Bisection method

SCT211-0583/2021 Charles Daud Balila

1 Binary Search Method for Finding Roots of a Function

In numerical analysis, the binary search method (also known as the bisection method) is a root-finding algorithm that works by repeatedly bisecting an interval and then selecting a subinterval in which a root must lie, until the subinterval is sufficiently small.

Here is an implementation of the binary search method in Python, applied to the function $f(x) = x^3 - 2x - 5$ to find a root of the function between $x = 2$ and $x = 3$:

```
import math
import timeit

def f(x):
    return x**3 - 2*x - 5

a = 2
b = 3
tolerance = 1e-6
max_iteration = 100

start_time = timeit.default_timer()

for i in range(max_iteration):
    c = (a + b) / 2
    if abs(f(c)) < tolerance:
        print(f'Root found at x={c:.6f}')
        break
    elif f(c) * f(a) < 0:
        b = c
    else:
        a = c

elapsed_time = timeit.default_timer() - start_time

print(f"Time taken to execute the code: {elapsed_time:.6f} seconds")
```

In this implementation, the function[1] `f(x)` is the function whose root we want to find. The variables `a` and `b` define the interval in which we are searching for a root. The variable `tolerance` specifies the desired level of accuracy for the root, and `max_iteration` limits the maximum number of iterations that will be performed.

The main loop of the algorithm repeatedly bisects the interval `[a, b]` to get the midpoint `c`, and then checks whether `f(c)` is close enough to zero (within the specified tolerance) to declare victory. If not, the algorithm updates the