Newton's Method

MATLAB Implementation

Tamas Kis | kis@stanford.edu

TAMAS KIS
https://github.com/tamaskis

Contents

1 Download and Installation

1.1 Download from MATLAB File Exchange

The `newtons_method` function is available for download on MATLAB® File Exchange at https://www.mathworks.com/matlabcentral/fileexchange/85735-newton-s-method-newtons_method.

1.2 Download from GitHub

The `newtons_method` function is available for download on GitHub at https://github.com/tamaskis/newtons_method-MATLAB.

1.3 Files Included With Download

There are **five** files included in the downloaded `zip` file:

1. `EXAMPLE.M` – *example for using the* `newtons_method` *function*
2. `LICENSE` – *license for the* `newtons_method` *function*
3. `Newton's Method - MATLAB Implementation.pdf` – *this PDF*
4. **`newtons_method.m`** – ***MATLAB function implementing Newton's method***
5. `README.md` – *markdown file for GitHub documentation*

1.4 Accessing the `newtons_method` Function in a MATLAB Script

There are **four** options for accessing the `newtons_method` function in a MATLAB script:

1. Copy the `newtons_method` function to the *end* of your MATLAB script.
2. Place the `newtons_method.m` file in the same folder as the MATLAB script.
3. Place the `newtons_method.m` file into whatever folder you want, and then use the `addpath(folderName)` command[1] where the `folderName` parameter is a string that stores the filepath of the folder that `newtons_method.m` is in *relative to* the folder that your script is in.
4. Make a toolbox by first opening `newtons_method.m`, then going to the `HOME` tab in MATLAB, and finally selecting `Package Toolbox` in the drop-down menu under `Add-Ons`. Once you package the `newtons_method` function as a toolbox, you can use it in any script.

[1] https://www.mathworks.com/help/matlab/ref/addpath.html

2 newtons_method

Calculates the root of a differentiable, univariate function using Newton's method.

Syntax

```
root = newtons_method(f,df,x0)
root = newtons_method(f,df,x0,TOL)
root = newtons_method(f,df,x0,[],imax)
root = newtons_method(f,df,x0,TOL,imax)
```

Description

`root = newtons_method(f,df,x0)` returns the root of a differentiable function $f(x)$ specified by the function handle `f`, where `df` is the derivative of $f(x)$ (i.e. $f'(x)$) and `x0` is an initial guess of the root. The default tolerance and maximum number of iterations are `TOL = 1e-12` and `imax = 1e6`, respectively.

`root = newtons_method(f,df,x0)` returns the root of a differentiable function $f(x)$ specified by the function handle `f`, where `df` is the derivative of $f(x)$ (i.e. $f'(x)$), `x0` is an initial guess of the root, and `TOL` is the tolerance. The default maximum number of iterations is `imax = 1e6`.

`root = newtons_method(f,df,x0,[],imax)` returns the root of a differential function $f(x)$ specified by the function handle `f`, where `df` is the derivative of $f(x)$ (i.e. $f'(x)$), `x0` is an initial guess of the root, and `imax` is the maximum number of iterations. The default tolerance is `TOL = 1e-12`.

`root = newtons_method(f,df,x0,TOL,imax)` returns the root of a differentiable function $f(x)$ specified by the function handle `f`, where `df` is the derivative of $f(x)$ (i.e. $f'(x)$), `x0` is an initial guess of the root, `TOL` is the tolerance, and `imax` is the maximum number of iterations.

Example

Example 2.1

Find the root(s) of $f(x) = x^2 - 1$.

■ ***SOLUTION***

To apply Newton's method to find the root(s) of $f(x)$, we first need to find $f'(x)$.

$$f'(x) = \frac{d}{dx}\left(x^2 - 1\right) = 2x$$

Defining $f(x)$ and $f'(x)$ in MATLAB,

```
% f(x) and its derivative
f = @(x) x^2-1;
df = @(x) 2*x;
```

We know $f(x) = x^2 - 1$ has roots at $x = \pm 1$, but let's pretend we don't know this, and solve this problem using a more general approach. Since $f(x)$ is a quadratic function, we know that it will have either 0 roots (in the case where $f(x)$ does not cross the x-axis) or 2 roots. Let's assume the latter case (otherwise it would be pointless to try and find roots of $f(x)$). Therefore, we use Newton's method twice, with two different guesses. Let's pick -10 and 10 as our initial guesses.

```
% finds first root of f(x)=x^2-1 using Newton's method
root1 = newtons_method(f,-10)

% finds second root of f(x)=x^2-1 using Newton's method
root2 = newtons_method(f,10)
```

This yields the result

```
root1 =

    -1


root2 =

     1
```

3 Newton's Method

Newton's method is a technique used to find the root (based on an initial guess[2] 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).

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

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**[3] as

$$\boxed{\varepsilon = |x_{i+1} - x_i|} \tag{2}$$

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 $\varepsilon < \mathrm{TOL}$. 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.

The algorithm itself is shown below in Algorithm 1 [1, 2].

[2] 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.

[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).

Algorithm 1: Newton's method.

Given: $f(x)$, $f'(x)$, x_0, TOL, $i_{\max}$

// initializes the error so the loop will be entered

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

Preallocate an $i_{\max} \times 1$ vector $\mathbf{x}$, where $\mathbf{x}$ is the vector storing the root estimate at each iteration.

// inputs initial guess for root into $\mathbf{x}$ vector

$x_1 = x_0$

// Newton's method

$i = 1$

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

  // updates estimate of root

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

  // calculates error

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

  // increments loop index

  $i = i + 1$

end

// returns root

$\text{root} = x_i$

return root

References

[1] James Hateley. *Nonlinear Equations*. MATH 3620 Course Reader (Vanderbilt University). 2019.

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