Bisection method vs Newton Raphson method

Dennis Mwendwa

March 2023

1 Bisection method

The bisection method is an iterative method to find the roots of a given continuous non-linear function, which assumes positive and negative values at two distinct points in its domain.

The main idea behind this root-finding method is to repeatedly bisect the interval, in which the function is continuous and assumes opposite sign values at the extremities of the interval. This process continues until we find a point xo within the interval, for which the function vanishes.

This method is based on the intermediate value theorem for continuous functions.

Algorithm for the bisection method:

- For any continuous function $f(x)$, find a closed interval $[a, b]$ such that $f(a).f(b) < 0$.
- Find the midpoint of a, b. Let $x1 = (a + b)/2$
- If $f(x1) = 0$, then $x1$ is the root.
- If $f(x1) \neq 0$, then
  - $f(a).f(x1) < 0$, root of $f(x)$ lies in $[a, x1]$, continue the above steps for interval $[a, x1]$.
  - $f(x1).f(b) < 0$, root of $f(x)$ lies in $[x1, b]$, continue the above steps for interval $[x1, b]$.
- Continue the process repeatedly until we find a point xo in $[a, b]$ for which $f(xo) = 0$.

For our equation, we have

$$f(x) = x^4–10$$

The equation was solved by Python using the following code:

```
import math
import timeit

def f(x):
    return x**4 - 10
a =2
b = 3
tolerance = 1 *(10**-6)
max_iteration = 100
for i in range(max_iteration):
    c = (a+b)/2
    print(f"Iteration number: {i}")
    if abs(f(c)) < tolerance:
        print(f"Root found at x = {c:.6f}")
        print("Time taken: ",timeit.timeit())
        break
    elif f(c)*f(a)<0:
        b=c
    else:
        a=c
```

```
  Output:
              Iteration number: 0
              Iteration number: 1
              Iteration number: 2
              Iteration number: 3
              Iteration number: 4
              Iteration number: 5
              Iteration number: 6
              Iteration number: 7
              Iteration number: 8
              Iteration number: 9
              Iteration number: 10
              Iteration number: 11
              Iteration number: 12
              Iteration number: 13
              Iteration number: 14
              Iteration number: 15
              Iteration number: 16
              Iteration number: 17
              Iteration number: 18
              Iteration number: 19
              Iteration number: 20
```

```
       Iteration number: 21
       Iteration number: 22


Root found at x = 1.778279
Time taken:  0.03730190001078881
```

2 Newton Raphson Method

The Newton Raphson Method is referred to as one of the most commonly used techniques for finding the roots of given equations. It can be efficiently generalised to find solutions to a system of equations. Moreover, we can show that when we approach the root, the method is quadratically convergent

The Newton Raphson Method Formula goes as follows:

- Suppose you need to find the root of a continuous, differentiable function $f(x)$, and you know the root you are looking for is near the point. Then Newton's method tells us that a better approximation for the root is

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

- This process may be repeated as many times as necessary to get the desired accuracy. In general, for any x-value x_n the next value is given by:

$$x_{n+1} = x_0 - \frac{f(x_0)}{f\prime(x_0)}$$

- Now, x_{n+1} is the better approximation than x_1.

For our equation, we have

$$f(x) = x^4–10$$

The equation was solved by Python using the following code:

```
import matplotlib.pyplot as plt
import numpy as np
import timeit


x = np.array(range(-20,20))
y = x**4 - 10

plt.plot(x,y)
plt.grid()
plt.show()

def f(x):
    return x**4 - 10
def df(x):
    return 4*x**3

x0 = 1
tolerance = 1e-6
max_iterr = 100

for i in range(max_iterr):
    fx= f(x0)
    dfx = df(x0)
    x1 = x0 - (fx/dfx)

    if abs(f(x1)) < tolerance:
        print(f"Root found at x = {x1:.2f}")
        print("The time taken is ",timeit.timeit())
        break
    else:
        x0 = x1
```

The output:

```
Root found at x = 1.78
The time taken is  0.03074449999257922
```

Hence this shows The Newton Raphson method is faster than Bisection method by 0.00655740001s