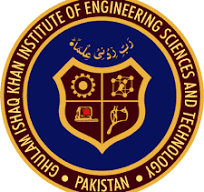
**Ghulam Ishaq Khan Institute of Engineering Sciences and Technology**

**Faculty of Computer Science and Engineering**

**CS342 – Numerical Analysis Project**

**Submitted To: Sir Aamir Shehzad**

**Dated: 4th May, 2025**

**Group Members:**

**Junaid Saleem--2022243**

**Abdul Mueed Khan--2022013**

**Muazzam Shah--2022312**

**Dua-e-Zahra Naqvi--2022151**

**Neha Abdul Rahim--2022481**

Table of Contents

[1. Bisection Method 3](#_Toc197179260)

[1.1. Method Overview 3](#_Toc197179261)

[1.2. Explanation of the Code 3](#_Toc197179262)

[1.3. Tables of Results 4](#_Toc197179263)

[1.4. Graphs 6](#_Toc197179264)

[1.5. Interpretation of Results 7](#_Toc197179265)

[Function 1: (x) = x3 - 4x + 1 on [0, 1] 8](#_Toc197179266)

[Function 2: f(x)= on [0, 10] 8](#_Toc197179267)

[Function 3: f(x)=cos(x)−x on[0, 1] 8](#_Toc197179268)

[Conclusion 8](#_Toc197179269)

[2. Fixed Point Iteration 8](#_Toc197179270)

[2.1. Method Overview 8](#_Toc197179271)

[2.2. Explanation of Code 8](#_Toc197179272)

[2.3. Table of Results 9](#_Toc197179273)

[2.4. Graphs 11](#_Toc197179274)

[2.5. Interpretation of Results 12](#_Toc197179275)

[Function 1: g(x)=cos(x), x0=0 13](#_Toc197179276)

[Function 2: g(x)= , x0=0 13](#_Toc197179277)

[Function 3: g(x)= , x0=2 13](#_Toc197179278)

[Conclusion 13](#_Toc197179279)

# Bisection Method

## ****Method Overview****

The bisection method is a root-finding algorithm that:

* Requires a continuous function that changes sign over an interval [a, b]
* Repeatedly bisects the interval and selects the subinterval containing the root
* Guarantees convergence to a root if the function is continuous
* Uses a tolerance of 1e-6 and maximum 100 iterations as stopping criteria
* Calculates error as the difference between consecutive approximations

## Explanation of the Code

The implementation consists of three main functions:

**bisectionMethod:**

* Takes function, interval bounds [a, b], tolerance, and max iterations
* Returns root approximation, iteration count, and iteration data
* Checks for sign change at interval endpoints
* Updates interval based on function value at midpoint

**printResults:**

* Shows current interval bounds [a, b] and their function values
* Displays midpoint (x) and its function value
* Calculates error between consecutive approximations
* Formats output in a clear tabular form

**plotResults:**

* Error vs Iterations plot showing convergence
* Function curve plot with root location

## Tables of Results

**Stopping Tolerance: 10-6**

**f(x) = x3- 4x + 1**

**Interval: [0,1]**

| **Iteration** | **a** | **f(a)** | **b** | **f(b)** | **x** | **f(x)** | **Error** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.000000 | 1.000000 | 1.000000 | -2.000000 | 0.500000 | -0.875000 | inf |
| 2 | 0.000000 | 1.000000 | 0.500000 | -0.875000 | 0.250000 | 0.015625 | 0.250000 |
| 3 | 0.250000 | 0.015625 | 0.500000 | -0.875000 | 0.375000 | -0.447266 | 0.125000 |
| 4 | 0.250000 | 0.015625 | 0.375000 | -0.447266 | 0.312500 | -0.219482 | 0.062500 |
| 5 | 0.250000 | 0.015625 | 0.312500 | -0.219482 | 0.281250 | -0.102753 | 0.031250 |
| 6 | 0.250000 | 0.015625 | 0.281250 | -0.102753 | 0.265625 | -0.043758 | 0.015625 |
| 7 | 0.250000 | 0.015625 | 0.265625 | -0.043758 | 0.257812 | -0.014114 | 0.007812 |
| 8 | 0.250000 | 0.015625 | 0.257812 | -0.014114 | 0.253906 | 0.000744 | 0.003906 |
| 9 | 0.253906 | 0.000744 | 0.257812 | -0.014114 | 0.255859 | -0.006688 | 0.001953 |
| 10 | 0.253906 | 0.000744 | 0.255859 | -0.006688 | 0.254883 | -0.002973 | 0.000977 |
| 11 | 0.253906 | 0.000744 | 0.254883 | -0.002973 | 0.254395 | -0.001115 | 0.000488 |
| 12 | 0.253906 | 0.000744 | 0.254395 | -0.001115 | 0.254150 | -0.000185 | 0.000244 |
| 13 | 0.253906 | 0.000744 | 0.254150 | -0.000185 | 0.254028 | 0.000279 | 0.000122 |
| 14 | 0.254028 | 0.000279 | 0.254150 | -0.000185 | 0.254089 | 0.000047 | 0.000061 |
| 15 | 0.254089 | 0.000047 | 0.254150 | -0.000185 | 0.254120 | -0.000069 | 0.000031 |
| 16 | 0.254089 | 0.000047 | 0.254120 | -0.000069 | 0.254105 | -0.000011 | 0.000015 |
| 17 | 0.254089 | 0.000047 | 0.254105 | -0.000011 | 0.254097 | 0.000018 | 0.000008 |
| 18 | 0.254097 | 0.000018 | 0.254105 | -0.000011 | 0.254101 | 0.000003 | 0.000004 |
| 19 | 0.254101 | 0.000003 | 0.254105 | -0.000011 | 0.254103 | -0.000004 | 0.000002 |
| 20 | 0.254101 | 0.000003 | 0.254103 | -0.000004 | 0.254102 | -0.000000 | 0.000001 |

**f(x) =**

**Interval: [0,10]**

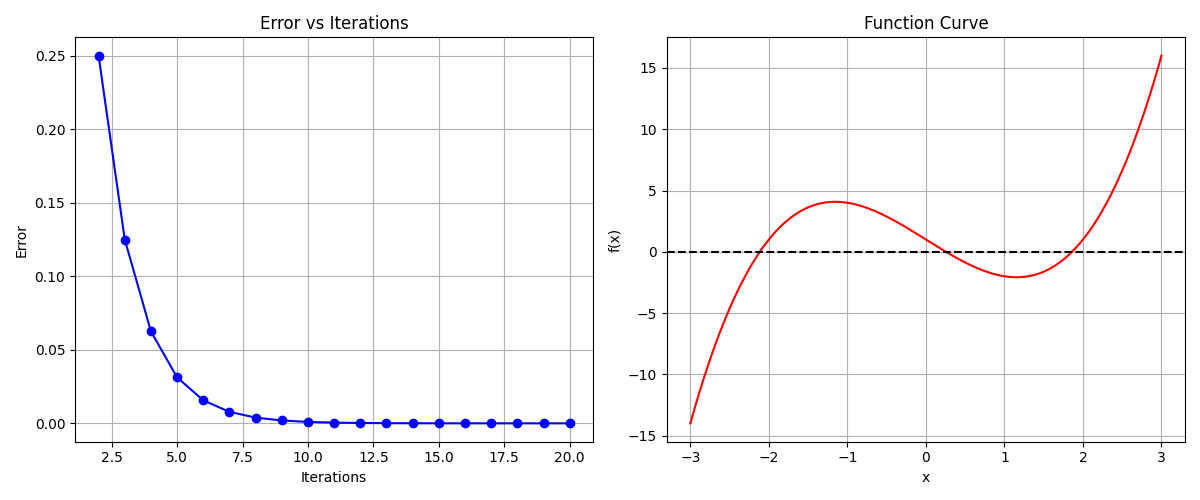
| **Iteration** | **a** | **f(a)** | **b** | **f(b)** | **x** | **f(x)** | **Error** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.000000 | -2.000000 | 10.000000 | 1.162278 | 5.000000 | 0.236068 | inf |
| 2 | 0.000000 | -2.000000 | 5.000000 | 0.236068 | 2.500000 | -0.418861 | 2.500000 |
| 3 | 2.500000 | -0.418861 | 5.000000 | 0.236068 | 3.750000 | -0.063508 | 1.250000 |
| 4 | 3.750000 | -0.063508 | 5.000000 | 0.236068 | 4.375000 | 0.091650 | 0.625000 |
| 5 | 3.750000 | -0.063508 | 4.375000 | 0.091650 | 4.062500 | 0.015564 | 0.312500 |
| 6 | 3.750000 | -0.063508 | 4.062500 | 0.015564 | 3.906250 | -0.023576 | 0.156250 |
| 7 | 3.906250 | -0.023576 | 4.062500 | 0.015564 | 3.984375 | -0.003910 | 0.078125 |
| 8 | 3.984375 | -0.003910 | 4.062500 | 0.015564 | 4.023438 | 0.005851 | 0.039062 |
| 9 | 3.984375 | -0.003910 | 4.023438 | 0.005851 | 4.003906 | 0.000976 | 0.019531 |
| 10 | 3.984375 | -0.003910 | 4.003906 | 0.000976 | 3.994141 | -0.001465 | 0.009766 |
| 11 | 3.994141 | -0.001465 | 4.003906 | 0.000976 | 3.999023 | -0.000244 | 0.004883 |
| 12 | 3.999023 | -0.000244 | 4.003906 | 0.000976 | 4.001465 | 0.000366 | 0.002441 |
| 13 | 3.999023 | -0.000244 | 4.001465 | 0.000366 | 4.000244 | 0.000061 | 0.001221 |
| 14 | 3.999023 | -0.000244 | 4.000244 | 0.000061 | 3.999634 | -0.000092 | 0.000610 |
| 15 | 3.999634 | -0.000092 | 4.000244 | 0.000061 | 3.999939 | -0.000015 | 0.000305 |
| 16 | 3.999939 | -0.000015 | 4.000244 | 0.000061 | 4.000092 | 0.000023 | 0.000153 |
| 17 | 3.999939 | -0.000015 | 4.000092 | 0.000023 | 4.000015 | 0.000004 | 0.000076 |
| 18 | 3.999939 | -0.000015 | 4.000015 | 0.000004 | 3.999977 | -0.000006 | 0.000038 |
| 19 | 3.999977 | -0.000006 | 4.000015 | 0.000004 | 3.999996 | -0.000001 | 0.000019 |
| 20 | 3.999996 | -0.000001 | 4.000015 | 0.000004 | 4.000006 | 0.000001 | 0.000010 |
| 21 | 3.999996 | -0.000001 | 4.000006 | 0.000001 | 4.000001 | 0.000000 | 0.000005 |
| 22 | 3.999996 | -0.000001 | 4.000001 | 0.000000 | 3.999999 | -0.000000 | 0.000002 |
| 23 | 3.999999 | -0.000000 | 4.000001 | 0.000000 | 4.000000 | -0.000000 | 0.000001 |
| 24 | 4.000000 | -0.000000 | 4.000001 | 0.000000 | 4.000000 | 0.000000 | 0.000001 |

**f(x) = cos(x) - x**

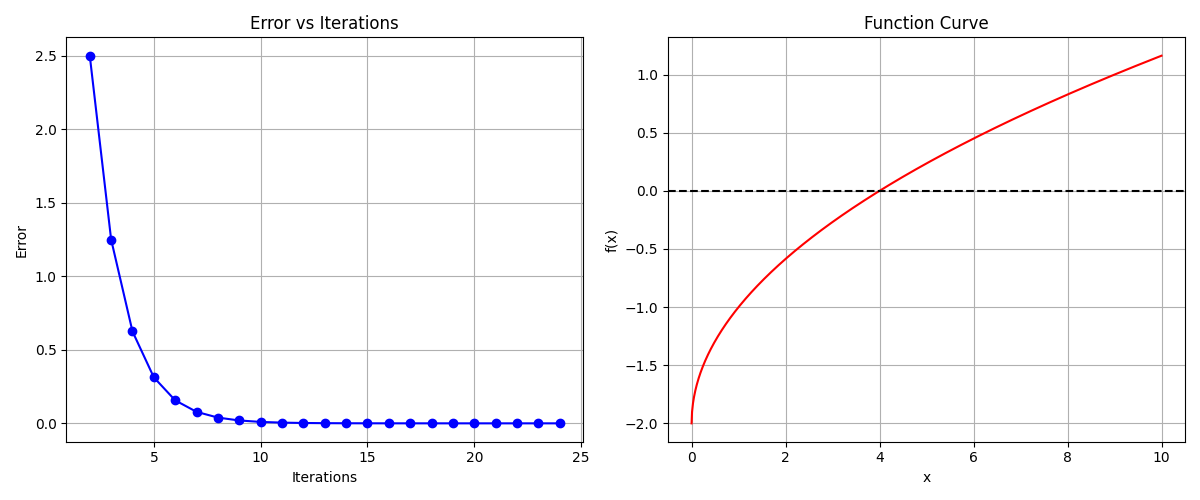
**Interval: [0,1]**

| **Iteration** | **a** | **f(a)** | **b** | **f(b)** | **x** | **f(x)** | **Error** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0.000000 | 1.000000 | 1.000000 | -0.459698 | 0.500000 | 0.377583 | inf |
| 2 | 0.500000 | 0.377583 | 1.000000 | -0.459698 | 0.750000 | -0.018311 | 0.250000 |
| 3 | 0.500000 | 0.377583 | 0.750000 | -0.018311 | 0.625000 | 0.185963 | 0.125000 |
| 4 | 0.625000 | 0.185963 | 0.750000 | -0.018311 | 0.687500 | 0.085335 | 0.062500 |
| 5 | 0.687500 | 0.085335 | 0.750000 | -0.018311 | 0.718750 | 0.033879 | 0.031250 |
| 6 | 0.718750 | 0.033879 | 0.750000 | -0.018311 | 0.734375 | 0.007875 | 0.015625 |
| 7 | 0.734375 | 0.007875 | 0.750000 | -0.018311 | 0.742188 | -0.005196 | 0.007812 |
| 8 | 0.734375 | 0.007875 | 0.742188 | -0.005196 | 0.738281 | 0.001345 | 0.003906 |
| 9 | 0.738281 | 0.001345 | 0.742188 | -0.005196 | 0.740234 | -0.001924 | 0.001953 |
| 10 | 0.738281 | 0.001345 | 0.740234 | -0.001924 | 0.739258 | -0.000289 | 0.000977 |
| 11 | 0.738281 | 0.001345 | 0.739258 | -0.000289 | 0.738770 | 0.000528 | 0.000488 |
| 12 | 0.738770 | 0.000528 | 0.739258 | -0.000289 | 0.739014 | 0.000120 | 0.000244 |
| 13 | 0.739014 | 0.000120 | 0.739258 | -0.000289 | 0.739136 | -0.000085 | 0.000122 |
| 14 | 0.739014 | 0.000120 | 0.739136 | -0.000085 | 0.739075 | 0.000017 | 0.000061 |
| 15 | 0.739075 | 0.000017 | 0.739136 | -0.000085 | 0.739105 | -0.000034 | 0.000031 |
| 16 | 0.739075 | 0.000017 | 0.739105 | -0.000034 | 0.739090 | -0.000008 | 0.000015 |
| 17 | 0.739075 | 0.000017 | 0.739090 | -0.000008 | 0.739082 | 0.000005 | 0.000008 |
| 18 | 0.739082 | 0.000005 | 0.739090 | -0.000008 | 0.739086 | -0.000002 | 0.000004 |
| 19 | 0.739082 | 0.000005 | 0.739086 | -0.000002 | 0.739084 | 0.000001 | 0.000002 |
| 20 | 0.739084 | 0.000001 | 0.739086 | -0.000002 | 0.739085 | -0.000000 | 0.000001 |

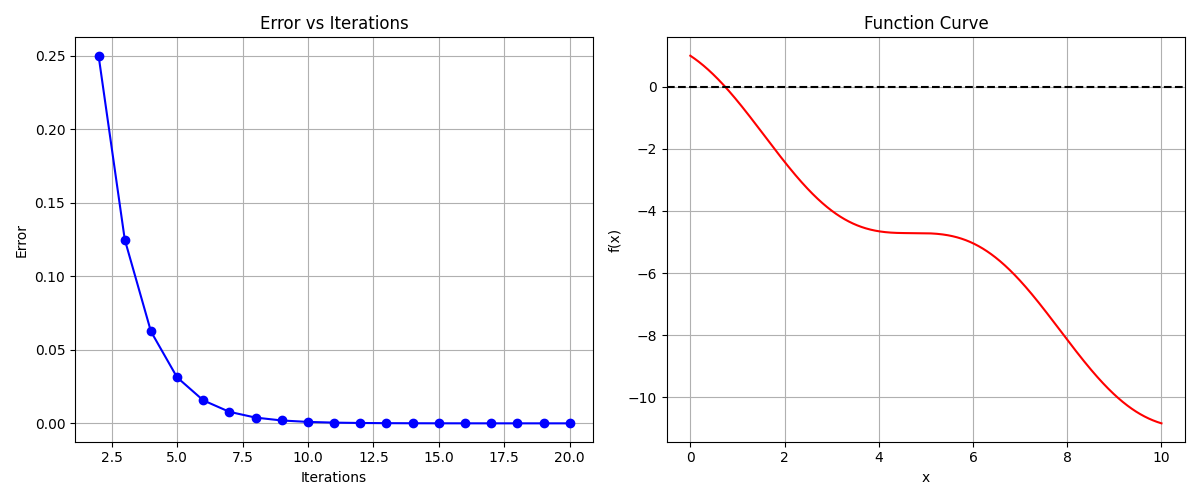
## Graphs

**f(x) = x3- 4x + 1**

**f(x) =**

****

**f(x) = cos(x) - x**

****

## ****Interpretation of Results****

The bisection method was applied to three functions, each with a known sign change in the given interval. In all cases, the method successfully converged to a root within the specified tolerance of 10−610^{-6}.

### ****Function 1:**** (x) = x3 - 4x + 1 ****on**** [0, 1]

* **Root found:** ≈ 0.254102
* **Iterations:** 20
* **Remarks:** Function changes sign over [0,1]; convergence was steady and accurate.

### ****Function 2:**** f(x)= ****on**** [0, 10]

* **Root found:** ≈ 4.000000
* **Iterations:** 24
* **Remarks:** Root at x=4x = 4 was accurately detected; error halved each step.

### ****Function 3:**** f(x)=cos(x)−x on[0, 1]

* **Root found:** ≈ 0.739085
* **Iterations:** 20
* **Remarks:** Classic fixed-point problem; bisection method converged reliably.

### ****Conclusion****

The method showed consistent convergence in all cases, with clear reduction in error and root approximation through midpoint updates. Tabulated results confirm the method's precision and stability.

# Fixed Point Iteration

## Method Overview

The Fixed Point Iteration method is a numerical technique used to find fixed points of a function, where x = g(x). A fixed point is a value that remains unchanged when the function is applied to it. This implementation provides a robust solution with visualization and detailed iteration tracking.

## Explanation of Code

**Main Function: fixedPointIteration()**

**Input Parameters:**

* g(x): The iteration function
* x0: Initial guess
* tolerance: Convergence criterion (default: 1e-6)
* maxIterations: Maximum allowed iterations (default: 100)
* Returns: (fixed point, iterations count, iteration data)

**Visualization: plotResults()**

Generates two plots:

* Error convergence over iterations
* Iteration function g(x) with y=x line intersection

**Results Display: printResults()**

* Presents iteration data in a formatted table
* Shows convergence progress and final results

## Table of Results

**Tolerance Limit = 10-6**

**g(x)=cos(x)**

**Initial guess: x0=0**

| **Iteration** | **xₙ** | **g(xₙ)** | **Error** |
| --- | --- | --- | --- |
| 1 | 1.000000 | 0.540302 | 1.000000 |
| 2 | 0.540302 | 0.857553 | 0.459698 |
| 3 | 0.857553 | 0.654290 | 0.317251 |
| 4 | 0.654290 | 0.793480 | 0.203263 |
| 5 | 0.793480 | 0.701369 | 0.139191 |
| 6 | 0.701369 | 0.763960 | 0.092112 |
| 7 | 0.763960 | 0.722102 | 0.062591 |
| 8 | 0.722102 | 0.750418 | 0.041857 |
| 9 | 0.750418 | 0.731404 | 0.028315 |
| 10 | 0.731404 | 0.744237 | 0.019014 |
| 11 | 0.744237 | 0.735605 | 0.012833 |
| 12 | 0.735605 | 0.741425 | 0.008633 |
| 13 | 0.741425 | 0.737507 | 0.005820 |
| 14 | 0.737507 | 0.740147 | 0.003918 |
| 15 | 0.740147 | 0.738369 | 0.002640 |
| 16 | 0.738369 | 0.739567 | 0.001778 |
| 17 | 0.739567 | 0.738760 | 0.001198 |
| 18 | 0.738760 | 0.739304 | 0.000807 |
| 19 | 0.739304 | 0.738938 | 0.000544 |
| 20 | 0.738938 | 0.739184 | 0.000366 |
| 21 | 0.739184 | 0.739018 | 0.000247 |
| 22 | 0.739018 | 0.739130 | 0.000166 |
| 23 | 0.739130 | 0.739055 | 0.000112 |
| 24 | 0.739055 | 0.739106 | 0.000075 |
| 25 | 0.739106 | 0.739071 | 0.000051 |
| 26 | 0.739071 | 0.739094 | 0.000034 |
| 27 | 0.739094 | 0.739079 | 0.000023 |
| 28 | 0.739079 | 0.739089 | 0.000016 |
| 29 | 0.739089 | 0.739082 | 0.000010 |
| 30 | 0.739082 | 0.739087 | 0.000007 |
| 31 | 0.739087 | 0.739084 | 0.000005 |
| 32 | 0.739084 | 0.739086 | 0.000003 |
| 33 | 0.739086 | 0.739085 | 0.000002 |
| 34 | 0.739085 | 0.739086 | 0.000001 |
| 35 | 0.739086 | 0.739085 | 0.000001 |

**g(x)=**

**Initial guess: x0=0**

| **Iteration** | **xₙ** | **g(xₙ)** | **Error** |
| --- | --- | --- | --- |
| 1 | 0.666667 | 0.888889 | 0.666667 |
| 2 | 0.888889 | 0.962963 | 0.222222 |
| 3 | 0.962963 | 0.987654 | 0.074074 |
| 4 | 0.987654 | 0.995885 | 0.024691 |
| 5 | 0.995885 | 0.998628 | 0.008230 |
| 6 | 0.998628 | 0.999543 | 0.002743 |
| 7 | 0.999543 | 0.999848 | 0.000914 |
| 8 | 0.999848 | 0.999949 | 0.000305 |
| 9 | 0.999949 | 0.999983 | 0.000102 |
| 10 | 0.999983 | 0.999994 | 0.000034 |
| 11 | 0.999994 | 0.999998 | 0.000011 |
| 12 | 0.999998 | 0.999999 | 0.000004 |
| 13 | 0.999999 | 1.000000 | 0.000001 |

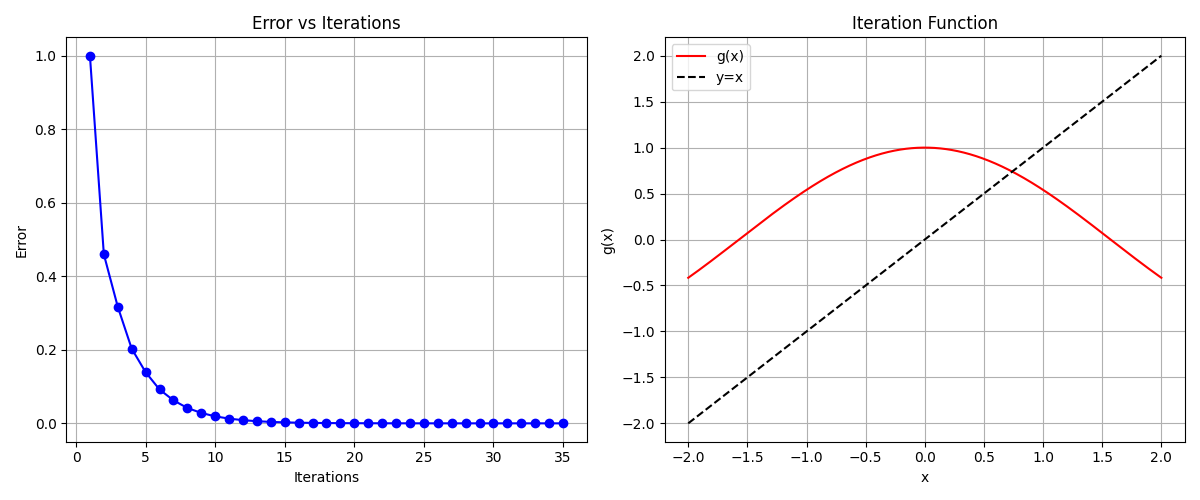
**g(x)=**

**Initial guess: x0=2**

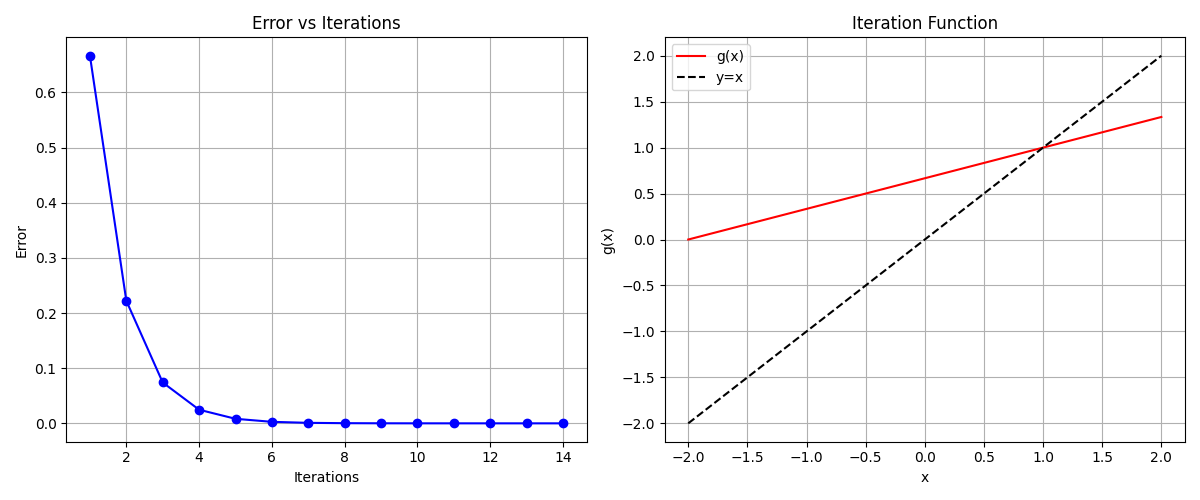
| **Iteration** | **xₙ** | **g(xₙ)** | **Error** |
| --- | --- | --- | --- |
| 1 | 1.250000 | 1.025000 | 0.750000 |
| 2 | 1.025000 | 1.000305 | 0.225000 |
| 3 | 1.000305 | 1.000000 | 0.024695 |
| 4 | 1.000000 | 1.000000 | 0.000305 |
| 5 | 1.000000 | 1.000000 | 0.000000 |

## Graphs

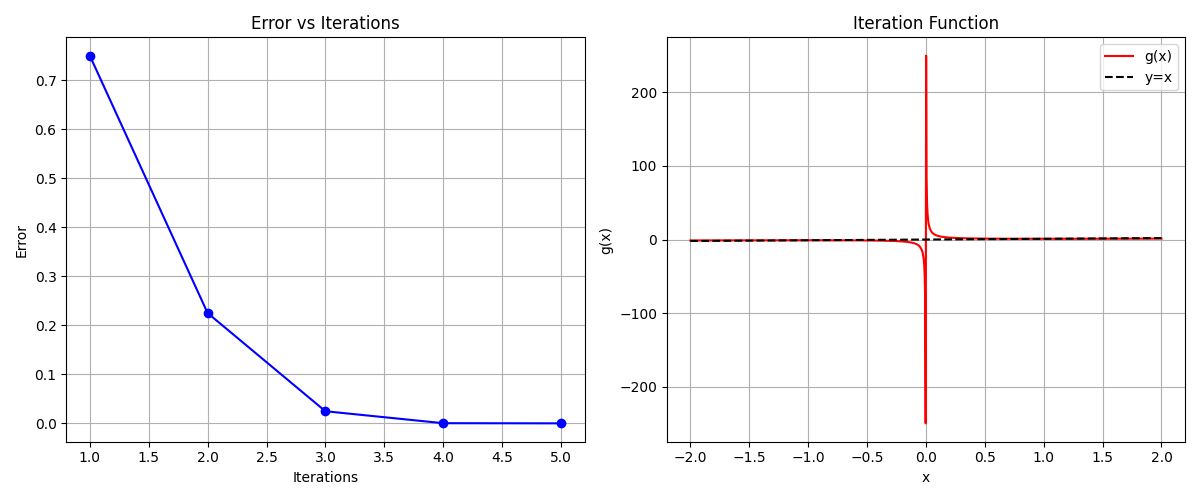
**g(x)=cos(x)**



**g(x)=**



**g(x)=**



## ****Interpretation of Results****

The Fixed Point Iteration method was applied to three functions using an initial guess and a convergence tolerance of 10−610^{-6}. Results are as follows:

### ****Function 1:**** g(x)=cos(x), x0=0

* **Fixed Point:** ≈ 0.739085
* **Iterations:** 35
* **Remarks:** Convergence was gradual due to the nature of cosine near the root, showing consistent error reduction.

### ****Function 2:**** g(x)= , x0=0

* **Fixed Point:** ≈ 1.000000
* **Iterations:** 13
* **Remarks:** Smooth and steady convergence with fewer iterations, demonstrating strong contractive behavior.

### ****Function 3:**** g(x)= , x0=2

* **Fixed Point:** ≈ 1.000000
* **Iterations:** 5
* **Remarks:** Fast convergence due to the function’s rapid correction of the initial guess.

### ****Conclusion****

The method successfully converged in all cases, with the speed of convergence dependent on the nature and contractiveness of the function. Proper choice of g(x)g(x) and initial guess significantly impacts performance.