section_3

February 7, 2025

1 Section 3. Bisection method

[1]:

```python
from typing import Callable
from math import sin, log, exp

def bisection_method(
        f: Callable[[float], float],
        a : float,
        b : float,
        tolerance : float) -> float:

    if a == b and (not abs(f(a)) < tolerance):
        return None

    a, b = (a, b) if a < b else (b, a)

    guesses : list[tuple[float, float]] = []

    guess = bisect(a, b)
    while abs(f(guess)) > tolerance:
        guesses.append((guess, f(guess)))
        if f(guess) > 0:
            b = guess
        elif f(guess) < 0:
            a = guess
        guess = bisect(a, b)


    return guesses


def bisect(a : float, b : float) -> float:
    return a + ((b - a) / 2)
```

Question (i) **(i)** Find a bound for the number of iteration needed to achieve an approximation with accuracy 10^{-3} to the solution of $x^3 + x - 4 = 0$ on the interval $[1, 4]$. Find an approximation to the root with this degree of accuracy

[6]:

```python
f_of_x = lambda x : x ** 3 + x - 4
root = bisection_method(f_of_x, 1, 4, 10 ** -3)[-1][0]

print(f"Root found at {root}, valued: {f_of_x(root)}")

print("Checking we're within tolerance...")

if f_of_x(root) > 10 ** -3:
    print("We're within tolerance")
else:
    print("We're outside of tolerance")
```

```
Root found at 1.37939453125, valued: 0.004008884658105671
Checking we're within tolerance…
We're within tolerance
```

We've found a root for the function at (1.37939453125, 0.004008884658105671)