Secant Method

Tamas Kis | tamas.a.kis@outlook.com | https://tamaskis.github.io

Contents

1 SECANT METHOD

Newton's method is a root-finding technique that uses the derivative of a function to find its root[1]. Newton's method is defined iteratively as [2, Eq. (1)]

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

But what if we don't know $f'(x)$? Then we need to approximate it using some numerical method. Specifically, for the secant method, we use the backward approximation of a derivative, given by Eq. (2) below.

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \tag{2}$$

This approximation can be visualized using the finite difference stencil shown in Fig. 1. Substituting Eq. (2) into Eq.

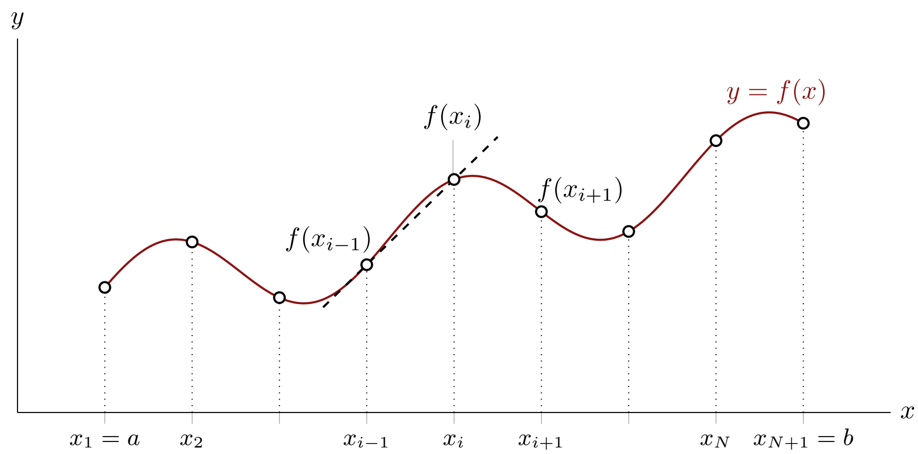

Figure 1: Backward approximation.

(1),

$$\begin{aligned} x_{i+1} &= x_i - \left[\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}\right] f(x_i) = \frac{[f(x_i) - f(x_{i-1})]\, x_i}{f(x_i) - f(x_{i-1})} - \frac{(x_i - x_{i-1}) f(x_i)}{f(x_i) - f(x_{i-1})} \\ &= \frac{x_i f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} + \frac{x_{i-1} f(x_i) - x_i f(x_i)}{f(x_i) - f(x_{i-1})} = \frac{x_i f(x_i) - x_i f(x_i) + x_{i-1} f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \end{aligned}$$

$$\boxed{x_{i+1} = \frac{x_{i-1} f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})}} \tag{3}$$

Equation (3) iteratively defines the **secant method**, which can be essentially thought of as a finite difference approximation of Newton's method for finding the root of a univariate function (based on an initial guess[2]). But how do we actually use Eq. (3)? Given an initial guess x_0, we can keep coming up with new estimates of the root. But how do we know when to stop? To resolve this issue, we define the **error**[3] as

$$\boxed{\varepsilon = |x_{i+1} - x_i|} \tag{4}$$

[1] For a discussion/MATLAB implementation of Newton's method, see [2].

[2] Often, a function $f(x)$ will have multiple roots. Therefore, the secant method typically finds the root closest to the initial guess x_0. *However*, this is not always the case; the algorithm depends heavily on the derivative of $f(x)$, which, depending on its form, may cause it to converge on a root further from x_0.

[3] Note that ε is an *approximate* error. The motivation behind using this definition of ε is that as i gets large (i.e. $i \to \infty$), $x_{i+1} - x_i$ approaches $x_{i+1} - x^*$ (*assuming* this sequence is convergent), where x^* is the true root (and therefore $x_{i+1} - x^*$ represents the *exact* error).

Once ε is small enough, we say that the estimate of the root has **converged** to the true root, within some **tolerance** (which we denote as TOL). Therefore, if we predetermine that, at most, we can *tolerate* an error of TOL, then we will keep iterating Eq. (3) until $\varepsilon < \text{TOL}$. In some cases, the error may never decrease below TOL, or take too long to decrease to below TOL. Therefore, we also define the **maximum number of iterations** ($i_{\max}$) so that the algorithm does not keep iterating forever, or for too long of a time.

In any implementation, we first have to make an initial guess x_0 for the root. Additionally, we need to set the root estimate at the second iteration (i.e. x_2) to a value slightly different than x_0 – otherwise, we will just have $x_{i+1} = x_i$ for all i and the algorithm will never "get started" (we can think of this has "kick-starting" the algorithm)[4] [1, 3, 4].

There are two basic algorithms for implementing the secant method. The first implementation, shown in Algorithm 1 below, does *not* store the result of each iteration. On the other hand, the second implementation, shown in Algorithm 2, *does* store the result of each iteration. `secant_method` implements both of these algorithms.

Since Algorithm 2 first needs to preallocate a potentially huge array to store all of the intermediate solutions, Algorithm 1 is significantly faster. Even if $i_{\max}$ (determines size of the preallocated array) is set to be a small number (for example, 10), Algorithm 1 is still faster. The reason we still consider and implement Algorithm 2 is so that convergence studies may be performed.

Algorithm 1:

Secant method (fast implementation).

Given:

- $f(x)$ - function
- x_0 - initial guess for root
- TOL - tolerance
- $i_{\max}$ - maximum number of iterations

Procedure:

1. Initialize the error so that the loop will be entered.

$$\varepsilon = (2)(\text{TOL})$$

2. Manually set the root estimates at the first and second iterations based on the initial guess.

$$x_{\text{old}} = x_0$$
$$x_{\text{int}} = 1.01 x_0$$

3. Initialize x_{new} so its scope will not be limited to within the while loop.

$$x_{\text{new}} = 0$$

4. Initialize the loop index.

$$i = 2$$

5. Find the root using the secant method.

[4] The alternative way to view this is by recalling that the derivative approximation is given by

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

If $x_i = x_{i-1}$, then the approximation to $f'(x_i)$ will become undefined, resulting in an error.

while $(\varepsilon > \text{TOL})$ **and** $(i < i_{\max})$

(a) Update root estimate.

$$x_{\text{new}} = \frac{x_{\text{old}} f(x_{\text{int}}) - x_{\text{int}} f(x_{\text{old}})}{f(x_{\text{int}}) - f(x_{\text{old}})}$$

(b) Calculate error.

$$\varepsilon = |x_{\text{new}} - x_{\text{int}}|$$

(c) Store current and previous estimates for next iteration.

$$x_{\text{old}} = x_{\text{int}}$$
$$x_{\text{int}} = x_{\text{new}}$$

(d) Increment loop index.

$$i = i + 1$$

end

Return:

- root $= x_{\text{new}}$ - converged root

Algorithm 2:

Secant method (storing intermediate root estimates).

Given:

- $f(x)$ - function
- x_0 - initial guess for root
- TOL - tolerance
- $i_{\max}$ - maximum number of iterations

Procedure:

1. Initialize the error so that the loop will be entered.

$$\varepsilon = (2)(\text{TOL})$$

2. Preallocate $\mathbf{x} \in \mathbb{R}^{i_{\max}}$ to store the estimates of the root at each iteration.
3. Manually set the root estimates at the first and second iterations based on the initial guess.

$$x_1 = x_0$$
$$x_2 = 1.01 x_0$$

4. Initialize the loop index.

$$i = 2$$

5. Find the root using the secant method.

while $(\varepsilon > \text{TOL})$ **and** $(i < i_{\max})$

(a) Update root estimate.

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

(b) Calculate error.

$$\varepsilon = |x_{i+1} - x_i|$$

(c) Increment loop index.

$$i = i + 1$$

end

Return:

- $\mathbf{x}$ - vector where the first element is the initial guess for the root, the subsequent elements are the intermediate root estimates, and the final element is the converged root

REFERENCES

[1] Richard L. Burden and J. Douglas Faires. “2.3 Newton’s Method and Its Extensions”. In: *Numerical Analysis*. 9th ed. Boston, MA: Brooks/Cole, Cengage Learning, 2011. Chap. 2.3, pp. 67–78.

[2] Tamas Kis. *Newton’s Method – MATLAB Implementation*. https://tamaskis.github.io/documentation/Newton's%20Method.pdf. 2021.

[3] *Newton’s method*. Wikipedia. https://en.wikipedia.org/wiki/Newton%27s_method (accessed: June 10, 2020).

[4] *Secant method*. Wikipedia. https://en.wikipedia.org/wiki/Secant_method (accessed: January 15, 2020).