How to Find Root Values:

Newton-Raphson and Binary Methods

1 Introduction

We will discuss two methods for finding the square root of a number, specifically the square root of 2. These methods are the Newton-Raphson method and the Binary method. We will provide a mathematical explanation for each method and include graphs to illustrate the process, using the function $f(x) = x^2 - 2$ as an example.

SageMath:`https://sagecell.sagemath.org/`

2 Newton-Raphson Method

2.1 Mathematical Explanation

The Newton-Raphson method is an iterative numerical method used to find the roots of a real-valued function. Given a function $f(x)$ and its derivative $f'(x)$, the method starts with an initial guess x_0 and iteratively refines the guess using the following formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

In our case, $f(x) = x^2 - 2$, and its derivative is $f'(x) = 2x$. Thus, the iterative formula for finding the square root of 2 becomes:

$$x_{n+1} = x_n - \frac{x_n^2 - 2}{2x_n} = \frac{x_n + \frac{2}{x_n}}{2}$$

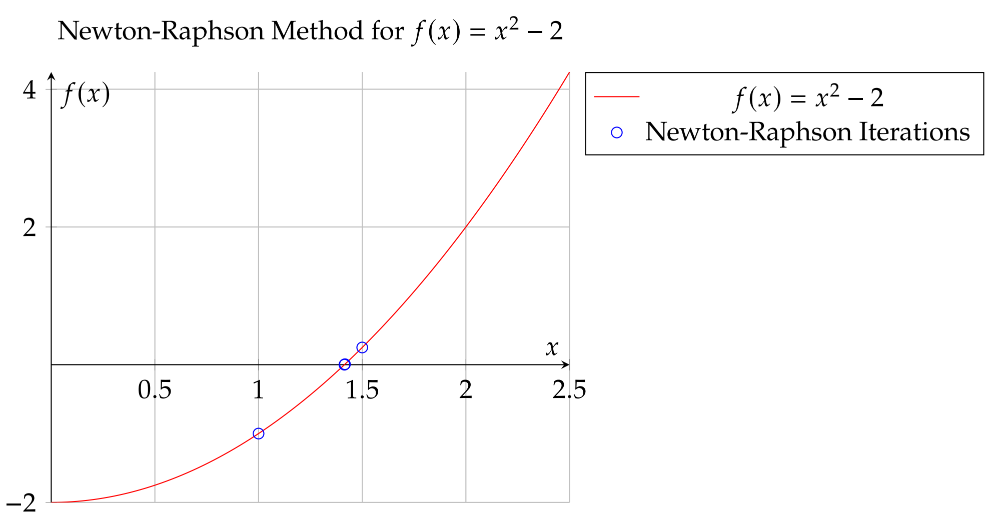

Figure 1: Newton-Raphson method applied to $f(x) = x^2 - 2$.

2.2 SAGE Code

Code 1: Newton-Raphson Method

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

def newton_sqrt(number, precision=1e-100, max_iter=100):
    if number < 0:
        raise ValueError("Cannot find the square root of a negative number.")

    x = 10  # Initial guess
    prev_x = 0

    # Lists to store the data for plotting
    x_values = [x]
    f_x_values = [x * x - number]

    iteration = 1
    for _ in range(max_iter):
        if abs(x - prev_x) <= precision:
            break

        prev_x = x
        x = (x + number / x) / 2
        x_values.append(x)
        f_x_values.append(x * x - number)

        print(f"Iteration {iteration}: x = {float(x):.50f}")
        iteration += 1

    return x, x_values, f_x_values

sqrt_2, x_values, f_x_values = newton_sqrt(2)

# Plot the graph
plt.figure()
plt.plot(x_values, f_x_values, 'o-', markersize=5)
plt.axhline(0, color='black', lw=0.5)
plt.xlabel("x")
plt.ylabel("f(x) = x^2 - 2")
plt.title("Newton-Raphson method for finding the square root of 2")
x_range = np.linspace(min(x_values), max(x_values), 1000)
y_range = x_range**2 - 2
plt.plot(x_range, y_range, color='red', label='y = x^2 - 2')
plt.legend()
plt.show()

print(f"\nSquare root of 2 using Newton-Raphson method: {float(sqrt_2):.50f}")
```

3 Binary Method

3.1 Mathematical Explanation

The Binary method, also known as the Bisection method, is another iterative numerical method to find the roots of a real-valued function. This method relies on the Intermediate Value Theorem, which states that if a continuous function takes on two values $f(a)$ and $f(b)$ with different signs, then it must take on the value 0 in the interval (a, b).

The Binary method works by narrowing down the interval containing the root. It starts with an interval $[a, b]$ such that $f(a) < 0$ and $f(b) > 0$. The midpoint of the interval is calculated as $c = \frac{a+b}{2}$. If $f(c) = 0$, then c is the root of the function. If $f(c) < 0$, then the new interval is $[c, b]$. If $f(c) > 0$, then the new interval is $[a, c]$. The process is repeated until the desired level of accuracy is achieved or the maximum number of iterations is reached.

For the function $f(x) = x^2 - 2$, we start with an interval $[1, 2]$ since $f(1) = -1$ and $f(2) = 2$. The Binary method will iteratively narrow down the interval containing the square root of 2.

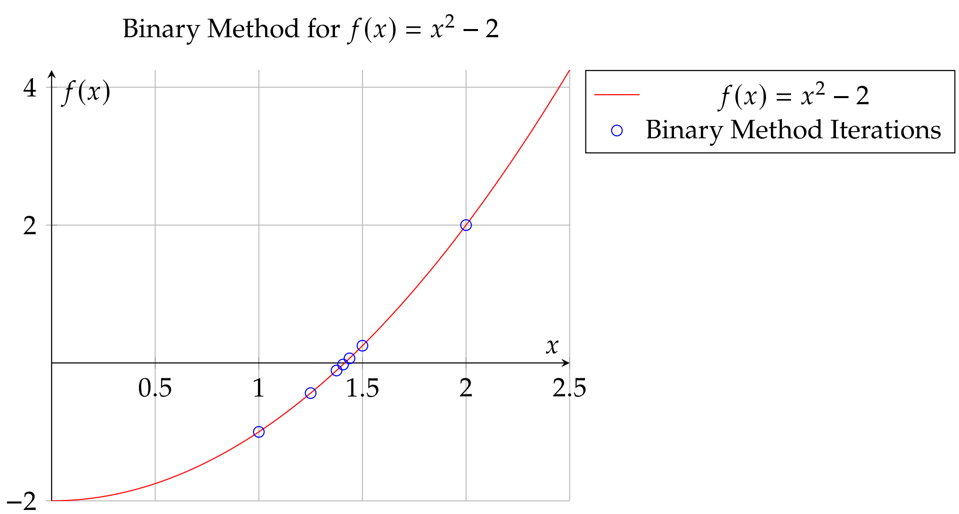

Figure 2: Binary method applied to $f(x) = x^2 - 2$.

3.2 SAGE code

Code 2: Binary Method

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

def binary_sqrt(number, precision=1e-10, max_iter=100):
    if number < 0:
        raise ValueError("Cannot find the square root of a negative number.")

    a, b = 1, 5
    x = (a + b) / 2

    # Lists to store the data for plotting
    x_values = [x]
    f_x_values = [x * x - number]

    iteration = 1
    for _ in range(max_iter):
        if abs(x * x - number) <= precision:
            break

        if x * x < number:
            a = x
        else:
            b = x

        x = (a + b) / 2
        x_values.append(x)
        f_x_values.append(x * x - number)

        print(f"Iteration {iteration}: x = {float(x):.50f}")
        iteration += 1

    return x, x_values, f_x_values

sqrt_2, x_values, f_x_values = binary_sqrt(2)

# Plot the graph
plt.figure()
plt.plot(x_values, f_x_values, 'o-', markersize=5)
plt.axhline(0, color='black', lw=0.5)
plt.xlabel("x")
plt.ylabel("f(x) = x^2 - 2")
plt.title("Binary method for finding the square root of 2")
x_range = np.linspace(min(x_values), max(x_values), 1000)
y_range = x_range**2 - 2
plt.plot(x_range, y_range, color='red', label='y = x^2 - 2')
plt.show()

print(f"\nSquare root of 2 using Binary method: {float(sqrt_2):.50f}")
```