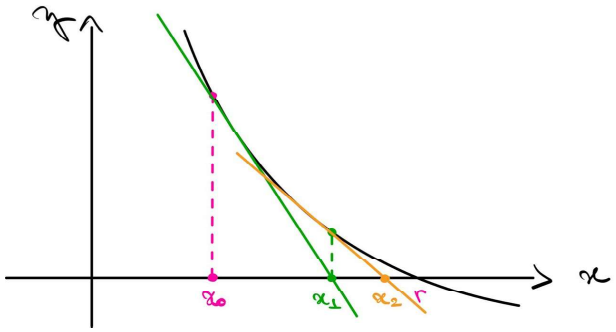

Figure 88: Second and third steps of the Newton method.

In general, we have the following. Suppose x_k is close to a root r of f. The tangent line at this point is given by

$$y = f(x_k) + f'(x_k)(x - x_k).$$

The next approximation x_{k+1} is taken to be the x-intercept of this tangent line, so we set $y = 0$ in the above equation and solve it for x_{k+1} as below

$$0 = f(x_k) + f'(x_k)(x_{k+1} - x_k).$$

Hence, in order to find the next approximation x_{k+1} we use the following formula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{23}$$

Algorithm 19.5 (Newton method for finding root of a function f)**.** Given: a differentiable function f and a tolerance ε.

0. Choose x_0.

1. Let $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

If $|x_{k+1} - x_k| < \varepsilon$ or $|f(x_{k+1})| < \varepsilon$ then the algorithm stops, output is x_{k+1}.

Otherwise increase k by one and go back to step 1.

Example 19.6. Again consider the function $f(x) = x^3 - x - 10$ with derivative $f'(x) = 3x^2 - 1$. Take $x_0 = 5$. Then, $f(5) = 110$ and $f'(5) = 74$, so

$$x_1 = 5 - \frac{110}{74} = 5 - \frac{55}{37} \approx 3.5135135.$$

Repeating the formula gives a sequence that converges rapidly to the root. The two computations in Figure 89 and Figure 90 show that Newton's method is remarkably fast. In both cases, despite using different tolerances, the method reaches the approximation after only 6 iterations. This is a substantial improvement over the bisection method, which required 15 iterations for the first tolerance (see Figure 85) and 25 iterations for the second (see Figure 86). Thus, while the bisection method is reliable but slow,

Newton's method achieves high accuracy with far fewer steps. This example clearly illustrates the main advantage of Newton's method: when it works well, it converges much more rapidly than bisection.

```
> myf:=x^3-x-10;
                                                    myf := x^3 - x - 10
:
> Root(myf,xinit=5,method=newton,tolerance=0.0001,digits=7);
k=00  x=  5.0000000   f(x)= 110.0000000  test= 110.0000000
k=01  x=  3.5135135   f(x)= 29.8600280  test= 1.4864865
k=02  x=  2.6848585   f(x)=  6.6688514  test= 0.8286550
k=03  x=  2.3615265   f(x)=  0.8082521  test= 0.3233320
k=04  x=  2.3101450   f(x)=  0.0185680  test= 0.0513815
k=05  x=  2.3089080   f(x)=  0.0000106  test= 0.0012370
k=06  x=  2.3089073   f(x)=  0.0000000  test= 0.0000007
Root (probably) found.
                                                          2.308907320
.
```

Figure 89: Newton's method for the function $f(x) = x^3 - x - 10$ with tolerance of 0.0001.

```
> Root(myf,xinit=5,method=newton,tolerance=0.0000001,digits=7);
k=00  x=  5.0000000   f(x)= 110.0000000  test= 110.0000000
k=01  x=  3.5135135   f(x)= 29.8600280  test= 1.4864865
k=02  x=  2.6848585   f(x)=  6.6688514  test= 0.8286550
k=03  x=  2.3615265   f(x)=  0.8082521  test= 0.3233320
k=04  x=  2.3101450   f(x)=  0.0185680  test= 0.0513815
k=05  x=  2.3089080   f(x)=  0.0000106  test= 0.0012370
k=06  x=  2.3089073   f(x)=  0.0000000  test= 0.0000007
Root (probably) found.
                                                          2.308907320
```

Figure 90: Newton's method for the function $f(x) = x^3 - x - 10$ with tolerance of 0.0000001.

When Newton's method works well (which is not always the case as we will see in a moment), it is much faster than bisection. Once the approximation is sufficiently close to a simple root, the number of correct digits increases fastly.

Remark 19.7. Although Newton's method is often very fast, it does not always work. Its success depends strongly on the choice of the initial approximation and on the shape of the function. In some situations, the method may stop, fail to improve, or even move away from the root instead of approaching it. Some common difficulties are the following. In fact, there is a huge dependence on the initial guess and this is one of the main weaknesses of Newton's method.

(a) If the derivative is zero at some iterate, that is, if $f'(x_k) = 0$, then Newton's formula $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ is not defined. Geometrically, this means that the tangent line at x_k is horizontal, so it does not intersect the x-axis in a useful way for the method. Therefore, the iteration cannot continue. A common case where this happens is when the function as an extreme point at x_k (see Figure 91).

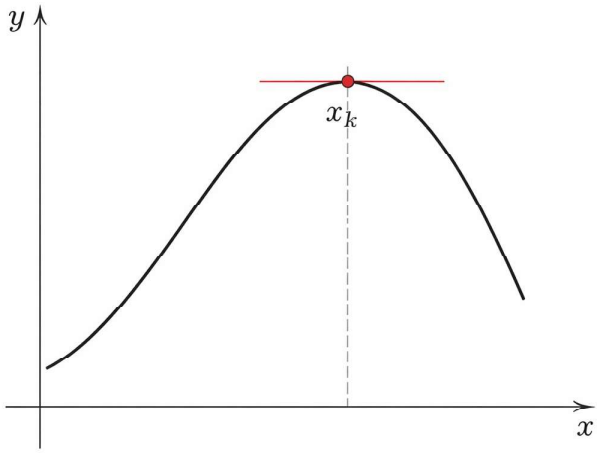

Figure 91: The tangent is horizontal and the Newton method does not work in this case.

(b) Consider the function $f(x) = \sqrt{x}$. In this case, Newton's iteration sends each value x_k to $-x_k$. Thus, the method jumps back and forth from one side to the other instead of approaching the root, so it does not work properly (see Figure 92).

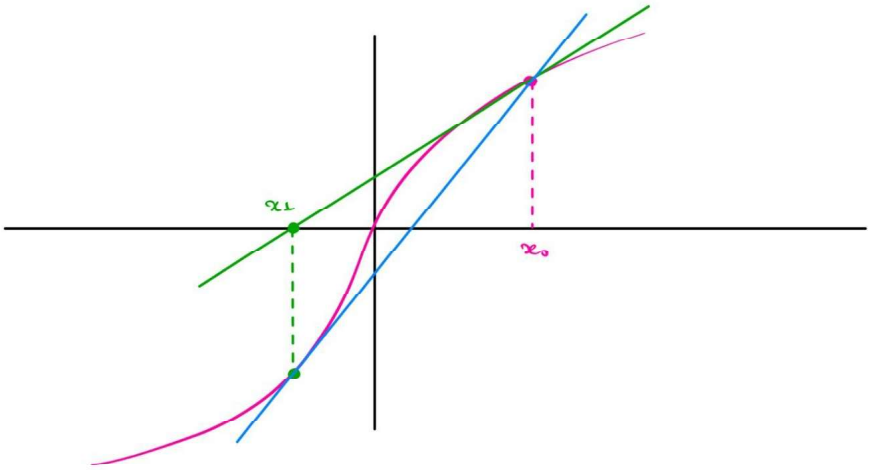

Figure 92: Newton's method does not work for the function $f(x) = \sqrt{x}$.

(c) There are also functions whose graphs have shapes for which the tangent lines send the iterates alternately from one side of the root to the other without converging. In such cases, the approximations may simply oscillate or "dance around" rather than settle down toward the root (see Figure 93).

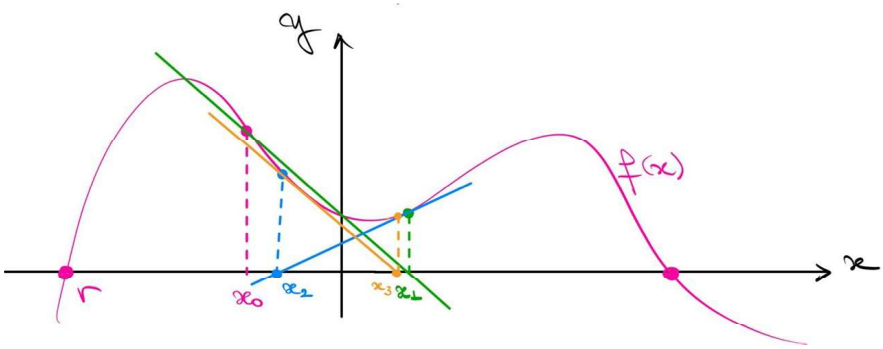

Figure 93: The tangent lines are just "dancing around" and the approximation is not obtained.

(d) There are also situations in which Newton's method does not merely fail to converge, but instead sends the approximations farther and farther away.

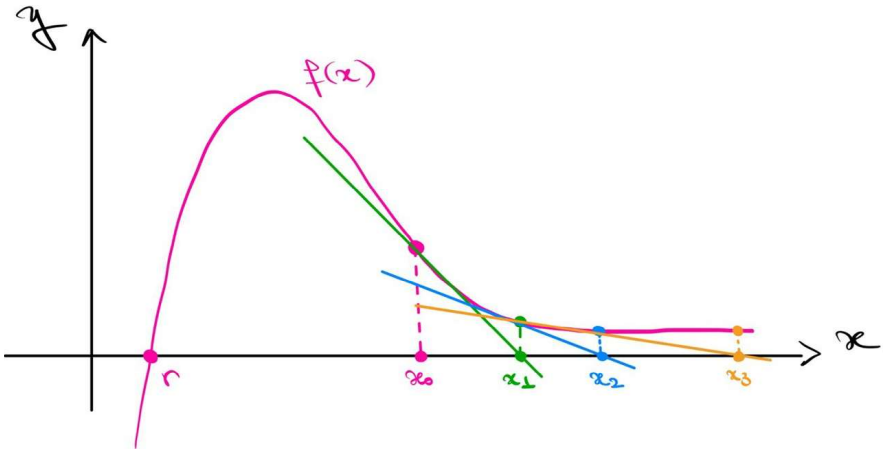

Figure 94: The approximations are going farther and farther away.

There are situations in which the convergence of Newton's method can be guaranteed (see Theorem 19.8 below). However, to obtain such a guarantee, one usually needs additional information about the derivatives of the function involved. In particular, the analysis often requires assumptions on f' and f'', and although these conditions are mathematically very useful, they may not always be easy to verify in practice.

Theorem 19.8. Let f be a function on an interval $[a, b]$ such that $f(a) \cdot f(b) < 0$. Assume that f is twice continuously differentiable on (a, b) and $f' \neq 0$, $f'' \neq 0$ on (a, b). If $x_0 \in (a, b)$ is chosen so that $f(x_0) \cdot f''(x_0) > 0$, then the sequence x_n generated by the Newton method converges and it converges to a root $r \in [a, b]$ of f.

Geometrically, this theorem says that Newton's method behaves well when the graph has a consistent shape and when the initial point is chosen on the "correct side". The assumptions $f'(x) \neq 0$ and $f''(x) \neq 0$ mean that, on the interval, the graph has no horizontal tangents and no inflection points: it is always increasing or always decreasing, and it is always either concave up or concave down. Therefore, the curve bends in only one direction, and the tangent lines behave in a regular way. The condition $f(x_0) \cdot f''(x_0) > 0$ tells us that the initial point x_0 must be chosen on the side where the graph and its concavity are compatible.

For example, if $f'' > 0$, then the graph is concave up, and we choose x_0 where $f(x_0) > 0$. In that case, the tangent line at x_0 cuts the x-axis closer to the root, but without jumping across it in an uncontrolled way. The same idea holds when $f'' < 0$: one chooses x_0 where $f(x_0) < 0$. This makes the tangent lines move steadily toward the root.

Activity: on the geometric interpretation of Newton's method

- It consists of giving a geometric interpretation of the following result concerning Newton's method.
- This activity is intended for the students who are in the classroom. Each group must choose one representative, who will be responsible for the presentation at the blackboard. The presentation should last no more than 20 minutes.
- During the presentation, the remaining students will be assigned questions and will ask them to the student at the blackboard. If the student at the blackboard cannot answer a question, the other members of the group must answer it.
- Each correctly answered question is worth 1 point, and each incorrectly answered question is worth -1 point. If the total score of the students is at least 4 points, then all the students in the classroom will receive 1 extra point on the midterm exam.

Theorem. Let f be a function on an interval $[a, b]$ such that $f(a)f(b) < 0$. Assume that f is twice continuously differentiable on (a, b) and $f' \neq 0$, $f'' \neq 0$ on (a, b). If $x_0 \in (a, b)$ is chosen so that $f(x_0)f''(x_0) > 0$, then the sequence $\{x_n\}$ generated by Newton's method converges to a root $r \in [a, b]$ of f.

Questions

1. What does $f(a)f(b) < 0$ mean geometrically for the graph of $y = f(x)$ on the interval $[a, b]$?
2. What is the geometric meaning of the assumption $f'(x) \neq 0$ for all $x \in (a, b)$? What does this tell us about the shape of the graph and the number of roots in (a, b)?
3. What is the geometric meaning of the assumption $f''(x) \neq 0$ for all $x \in (a, b)$? What does this tell us about the concavity of the graph?
4. Newton's method is given by
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$
Explain geometrically how the point x_{n+1} is obtained from the graph of $y = f(x)$.
5. Interpret geometrically the condition $f(x_0)f''(x_0) > 0$. Why does this condition describe a "good" initial point for Newton's method?
6. Suppose the graph is increasing and concave up near the root. On which side of the root should x_0 be chosen so that $f(x_0)f''(x_0) > 0$ holds? Explain with a sketch.
7. Why do the tangent lines in Newton's method produce a sequence $\{x_n\}$ that moves toward the root instead of away from it, under the hypotheses of the theorem?
8. What can go wrong geometrically if x_0 is chosen so that $f(x_0)f''(x_0) < 0$? Describe the possible behavior of the tangent line and the next iterate.

Suggested Answers

1. The condition $f(a)f(b) < 0$ means that $f(a)$ and $f(b)$ have opposite signs. Geometrically, the graph of $y = f(x)$ is above the x-axis at one endpoint and below the x-axis at the other. Since f is continuous, the graph must cross the x-axis at least once in $[a, b]$.

2. The condition $f'(x) \neq 0$ means that the tangent to the curve is never horizontal on (a, b). Therefore f is strictly monotone on (a, b): either always increasing or always decreasing. Geometrically, the graph keeps moving in one direction and cannot turn back. Hence it can cross the x-axis at most once, so the root in (a, b) is unique.

3. The condition $f''(x) \neq 0$ means that the graph has the same type of curvature throughout the interval. Since f'' never vanishes and is continuous, it keeps one sign on (a, b). Therefore the graph is either strictly concave up on all of (a, b) or strictly concave down on all of (a, b).

4. The iterate x_{n+1} is obtained by drawing the tangent line to the graph at the point $(x_n, f(x_n))$ and then taking the x-intercept of that tangent line. So Newton's method replaces the curve locally by its tangent line and uses the zero of that line as a better approximation to the true root.

5. The condition $f(x_0)f''(x_0) > 0$ means that $f(x_0)$ and the concavity have the same sign.

   - If $f''(x_0) > 0$ (concave up), then we need $f(x_0) > 0$, so the starting point lies where the graph is above the x-axis.
   - If $f''(x_0) < 0$ (concave down), then we need $f(x_0) < 0$, so the starting point lies where the graph is below the x-axis.

   Geometrically, this places the starting point on the side from which the tangent line cuts the x-axis between x_0 and the root, making the Newton step move safely toward the root.

6. If the graph is increasing and concave up near the root, then $f'' > 0$. The condition $f(x_0)f''(x_0) > 0$ becomes $f(x_0) > 0$. Since the function is increasing, points to the right of the root have positive function values. Therefore x_0 should be chosen to the right of the root. In that case the tangent line intersects the x-axis to the left of x_0, moving the iteration toward the root.

7. Under the hypotheses, the graph is monotone and has only one concavity. When x_0 satisfies $f(x_0)f''(x_0) > 0$, the tangent line at $(x_0, f(x_0))$ intersects the x-axis on the correct side, between x_0 and the root. Repeating this construction gives points that stay on the same side of the root and move closer to it. Thus the sequence is monotone and bounded by the root, so it converges to the root.

8. If $f(x_0)f''(x_0) < 0$, the starting point is on the wrong side from the geometric point of view. Then the tangent line may intersect the x-axis farther away from the root, or even outside the relevant interval. So the next iterate may overshoot, move away from the root, or behave irregularly before eventually converging or even failing to converge in general.

19.4 Stopping conditions

In the bisection method, the error can be controlled explicitly because the root always remains inside a known interval, and the length of that interval gives a direct bound for the approximation error. In Newton's method, this is not the case. Although the method is often much faster, in principle there is no simple error formula available at each step that plays the same role as the interval length in bisection. As a result, we usually do not know the true error $|x_k - r|$ while the method is running, since the exact root r is precisely what we are trying to compute. For this reason, in practical computations one replaces the unavailable exact error by stopping conditions that can be checked from the iterates themselves. These conditions are designed to indicate when the approximations have likely become sufficiently accurate, either because two consecutive iterates are very close, or because the function value is already very small.

Definition 19.9. We have the following popular **stopping conditions**.

(1) $|x_k - x_{k-1}| < \varepsilon$ (absolute difference),

(2) $\left|\dfrac{x_k - x_{k-1}}{x_k}\right| < \varepsilon$ (relative difference) and

(3) $|f(x_k)| < \varepsilon$ (value (residual)).

None of these conditions is perfect in every situation. In fact, a small step size does not always imply that we are close to the root. Also, relative error can be misleading when the solution is close to zero. Furthermore, a small residual $|f(x_k)|$ does not automatically guarantee small error if the function is very flat near the root. In practice, one often combines several criteria and also imposes a maximum number of iterations.

Remark 19.10. One practical way to check whether an approximation $\hat{r}$ is accurate is to look at the values of the function a small distance to the left and to the right of $\hat{r}$. More precisely, given a tolerance $\varepsilon > 0$, one evaluates the function at the two points $\hat{r} - \varepsilon$ and $\hat{r} + \varepsilon$. If these two function values have opposite signs, then by the Intermediate Value Theorem there must exist at least one root in the interval $[\hat{r} - \varepsilon, \hat{r} + \varepsilon]$. Therefore, there exists a root r such that $|r - \hat{r}| < \varepsilon$. An equivalent way to describe the same idea is to say that if, when moving from $\hat{r}$ to either $\hat{r} - \varepsilon$ or $\hat{r} + \varepsilon$, one detects a change of sign, then a root must lie between those two points. Thus the root belongs either to $[\hat{r} - \varepsilon, \hat{r}]$ or to $[\hat{r}, \hat{r} + \varepsilon]$, and in both cases the distance from r to $\hat{r}$ is smaller than ε. This criterion is useful because it gives a way of certifying that the approximation is close to a root without knowing the exact root in advance. Its validity, however, depends on the continuity of the function and on the existence of a sign change in that small neighborhood of $\hat{r}$.

19.5 Speed of convergence

Suppose that we have a sequence (x_k) such that $x_k \to x_\infty$. A natural question is how fast this convergence takes place, that is, how quickly $x_k - x_\infty \to 0$. In Numerical Analysis, we usually define $E_k = x_k - x_\infty$ and call E_k the error at step k. For example, in the bisection method we have seen that

$$|E_{k+1}| \leqslant \frac{1}{2}|E_k|$$

for every $k \in \mathbb{N}$. More generally, we are interested in estimates of the form

$$|E_{k+1}| \leqslant C \cdot |E_k|,$$

for some constant $C > 0$. Let us perform a simple experiment to compare different types of error reduction.

Example 19.11. From the table below we can make several observations. First, when the error satisfies an estimate of the form $|E_{k+1}| \leqslant C\,|E_k|$, the size of the constant C plays an important role: the smaller C is, the faster the error decreases, and therefore the more reliable digits we gain at each step. For instance, the case $C = 1/5$ produces a much more significant improvement than the case $C = 1/2$. Moreover, the situation is even better when the new error depends on a power of the previous one, such as $|E_{k+1}| \leqslant |E_k|^{3/2}$ or $|E_{k+1}| \leqslant |E_k|^2$. In these cases, once the error is already small, it decreases dramatically from one step to the next. This means that the number of correct digits in the approximation increases much more rapidly than in the linear case. Therefore, depending on the relation between E_{k+1} and E_k, and especially on the constant appearing in an inequality of the form $|E_{k+1}| \leqslant C|E_k|$, the precision of our approximations may improve slowly or very quickly.

	$E_k = 10^{-4}$	E_{k+1}	E_{k+2}
$\lvert E_{k+1}\rvert \leqslant (1/2)\cdot \lvert E_k\rvert$	0.0001	0.00005	0.000025
$\lvert E_{k+1}\rvert \leqslant (1/5)\cdot \lvert E_k\rvert$	0.0001	0.00002	0.000004
$\lvert E_{k+1}\rvert \leqslant \lvert E_k\rvert^{3/2}$	0.0001	0.000001	0.000000001
$\lvert E_{k+1}\rvert \leqslant \lvert E_k\rvert^{2}$	0.0001	0.00000001	0.0000000000000001

The calculations in the table are obtained by applying each error estimate step by step, starting from the initial value $E_k = 10^{-4} = 0.0001$. For each row, we use the given relation to compute the next error E_{k+1}, and then we apply the same rule again to compute E_{k+2}. For example, if $|E_{k+1}| \leqslant (1/2)\cdot|E_k|$, then starting from $E_k = 0.0001$ we get

$$E_{k+1} \leqslant \frac{1}{2}\cdot 0.0001 = 0.00005 \quad \text{and} \quad E_{k+2} \leqslant \frac{1}{2}\cdot 0.00005 = 0.000025.$$

In view of the previous example, our goal is to identify the largest possible value of q such that $|E_{k+1}| \leqslant C\,|E_k|^q$, as higher orders q yield a significantly faster convergence. We have the following definition.

Definition 19.12. Consider a sequence $\{x_k\}$ that converges to some x_∞. We say that it converges with **order** (or **rate**) **of convergence** $q > 0$ if there exists a constant $C > 0$ such that

$$|x_\infty - x_{k+1}| \leqslant C|x_\infty - x_k|^q \qquad \text{for all } k.$$

Sometimes this is called the **Q-order** of convergence. We say that it converges with **R-order** (or **R-rate**) **of convergence** $q > 0$ if there exists a sequence $\{e_k\}$ of upper bounds for $|x_\infty - x_k|$, that is,

$$|x_\infty - x_k| \leqslant e_k \qquad \text{for all } k,$$

such that $\{e_k\}$ converges to zero with Q-order q.

At this point, it is important to distinguish between two notions of order of convergence: the *Q-order* and the *R-order*. Although they are closely related, they are not the same in general. The Q-order describes the convergence directly in terms of the actual errors $|x_\infty - x_k|$. More precisely, Q-order gives a direct comparison between two consecutive errors. By contrast, the R-order is defined in terms of a

sequence of upper bounds $\{e_k\}$ satisfying $|x_\infty - x_k| \leqslant e_k$ for all k, where the sequence $\{e_k\}$ converges to zero with Q-order q. In this case, the convergence is not measured through the exact errors themselves, but through a sequence that controls them from above. Therefore, these two notions are different in general: Q-convergence implies a precise relation for the true errors, whereas R-convergence only requires the existence of a suitable sequence of upper estimates with that behavior. For this reason, Q-order is usually a stronger notion than R-order. In Numerical Analysis, one does not always have the freedom to choose between them. Depending on the method and on the information available, sometimes one can prove a relation directly for the errors, and then Q-order is the natural notion to use. In other situations, however, the exact errors are too difficult to handle, and one can only derive convenient upper bounds; in that case, R-order is the appropriate framework. Thus, the choice between Q-order and R-order is not merely a matter of preference, but is dictated by the structure of the problem under consideration.

This brings to the following definition.

Definition 19.13. Consider a certain iterating method for finding roots of functions. We say that it is a **method of order** q, or that it has **error of order** q, where $q > 0$, if it satisfies the following condition: whenever this method produces a sequence $\{x_k\}$ converging to a certain root r of a function f, this root is simple and the function sufficiently smooth, then $\{x_k\}$ converges to r with rate q.

The assumption that the root r is simple is essential. When the root has higher multiplicity, the behavior of an iterative method may change substantially, and the expected order of convergence can deteriorate. Recall that a root r of f has multiplicity $m > 1$ if $f(r) = 0, f'(r) = 0, \ldots, f^{(m-1)}(r) = 0$, but $f^{(m)}(r) \neq 0$. In this situation, the function is much flatter near the root than in the simple-root case. As a consequence, the local information used by the iterative method becomes less effective, and the approximations usually improve more slowly as we can see in the next example.

Example 19.14. The following tables report the number of iterations required to reach the prescribed tolerance using the Newton method in several experiments carried out with Maple. In each case, the goal is to approximate one of the roots -2, 0, or 2, starting from the nearby initial guesses indicated in the header. For the polynomial $x(x-2)(x+2)$, the behavior is very regular: the number of iterations is essentially the same for all three roots, and it increases only slightly when the tolerance goes from $\varepsilon = 10^{-4}$ to $\varepsilon = 10^{-7}$. This reflects the fact that all three roots are simple, so the method keeps its expected efficiency in each case.

tolerance $\varepsilon = 10^{-4}$	-2 (closeby point -1.7)	0 (closeby point 0.7)	2 (closeby point 1.7)
$x(x-2)(x+2)$	4 iterations	4 iterations	4 iterations
$x^2(x-2)(x+2)$	5 iterations	13 iterations	5 iterations

The situation changes for the polynomial $x^2(x-2)(x+2)$. Although the roots -2 and 2 still behave well, the root at $x = 0$ now has higher multiplicity, and this creates difficulties for the method. Indeed, the number of iterations needed near 0 becomes much larger: 13 iterations for $\varepsilon = 10^{-4}$ and 23 iterations for $\varepsilon = 10^{-7}$. In other words, the factor x^2 causes a clear loss of efficiency, showing that higher multiplicity gives rise to computational problems.

For the two methods we have studied so far, we conclude this section with the following results.

Theorem 19.15. Assume that f is continuous on $[a, b]$ and satisfies $f(a) \cdot f(b) < 0$. Then the sequence $\{m_k\}$ generated by the bisection method with initial values $a_0 = a$ and $b_0 = b$ converges to a root of f. Moreover, the bisection method has linear convergence.

tolerance $\varepsilon = 10^{-7}$	-2 (closeby point -1.7)	0 (closeby point 0.7)	2 (closeby point 1.7)
$x(x-2)(x+2)$	5 iterations	5 iterations	5 iterations
$x^2(x-2)(x+2)$	5 iterations	23 iterations	5 iterations

Theorem 19.16. Newton's method has order 2 for twice continuously differentiable functions, provided the root is simple. For roots of higher multiplicity, its order drops to 1.

The difficulties that roots of higher multiplicity create for these methods will not be covered in this course. Therefore, students will not be expected to answer questions on this topic in the midterm or in the final exams.