4. Roots 2

This week we will continue our study of root finding methods. We will start with Newton's method, then move on to the secant method and finally discuss the theory behind the order of convergence of these methods.

4.1. Newton's Method

In the Bisection Method (Section 3.2) we had used only the sign of the function at the guessed points. We will now investigate how we can use also the value and the slope (derivative) of the function to get a much improved method.

4.1.1. Intuition

Root finding is really the process of finding the x-intercept of the function. If the function is complicated (e.g. highly nonlinear or does not lend itself to traditional by-hand techniques) then we can approximate the x-intercept by creating a Taylor Series approximation of the function at a nearby point and then finding the x-intercept of that simpler Taylor Series. The simplest non-trivial Taylor Series is a linear function – a tangent line!

Exercise 4.1. A tangent line approximation to a function $f(x)$ near a point x_0 is

$$y = f(x_0) + f'(x_0)\,(x - x_0)\,.$$

Set y to zero and solve for x to find the x-intercept of the tangent line.

$$x\text{-intercept of tangent line is } x = \underline{\qquad}$$

The idea of approximating the function by its tangent line gives us an algorithm for approximating the root of a function:

- Given a value of x that is a decent approximation of the root, draw a tangent line to $f(x)$ at that point.
- Find where the tangent line intersects the x axis.
- Use this intersection as the new x value and repeat.

The first step has been shown for you in Figure 4.1. The tangent line to the function $f(x)$ at the point $(x_0, f(x_0))$ is shown in red. The x-intercept of the tangent line is the new x value, x_1. The process is then repeated with x_1 as the new x_0 and so on.

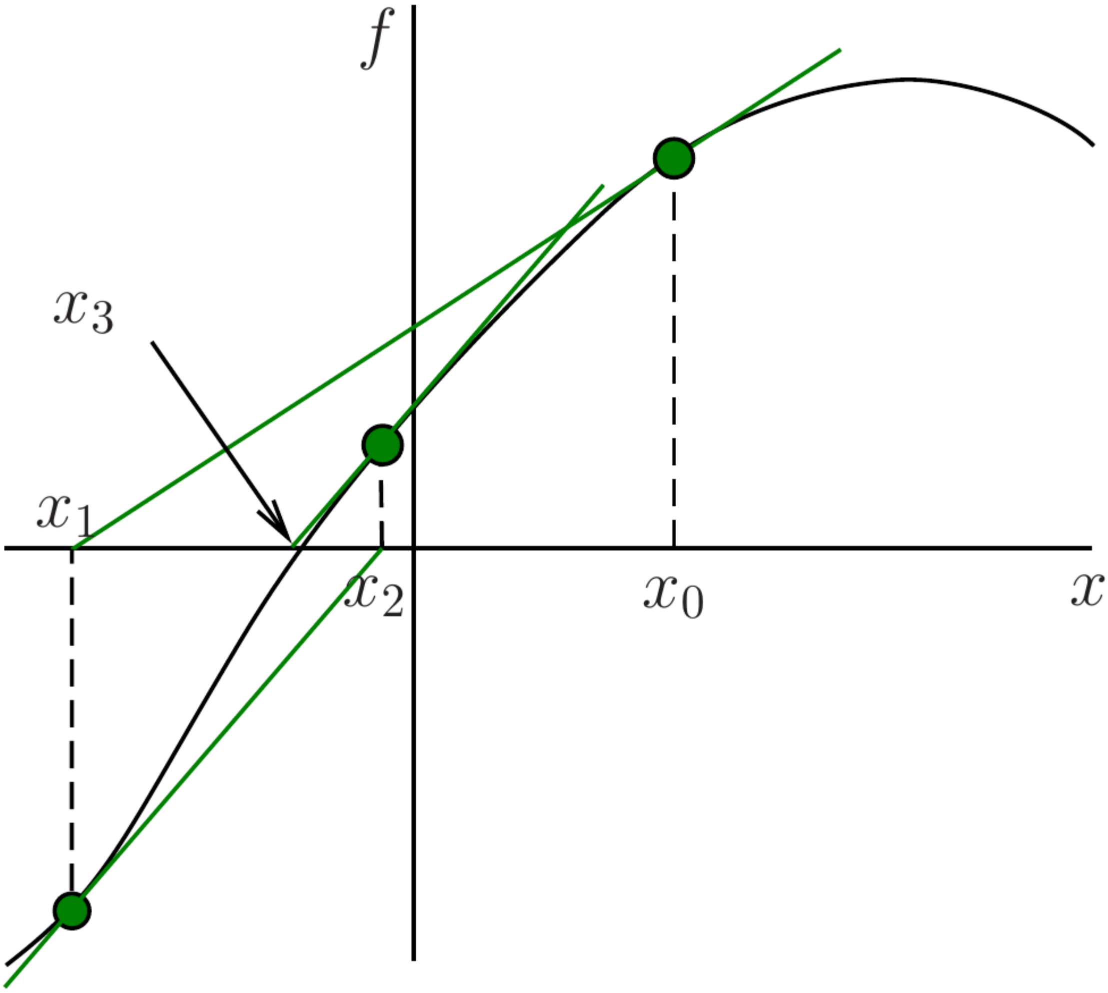

Figure 4.1.: Using successive tangent line approximations to find the root of a function

The graphical method illustrated in Figure 4.1 was introduced by Sir Isaac Newton and is therefore known as Newton's Method. Joseph Raphson then gave the algebraic formulation and popularised the method, which is therefore also known as the Newton-Raphson method.

Exercise 4.2. If we had started not at x_0 in Figure 4.1 but instead at the very end of the x-axis in that figure, what would have happened? Would this initial guess have worked to eventually approximate the root?

Exercise 4.3. Sketch some other function $f(x)$ with a root and choose an initial point x_0 and graphically perform the Newton iteration a few times, similar to Figure 4.1. Does the algorithm appear to converge to the root in your example? Do you think that this will generally take more or fewer steps than the Bisection Method?

Exercise 4.4. Consider the function $f(x) = \sin(x) + 1$. It has roots at $x = 2n\pi + 3\pi/2$ for $n \in \mathbb{Z}$. However at the roots we have $f'(x) = 0$. Make yourself a sketch to see why. What will happen when you apply Newton's method with a starting value of $x_0 = \pi$? You should be able to answer this just by looking at the sketch.

Exercise 4.5. Using your result from Exercise 4.1, write the formula for the x-intercept of the tangent line to $f(x)$ at the point $(x_n, f(x_n))$. This is the formula for the next guess x_{n+1} in Newton's Method. Newton's method is a fixed point iteration method of the form $x_{n+1} = g(x_n)$ with

$$g(x) = \dots .$$

Fill in the blank in the above formula.

Exercise 4.6. Apply Newton's method to find the root of the function $f(x) = x^2 - 2$ with an initial guess of $x_0 = 1$. Calculate the first two iterations of the sequence by hand (you do not need a calculator or computer for this). Use a calculator or computer to calculate the next two iterations and fill in the following table:

n	x_n	$f(x_n)$	$f'(x_n)$
0	$x_0 = 1$	$f(x_0) = -1$	$f'(x_0) = 2$
1	$x_1 = 1 - \frac{-1}{2} = \frac{3}{2}$	$f(x_1) =$	$f'(x_1) =$

n	x_n	$f(x_n)$	$f'(x_n)$
2	$x_2 =$	$f(x_2) =$	$f'(x_2) =$
3	$x_3 =$	$f(x_3) =$	$f'(x_3) =$
4	$x_4 =$		

4.1.2. Implementation

Exercise 4.7. The following is an outline of a Python function called `newton()` for Newton's method. The function needs to accept a Python function for $f(x)$, a Python function for $f'(x)$, an initial guess, and an optional error tolerance. Write the code.

```
def newton(f, fprime, x0, tol=1e-10):
    """
    Find root of f(x) using Newton's Method.

    Parameters:
        f (function): Function whose root we want to find
        fprime (function): Derivative of f
        x0 (float): Initial guess for the root
        tol (float, optional): Error tolerance. Defaults to 1e-10

    Returns:
        float: Approximate root of f(x)
        or error message if method fails
    """

    # Set x equal to initial guess x0

    # Loop for maximum of 30 iterations:

    #   1. Calculate next Newton iteration:
    #      x_new = x - f(x)/f'(x)

    #   2. Check if within tolerance:
    #      If |x_new - x| < tol, return x_new as root
    #      If not, update x = x_new

    # 3. If loop completes without finding root,
    #    return message that method did not converge
```

Exercise 4.8. Use your implementation from Exercise 4.7 to approximate the root of the function $f(x) = x^2 - 2$ with an initial guess of $x_0 = 1$. Use a tolerance of 10^{-10}.

4.1.3. Failures

There are several ways in which Newton's Method will behave unexpectedly – or downright fail. Some of these issues can be foreseen by examining the Newton iteration formula

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

Some of the failures that we will see are a little more surprising.

Exercise 4.9. One of the failures of Newton's Method is that it requires a division by $f'(x_n)$. If $f'(x_n)$ is zero then the algorithm completely fails. Go back to your Python function and put an `if` statement in the function that catches instances when Newton's Method fails in this way.

Exercise 4.10. An interesting failure can occur with Newton's Method that you might not initially expect. Consider the function $f(x) = x^3 - 2x + 2$. This function has a root near $x = -1.77$. Fill in the table below with pen and paper. You really do not need a computer for this. Then draw the tangent lines into Figure 4.2 for approximating the solution to $f(x) = 0$ with a starting point of $x = 0$.

n	x_n	$f(x_n)$	$f'(x_n)$
0	$x_0 = 0$	$f(x_0) = 2$	$f'(x_0) = -2$
1	$x_1 = 0 - \frac{f(x_0)}{f'(x_0)} = 1$	$f(x_1) = 1$	$f'(x_1) = 1$
2	$x_2 = 1 - \frac{f(x_1)}{f'(x_1)} =$	$f(x_2) =$	$f'(x_2) =$
3	$x_3 =$	$f(x_3) =$	$f'(x_3) =$
4	$x_4 =$	$f(x_4) =$	$f'(x_4) =$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

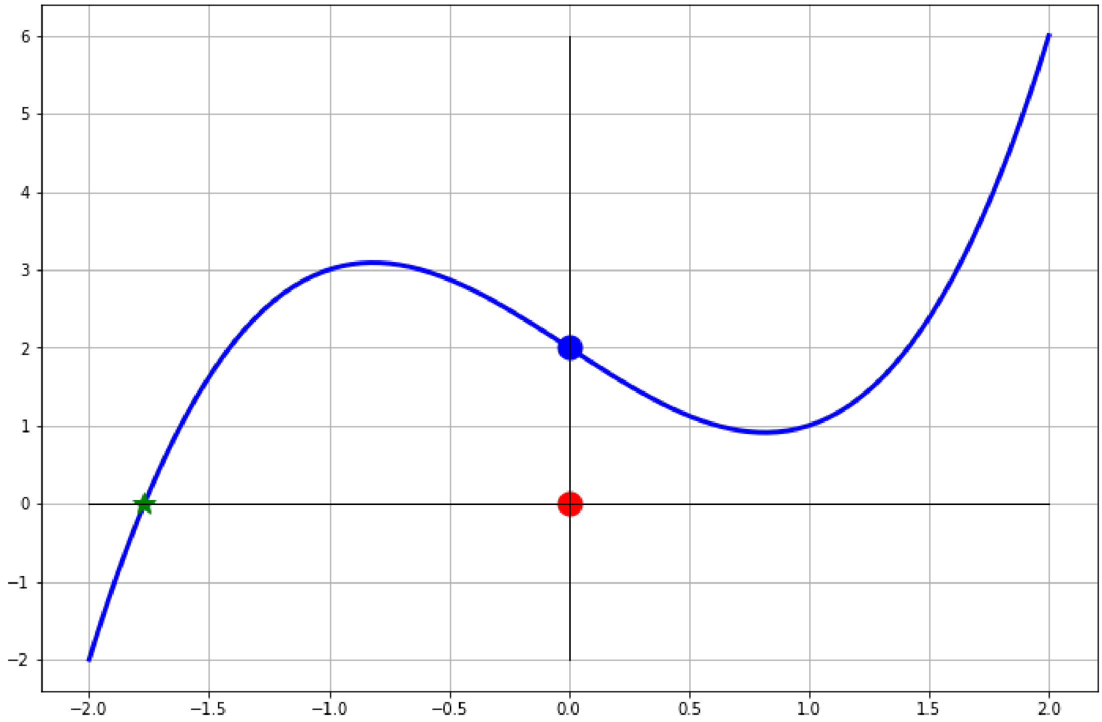

Figure 4.2.: An interesting Newton's Method failure when $f(x) = x^3 - 2x + 2$.

Exercise 4.11. Repeat the previous exercise with the function $f(x) = x^3 - 5x$ with the starting point $x_0 = -1$. Again this is easy to do with pen and paper.

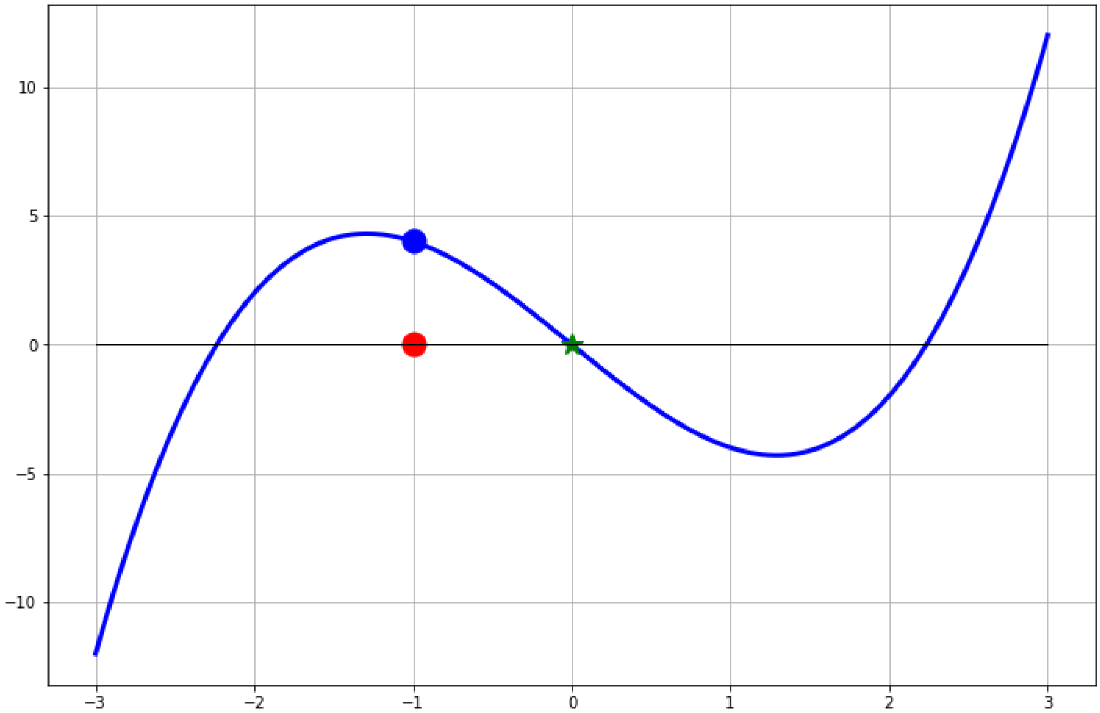

Figure 4.3.: Another surprising Newton's Method failure when $f(x) = x^3 - 5x$.

Exercise 4.12. Now let us consider the function $f(x) = \sqrt[3]{x}$. This function has a root $x = 0$. Furthermore, it is differentiable everywhere except at $x = 0$ since

$$f'(x) = \frac{1}{3}x^{-2/3} = \frac{1}{3x^{2/3}}.$$

The point of this exercise is to show what can happen when the point of non-differentiability is precisely the point that you are looking for.

1. Fill in the table of iterations starting at $x = -1$, draw the tangent lines on the plot, and make a general observation of what is happening with the Newton iterations.

n	x_n	$f(x_n)$	$f(x_n)$
0	$x_0 = -1$	$f(x_0) = -1$	$f'(x_0) =$
1	$x_1 = -1 - \frac{f(-1)}{f'(-1)} =$	$f(x_1) =$	$f'(x_1) =$
2			
3			
4			
$\vdots$	$\vdots$	$\vdots$	$\vdots$

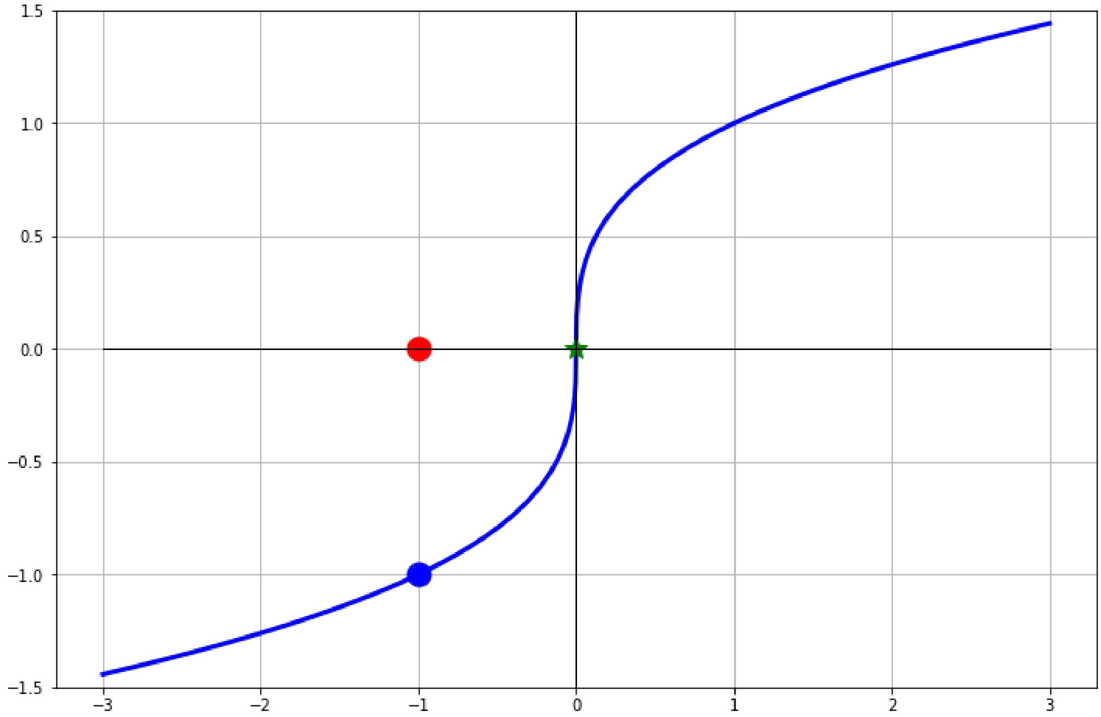

Figure 4.4.: Another surprising Newton's Method failure when $f(x) = \sqrt[3]{x}$.

2. Now let us look at the Newton iteration in a bit more detail. Since $f(x) = x^{1/3}$ and $f'(x) = \frac{1}{3}x^{-2/3}$ the Newton iteration can be simplified as

$$x_{n+1} = x_n - \frac{x_n^{1/3}}{\left(\frac{1}{3}x_n^{-2/3}\right)} = x_n - 3\frac{x_n^{1/3}}{x_n^{-2/3}} = x_n - 3x_n = -2x_n.$$

   What does this tell us about the Newton iterations?
   Hint: You should have found the exact same thing in the numerical experiment in part 1.

3. Was there anything special about the starting point $x_0 = -1$? Will this problem exist for every starting point?

Exercise 4.13. You have now seen several reasons why Newton's method could fail. Work with your group to come up with a list of reasons. Support each of your reasons with a sketch or an example.

4.1.4. Rate of Convergence

In this section we will look at the convergence rate of Newton's Method and we will show that we can greatly outperform the Bisection method.

We saw in Example 3.2 how we could empirically determine the rate of convergence of the bisection method by plotting the error in the new iterate on the y-axis and the error in the old iterate on the x axis of a log-log plot. Take a look at that example again and make sure you understand the code and the explanation of the plot. Then, to investigate the rate of convergence of Newton's method you can do the same thing.

Exercise 4.14. Write a Python function `newton_with_error_tracking()` that returns a list of absolute errors between the iterates and the exact solution. You can start with your code from `newton()` from Exercise 4.7 and add the collection of the errors in a list as in the `bisection_with_error_tracking()` function that we wrote in Example 3.1. Your new function should accept a Python function for $f(x)$, a Python function for $f'(x)$, the exact root, an initial guess, and an optional error tolerance.

Then, use this function to list the error progression for Newton's method for finding the root of $f(x) = x^2 - 2$ with the initial guess $x_0 = 1$ and tolerance 10^{-10}. You should find that the error goes down extremely quickly and that for the final iteration Python can no longer detect a difference between the new iterate and its own value for $\sqrt{2}$.

Exercise 4.15.

1. Plot the errors you calculated in Exercise 4.14 using the `plot_error_progression()` function that we wrote in Example 3.2. Note that you will first need to remove the zeros from the error list, otherwise you will get an error. You can remind yourself of how to work with lists in Section A.2.2.

2. Give a thorough explanation for how to interpret the plot that you just made.

Exercise 4.16. You will have observed that the points in your log-log plot lie approximately on a straight line, i.e.,

$$\log(\epsilon_{n+1}) \approx m \log(\epsilon_n) + b$$

Extract the slope m and intercept b of the line that you fitted to the log-log plot of the error progression using the `np.polyfit` function.

Convert this into a statement about the ratio between ϵ_{n+1} and ϵ_n,

$$\epsilon_{n+1} \approx \; ???\,(\epsilon_n)^{???}$$

Exercise 4.17. Reproduce plots like those in the previous example but for the function

$$f(x) = e^{x-3} + \sqrt{x+6} - 4$$

that you used in Exercise 3.15 with the initial guess $x_0 = 1$ and tolerance 10^{-10}. Again check that the plots have the expected shape. What is the approximate relationship between ϵ_{n+1} and ϵ_n in this case?

4.2. The Secant Method

4.2.1. Intuition and Implementation

Newton's Method has second-order (quadratic) convergence and, as such, will perform faster than the Bisection method. However, Newton's Method requires that you have a function and a derivative of that function. The conundrum here is that sometimes the derivative is cumbersome or impossible to obtain but you still want to have the great quadratic convergence exhibited by Newton's method.

Recall that Newton's method is

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

If we replace $f'(x_n)$ with an approximation of the derivative then we may have a method that is *close* to Newton's method in terms of convergence rate but is less troublesome to compute. Any method that replaces the derivative in Newton's method with an approximation is called a **Quasi-Newton Method**.

The first, and most obvious, way to approximate the derivative is just to use the slope of a secant line instead of the slope of the tangent line in the Newton iteration. If we choose two starting points that are quite close to each other then the slope of the secant line through those points will be approximately the same as the slope of the tangent line.

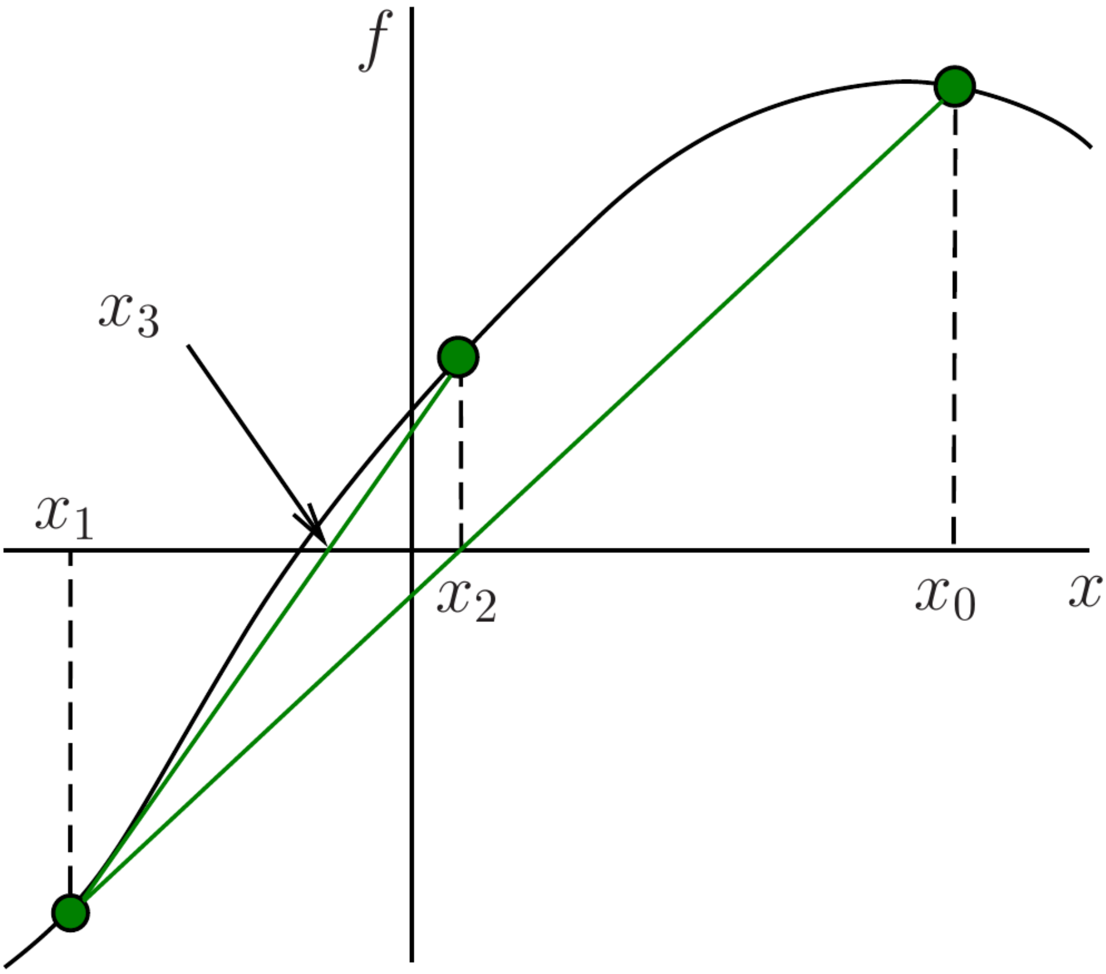

Figure 4.5.: Using successive secant line approximations to find the root of a function

---

Exercise 4.18. Use the backward difference

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

to approximate the derivative of f at x_n. Discuss why this approximates the derivative. Use this approximation of $f'(x_n)$ in the expression for x_{n+1} of Newton's method. Show that that the result simplifies to

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

Exercise 4.19. Notice that the iteration formula for x_{n+1} that you derived depends on both x_n and x_{n-1}. So to start the iteration you need to choose two points x_0 and x_1 before you can calculate $x_2, x_3, \ldots.$ Draw several pictures showing what the secant method does pictorially. Discuss whether it is important to choose these starting points close to the root and close to each other.

Exercise 4.20. Write a Python function for solving equations of the form $f(x) = 0$ with the secant method. Your function should accept a Python function, two starting points, and an optional error tolerance. Also write a test script that clearly shows that your code is working.

4.2.2. Analysis

Up to this point we have done analysis work on the Bisection Method and Newton's Method. We have found that the methods are first order and second order respectively. We end this chapter by doing the same for the Secant Method.

Exercise 4.21. Write a function `secant_with_error_tracking()` that returns a list of absolute errors between the iterates and the exact solution. You can start with your code from `secant()` from Exercise 4.20 and add the collection of the errors in a list as in the `bisection_with_error_tracking()` function from Example 3.1. Your new function should accept a Python function, the exact root, two starting points, and an optional error tolerance.

Use this function to list the error progression for the secant method for finding the root of $f(x) = x^2 - 2$ with the starting points $x_0 = 1$ and $x_1 = 3$. Use a tolerance of 10^{-10}. Use the `plot_error_progression()` function that we wrote in Example 3.2 to plot the error progression.

Exercise 4.22. Using the `np.polyfit` function, extract the slope of the line that you had fit to the log-log plot of the error progression in Exercise 4.21. What does this slope tell you about how the error at each iteration is related to the error at the previous iteration?

Exercise 4.23. Make error progression plots for a few different functions and look at the slopes. What do you notice?

4.3. Order of Convergence

You will by now have noticed that Newton's method converges much faster than the bisection method. The secant method also converges faster than the bisection method, though not quite as fast as Newton's method. You will have observed that reflected in the different slopes of the error progression graphs. In this section we summarize your observations theoretically. Thus this section is untypical in that it does not contain further explorations for you to do but instead just consolidates what you have already done. In class this will be presented as a lecture.

4.3.1. Definition

Definition 4.1. Suppose that $x_n \to p$ as $n \to \infty$ and $x_n \neq p$ for all n. The sequence $\{x_n\}$ is said to have **order of convergence** $\alpha \geq 1$ if there exists a constant $\lambda > 0$ such that

$$\lim_{n\to\infty} \frac{E_{n+1}}{E_n^\alpha} = \lambda. \tag{4.1}$$

Here E_n denotes the absolute error in the nth approximation: $E_n = |x_n - p|$.

If $\alpha = 1, 2, 3, ...$, the convergence is said to be *linear*, *quadratic*, *cubic*, ..., respectively. Note that if the convergence is linear, then the positive constant λ that appears in the above definition must be smaller than 1 ($0 < \lambda < 1$), because otherwise the sequence will not converge.

A sequence with a higher order of convergence converges much more rapidly than a sequence with a lower order of convergence. To see this, let us consider the following example:

Example 4.1. Let $\{x_n\}$ and $\{y_n\}$ be sequences converging to zero and let, for $n \geq 0$,

$$|x_{n+1}| = k|x_n| \quad \text{and} \quad |y_{n+1}| = k|y_n|^2, \tag{4.2}$$

where $0 < k < 1$. According to the definition, $\{x_n\}$ is linearly convergent and $\{y_n\}$ is quadratically convergent.

Also, we have

$$\begin{aligned} |x_n| &= k|x_{n-1}| = k^2|x_{n-2}| = ... = k^n|x_0|, \\ |y_n| &= k|y_{n-1}|^2 = k|k|y_{n-2}|^2|^2 = k^3|y_{n-2}|^4 = k^7|y_{n-3}|^8 = ... = k^{2^n-1}|y_0|^{2^n}. \end{aligned} \tag{4.3}$$

This illustrates that the quadratic convergence is much faster that the linear convergence.

We have defined the order of convergence of a converging sequence. We will also say than an iterative method has order of convergence α if the sequence of approximations that it produces on a generic problem has order of convergence α.

4.3.2. Fixed Point Iteration

Suppose that $g(x)$ satisfies the conditions of the Fixed Point Theorem on interval $[a, b]$, so that the sequence $\{x_n\}$ generated by the formula $x_{n+1} = g(x_n)$ with $x_0 \in [a, b]$ converges to a fixed point p. Then, using the Mean Value Theorem, we obtain

$$\begin{aligned} E_{n+1} &= |x_{n+1} - p| = |g(x_n) - g(p)| \\ &= |g'(\xi_n)(x_n - p)| = E_n|g'(\xi_n)|, \end{aligned} \tag{4.4}$$

where ξ_n is a number between x_n and p. This implies that if $x_n \to p$, then $\xi_n \to p$ as $n \to \infty$. Therefore,

$$\lim_{n\to\infty} \frac{E_{n+1}}{E_n} = |g'(p)|. \tag{4.5}$$

In general, $g'(p) \neq 0$, so that the fixed point iteration produces a linearly convergent sequence.

Can the fixed-point iteration produce convergent sequences with convergence of order 2, 3, etc. ? It turns out that, under certain conditions, this is possible.

We will prove the following

Theorem 4.1. *Let $m > 1$ be an integer, and let $g \in C^m[a,b]$. Suppose that $p \in [a,b]$ is a fixed point of g, and a point $x_0 \in [a,b]$ exists such that the sequence generated by the formula $x_{n+1} = g(x_n)$ converges to p. If $g'(p) = \cdots = g^{(m-1)}(p) = 0$, then $\{x_n\}$ has the order of convergence m.*

Proof. Expanding $g(x_n)$ in Taylor's series at point p, we obtain:

$$\begin{aligned} x_{n+1} = g(x_n) = g(p) + (x_n - p)g'(p) + \ldots \\ + \frac{(x_n-p)^{m-1}}{(m-1)!} g^{(m-1)}(p) \\ + \frac{(x_n-p)^m}{m!} g^{(m)}(\xi_n) \\ = p + \frac{(x_n-p)^m}{(m)!} g^{(m)}(\xi_n), \end{aligned} \tag{4.6}$$

where ξ_n is between x_n and p and, therefore, in $[a,b]$ ($x_n \in [a,b]$ at least for sufficiently large n). Then we have

$$\begin{aligned} E_{n+1} = |x_{n+1} - p| = |g(x_n) - p| = \left| \frac{(x_n-p)^m}{(m)!} g^{(m)}(\xi_n) \right| \\ = E_n^m \frac{|g^{(m)}(\xi_n)|}{m!}. \end{aligned} \tag{4.7}$$

Therefore (using the fact that $\xi_n \to p$),

$$\lim_{n\to\infty} \frac{E_{n+1}}{E_n^m} = \frac{|g^{(m)}(p)|}{m!}, \tag{4.8}$$

which means that $\{x_n\}$ has convergence of order m. □

4.3.3. Newton's Method

Newton's method for approximating the root p of the equation $f(x) = 0$ is equivalent to the fixed-point iteration $x_{n+1} = g(x_n)$ with

$$g(x) = x - \frac{f(x)}{f'(x)}. \tag{4.9}$$

Suppose that sequence $\{x_n\}$ converges to p and $f'(p) \neq 0$. We have

$$g'(x) = \frac{f(x)f''(x)}{f'(x)^2} \quad \Rightarrow \quad g'(p) = \frac{f(p)f''(p)}{f'(p)^2} = 0. \tag{4.10}$$

It follows from the above theorem that the order of convergence of Newton's method is at least 2. We also have

$$g''(x) = \frac{f'(x)f''(x) + f(x)f'''(x)}{f'(x)^2} - \frac{f(x)f''(x)^2}{f'(x)^3}$$

and hence, again using that $f(p) = 0$, we find

$$g''(p) = \frac{f''(p)}{f'(p)} \neq 0 \qquad \text{if } f''(p) \neq 0. \tag{4.11}$$

If $f''(p) = 0$, then $g''(p) = 0$ and the order of convergence is at least 3.

4.3.4. Secant Method

The situation with the secant method is more complicated (since it cannot be reduced to the fixed point iteration) and requires a separate treatment. The result is that the secant method has order of convergence $\alpha = \frac{1+\sqrt{5}}{2} \approx 1.618$.

Note that α is known as the *golden ratio*. If you are intrigued to see the golden ratio appear in this context, you can find a proof below. If you are happy to just accept the miracle, you can skip the proof.

Suppose that a sequence $\{x_n\}$, generated by the secant method

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}, \tag{4.12}$$

converges to p. Let

$$e_n = x_n - p, \tag{4.13}$$

so that $E_n = |e_n|$, and we assume that $E_n \ll 1$, which is definitely true for sufficiently large n (since the sequence $\{x_n\}$ is converging to p). Subtracting p from both sides of Eq. 4.12, we obtain

$$e_{n+1} = e_n - \frac{f(p + e_n)(e_n - e_{n-1})}{f(p + e_n) - f(p + e_{n-1})}, \tag{4.14}$$

Expanding $f(p+e_n)$ and $f(p+e_{n-1})$ in Taylor series about p and taking into account that $f(p)=0$, we find that

$$\begin{aligned} f(p+e_n) &= e_n f'(p) + \frac{e_n^2}{2} f''(p) + \cdots \\ &= e_n f'(p)(1+e_n Q) + \cdots, \\ f(p+e_{n-1}) &= e_{n-1} f'(p) + \frac{e_{n-1}^2}{2} f''(p) + \cdots \\ &= e_{n-1} f'(p)(1+e_{n-1} Q) + \cdots, \end{aligned} \tag{4.15}$$

where

$$Q = \frac{f''(p)}{2f'(p)}. \tag{4.16}$$

Substitution of Eq. 4.15 into Eq. 4.14 yields

$$\begin{aligned} e_{n+1} &= e_n - \frac{e_n(e_n - e_{n-1}) f'(p)(1+e_n Q) + \cdots}{f'(p)\left[e_n - e_{n-1} + Q(e_n^2 - e_{n-1}^2) + \cdots\right]} \\ &= e_n \left(1 - \frac{1 + e_n Q + \cdots}{1 + Q(e_n + e_{n-1}) + \cdots}\right). \end{aligned} \tag{4.17}$$

Since, for small x,

$$\frac{1}{1+x+\cdots} = 1 - x + \cdots, \tag{4.18}$$

we obtain

$$\begin{aligned} e_{n+1} &= e_n\left(1 - (1+e_n Q + \cdots)\left(1 - Q(e_n + e_{n-1}) + \cdots\right)\right) \\ &= Q e_n e_{n-1} + \cdots. \end{aligned} \tag{4.19}$$

Thus, for sufficiently large n, we have

$$e_{n+1} \approx Q e_n e_{n-1}. \tag{4.20}$$

Hence,

$$E_{n+1} \approx |Q|\, E_n E_{n-1}. \tag{4.21}$$

Now we assume that (for all sufficiently large n)

$$E_{n+1} \approx \lambda E_n^{\alpha}, \tag{4.22}$$

where λ and α are positive constants. Substituting Eq. 4.22 into Eq. 4.21, we find

$$\lambda E_n^{\alpha} \approx |Q| E_n E_{n-1} \quad \text{or} \quad \lambda E_n^{\alpha-1} \approx |Q| E_{n-1}. \tag{4.23}$$

Applying Eq. 4.22 one more time (with n replaced by $n-1$), we obtain

$$\lambda\left(\lambda E_{n-1}^{\alpha}\right)^{\alpha-1} \approx |Q| E_{n-1} \tag{4.24}$$

or, equivalently,

$$\lambda^{\alpha} E_{n-1}^{\alpha(\alpha-1)} \approx |Q| E_{n-1}. \tag{4.25}$$

The last equation will be satisfied provided that

$$\lambda^\alpha = |Q|, \quad \alpha(\alpha - 1) = 1, \tag{4.26}$$

which requires that

$$\lambda = |Q|^{1/\alpha}, \quad \alpha = (1 + \sqrt{5})/2 \approx 1.62. \tag{4.27}$$

Thus, we have shown that if $\{x_n\}$ is a convergent sequence generated by the secant method, then

$$\lim_{n\to\infty} \frac{E_{n+1}}{E_n^\alpha} = |Q|^{1/\alpha}. \tag{4.28}$$

Thus, the secant method has *superlinear* convergence.

Further reading: Section 2.4 of (Burden and Faires 2010).

4.4. Exam-Style Question

(a) Briefly explain how the Secant method modifies Newton's method. Give one advantage and one disadvantage of the Secant method compared to Newton's method. [3 marks]

(b) Define what it means for a sequence $\{x_n\}$ converging to p to have an *order of convergence* $\alpha \geq 1$. State the order of convergence for the Bisection method, the Secant method, and Newton's method. [4 marks]

(c) Consider finding the root of $f(x) = x^2 - 5$.

  i) Apply Newton's method with an initial guess $x_0 = 2$ to find the next approximation x_1. [2 marks]
  ii) Apply the Secant method with initial guesses $x_0 = 3$ and $x_1 = 2$ to find the next approximation x_2. [2 marks]

(d) Suppose a root-finding method produces a sequence $\{x_n\}$ with quadratic convergence, and the errors $E_n = |x_n - p|$ satisfy $E_{n+1} \approx E_n^2$. If the current error is $E_2 = 10^{-3}$, roughly what is the expected error E_4? [1 mark]

(e) Show that if p is a simple root of $f(x)$ (that is, $f(p) = 0$ and $f'(p) \neq 0$) and $f(x)$ is sufficiently smooth, Newton's method has an order of convergence of at least $\alpha = 2$. You may use the theorem on the order of convergence for fixed-point iteration. [4 marks]
