
Newton's Method

Tamas Kis | kis@stanford.edu | <https://github.com/tamaskis>

Contents

1	Newton's Method	2
	References	4

Copyright © 2021 Tamas Kis

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



1 NEWTON'S METHOD

Newton's method is a technique used to find the root (based on an initial guess¹ x_0) of a *differentiable*, univariate function $f(x)$. The equation of the tangent line to the curve $y = f(x)$ at $x = x_0$ is

$$y = f'(x_0)(x - x_0) + f(x_0)$$

where $f'(x_0)$ is the derivative of $f(x)$ evaluated at x_0 . The x -intercept of this tangent line, $x = x_1$, can be solved by setting $y = 0$.

$$0 = f'(x_0)(x_1 - x_0) + f(x_0)$$

$$\therefore x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

x_1 is an updated estimate of the root of $f(x)$. To keep refining our estimate, we can keep iterating through this procedure using Eq. (1).

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

So how do we actually use Eq. (1)? 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**² as

$$\varepsilon = |x_{i+1} - x_i| \quad (2)$$

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. (1) 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 [1, 2].

There are two basic algorithms for implementing Newton's 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. `newtons_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:

Newton's method [fast implementation].

Given:

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

¹ Often, a function $f(x)$ will have multiple roots. Therefore, Newton's 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 .

² Note that ε is an *approximate* error. The motivation behind using this definition of ε is that as i gets large (i.e. $i \rightarrow \infty$), $x_{i+1} - x_i$ 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).

Procedure:

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

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

2. Manually set the root estimate at the first iteration based on the initial guess.

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

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

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

4. Find the root using Newton's method.

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

- (a) Update root estimate.

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

- (b) Calculate error.

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

- (c) Store the current root estimate for the next iteration.

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

- (d) Increment loop index.

$$i = i + 1$$

end

Return:

- root = x_{new} - converged root

Algorithm 2:

Newton's method [storing intermediate root estimates].

Given:

- $f(x)$ - function
- $f'(x)$ - derivative of $f(x)$
- 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 $\mathbf{x} \in \mathbb{R}^{i_{\text{max}}}$ to store the estimates of the root at each iteration.

3. Manually set the root estimate at the first iteration based on the initial guess (note that x_1 is the first element of \mathbf{x} , while x_0 is the input initial guess).

$$x_1 = x_0$$

4. Find the root using Newton's method.

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

(a) Update root estimate.

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

(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] James Hateley. *Nonlinear Equations*. MATH 3620 Course Reader (Vanderbilt University). 2019.
- [2] *Newton's method*. Wikipedia. https://en.wikipedia.org/wiki/Newton%27s_method (accessed: June 10, 2020).