

---

# Fixed-Point Iteration

Tamas Kis | [tamas.a.kis@outlook.com](mailto:tamas.a.kis@outlook.com) | <https://tamaskis.github.io>

---

## CONTENTS

<b>1 Fixed-Point Iteration</b>	<b>2</b>
1.1 Iterative Approaches in Engineering . . . . .	4
1.2 Root-Finding Using Fixed-Point Iteration . . . . .	4
<b>References</b>	<b>5</b>

---

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 FIXED-POINT ITERATION

Consider a univariate function  $f(x)$ . A **fixed point** of  $f(x)$ , which we denote as  $c$ , satisfies

$$f(c) = c$$

Essentially, at the fixed point, the value of the dependent variable is the same as the value of the independent variable. Another way to view fixed points are as the intersections of the curve  $y = f(x)$  with the curve  $y = x$ .

To solve for the fixed point  $c$ , we use a technique called **fixed-point iteration**. There are two basic algorithms for implementing fixed-point iteration. 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. `fixed_point_iteration` implements both of these algorithms [1].

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:

Fixed-point iteration ["fast" implementation].

### Given:

- $f(x)$  - univariate, scalar-valued function ( $f : \mathbb{R} \rightarrow \mathbb{R}$ )
- $x_0 \in \mathbb{R}$  - initial guess for fixed point
- $\text{TOL} \in \mathbb{R}$  - tolerance
- $i_{\max} \in \mathbb{Z}$  - maximum number of iterations

### Procedure:

1. Manually set the fixed point estimate at the first iteration based on the initial guess.

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

2. Initialize  $x_{\text{new}}$  so its scope will not be limited to within the while loop.

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

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

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

4. Find the fixed point using fixed-point iteration.

$$i = 1$$

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

```

    (a) Update fixed point estimate.
        
$$x_{\text{new}} = f(x_{\text{old}})$$

    (b) Calculate error.
        
$$\varepsilon = |x_{\text{new}} - x_{\text{int}}|$$

    (c) Store the current fixed point estimate for the next iteration.
        
$$x_{\text{old}} = x_{\text{new}}$$

    (d) Increment loop index.
        
$$i = i + 1$$

end

```

**Return:**

- $c = x_{\text{new}} \in \mathbb{R}$  - converged fixed point

**Algorithm 2:**

Fixed-point iteration ["return all" implementation].

**Given:**

- $f(x)$  - univariate, scalar-valued function ( $f : \mathbb{R} \rightarrow \mathbb{R}$ )
- $x_0 \in \mathbb{R}$  - initial guess for fixed point
- $\text{TOL} \in \mathbb{R}$  - tolerance
- $i_{\text{max}} \in \mathbb{Z}$  - maximum number of iterations

**Procedure:**

1. Preallocate  $\mathbf{x} \in \mathbb{R}^{i_{\text{max}}}$  to store the estimates of the fixed point at each iteration.
2. Manually set the fixed point 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).

$$x_1 = x_0$$

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

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

4. Find the fixed point using fixed-point iteration.

$$i = 1$$

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

```

      (a) Update fixed point estimate.
           $x_{i+1} = f(x_i)$ 
      (b) Calculate error.
           $\varepsilon = |x_{i+1} - x_i|$ 
      (c) Increment loop index.
           $i = i + 1$ 
      end

```

**Return:**

- $\mathbf{c} = \mathbf{x} \in \mathbb{R}^n$  - vector where the first element is the initial guess for the fixed point ( $x_0$ ), the subsequent elements are the intermediate fixed point estimates, and the final element is the converged fixed point ( $c$ )

## 1.1 Iterative Approaches in Engineering

Fixed-point iteration is especially useful for solving many problems in engineering. Consider the case where we have two unknown quantities, and two highly nonlinear equations that relate them to one another. Mathematically, we have two variables,  $x$  and  $y$ , and the following two functions:

$$y = f(x) \quad (1)$$

$$x = g(y) \quad (2)$$

In one function, you input  $x$  and get  $y$ , while in the other function, you input  $y$  and get  $x$ . Since  $f(x)$  and  $g(y)$  are nonlinear (as previously mentioned), we cannot obtain closed-form solutions for  $x$  and  $y$ .

Let's say we're primarily interested in the variable  $x$ , where the variable  $y$  mainly serves to place a constraint on  $x$ . We want to find the value  $x = c$  such that both equations are satisfied simultaneously (i.e.  $y = f(c)$  and  $c = g(y)$ ). Then we can define a new function  $h(x)$  as the function composition  $h = g \circ f$  by substituting Eq. (1) into Eq. (2).

$$h(x) = g(f(x))$$

The solution to our problem,  $x = c$ , is just the fixed point of  $h(x)$ . See Examples #4a and #4b on the "Examples" tab of [https://www.mathworks.com/matlabcentral/fileexchange/86992-fixed-point-iteration-fixed\\_point\\_iteration](https://www.mathworks.com/matlabcentral/fileexchange/86992-fixed-point-iteration-fixed_point_iteration) for an implementation of this approach to a pipe flow problem<sup>1</sup>.

## 1.2 Root-Finding Using Fixed-Point Iteration

Consider a function,  $f(x)$ . To find a root of  $f(x)$ , we need to find a value of  $x$  that satisfies

$$f(x) = 0$$

Similarly, to find a fixed-point of  $f(x)$ , we know we need to find a value of  $x$  that satisfies

$$f(x) = x$$

<sup>1</sup> This example is adapted from my personal solutions to Problem 8.96 in [2, p. 476]. However, this fluid mechanics text, in general, does not take a computational approach to such problems. Rather, it performs a "trial and error" procedure (including hand calculations and reading values off of a chart), which essentially follows the same process as fixed-point iteration – an example of this can be found in Example 8.7 [2, p. 444].

Therefore, it is simple to convert a root-finding problem to a fixed-point-finding problem.

Finding the root of  $f(x)$  is equivalent to finding the fixed-point of  $g(x)$ , where  $g(x)$  is defined as [1]

$$g(x) = x - f(x)$$

## REFERENCES

- [1] Richard L. Burden and J. Douglas Faires. “Fixed-Point Iteration”. In: *Numerical Analysis*. 9th ed. Boston, MA: Brooks/Cole, Cengage Learning, 2011. Chap. 2.2, pp. 56–66.
- [2] Bruce R. Munson et al. *Fundamentals of Fluid Mechanics*. 7<sup>th</sup>. Hoboken, NJ: John Wiley & Sons, 2013.