# Newton's Method

*MATLAB Implementation*

Tamas Kis | kis@stanford.edu

TAMAS KIS
https://github.com/tamaskis

# Contents

# `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)
root = newtons_method(__,'all')
```

## 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,TOL)` 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.

`root = newtons_method(__,'all')` returns a vector, where the first element of this vector is the initial guess, all intermediate elements are the intermediate estimates of the root, and the last element is the converged root. This identifier `'all'` may be appended to any of the syntaxes used above.

## Examples

**Example 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
```

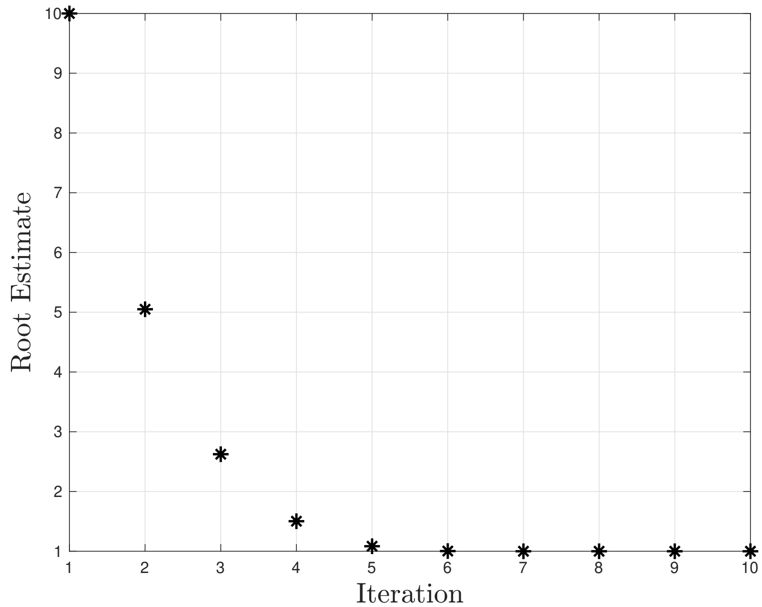
**Example 2**

Plot the intermediate root estimates vs. iteration number for finding the root of $f(x) = x^2 - 1$ with an initial guess $x_0 = 10$.

## ■ *SOLUTION*

```
% plots the intermediate root estimates for an initial guess of x0 = 10
plot(newtons_method(f,df,10,[],[],'all'),'k*','markersize',9,...
    'linewidth',1.5);
grid on;
xlabel('Iteration','interpreter','latex','fontsize',18);
ylabel('Root Estimate','interpreter','latex','fontsize',18);
```

This yields the following plot:




---

## Links

MATLAB® Central's File Exchange:
https://www.mathworks.com/matlabcentral/fileexchange/85735-newton-s-method-newtons_method

GitHub®:
https://github.com/tamaskis/newtons_method-MATLAB

# Newton's Method

**Newton's method** is a technique used to find the root (based on an initial guess[1] $x_0$) 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**[2] as

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

Once $\varepsilon$ is small enough, we say that the estimate of the root has **converged** to the true root, within some **tolerance** (which we denote as $\text{TOL}$). Therefore, if we predetermine that, at most, we can *tolerate* an error of $\text{TOL}$, then we will keep iterating Eq. (1) until $\varepsilon < \text{TOL}$. In some cases, the error may never decrease below $\text{TOL}$, or take too long to decrease to below $\text{TOL}$. 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.

---

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 $\varepsilon$ is an *approximate* error. The motivation behind using this definition of $\varepsilon$ 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*

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

*// sets root estimate at the first iteration of Newton's method as the initial guess*

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

*// Newton's method*

$i = 1$

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

  *// updates estimate of root*

  $x_{\mathrm{new}} = x_{\mathrm{old}} - \dfrac{f(x_{\mathrm{old}})}{f'(x_{\mathrm{old}})}$

  *// calculates error*

  $\varepsilon = |x_{\mathrm{new}} - x_{\mathrm{old}}|$

  *// stores updated root estimate for next iteration*

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

  *// increments loop index*

  $i = i + 1$

**end**

*// returns root*

$\mathrm{root} = x_{\mathrm{old}}$

**return** root

**Algorithm 2:** Newton's method with intermediate root estimates.

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

*// initializes the error so the loop will be entered*

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

Preallocate an $i_{\max} \times 1$ vector $\mathbf{x} = (x_1, ..., x_{i_{\max}})^T$, where $\mathbf{x}$ is the vector storing the root estimate at each iteration.

*// inputs initial guess for root into $\mathbf{x}$ vector (note that $x_1$ is the first element of $\mathbf{x}$, while $x_0$ is the input initial guess)*

$x_1 = x_0$

*// Newton's method*

$i = 1$

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

    *// updates estimate of root*

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

    *// calculates error*

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

    *// increments loop index*

    $i = i + 1$

**end**

*// stores intermediate root estimates (where last element of $\mathrm{root}$ is the converged root)*

$\mathrm{root} = (x_1, ..., x_i)^T$

*// returns root*

**return** $\mathrm{root}$

# 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).