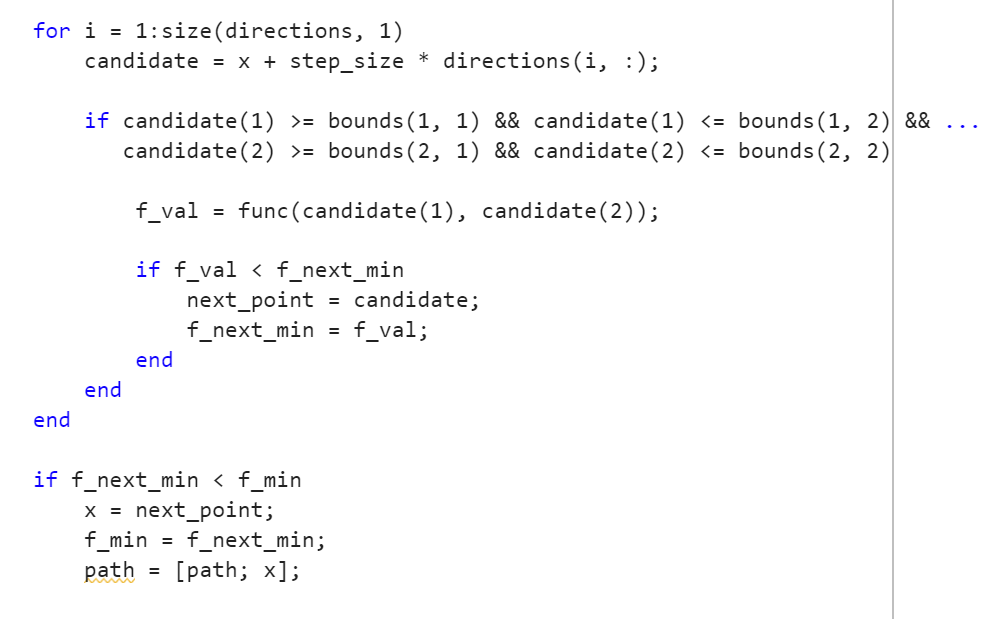
**Problem1A:**

**Main ideas:**

The pattern search algorithm starts from an initial point, exploring in four directions to find a lower function value; if a better point is found, it moves there, otherwise, the step size is reduced until it meets the tolerance.

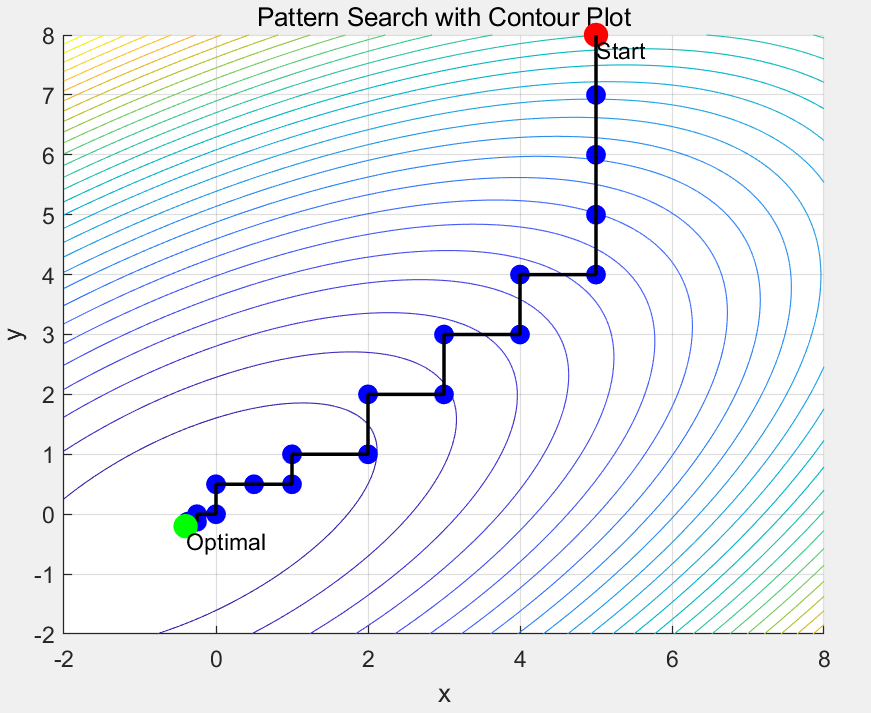
**Key codes:**



**Case1:**

Optimal point: (-0.4, -0.2)

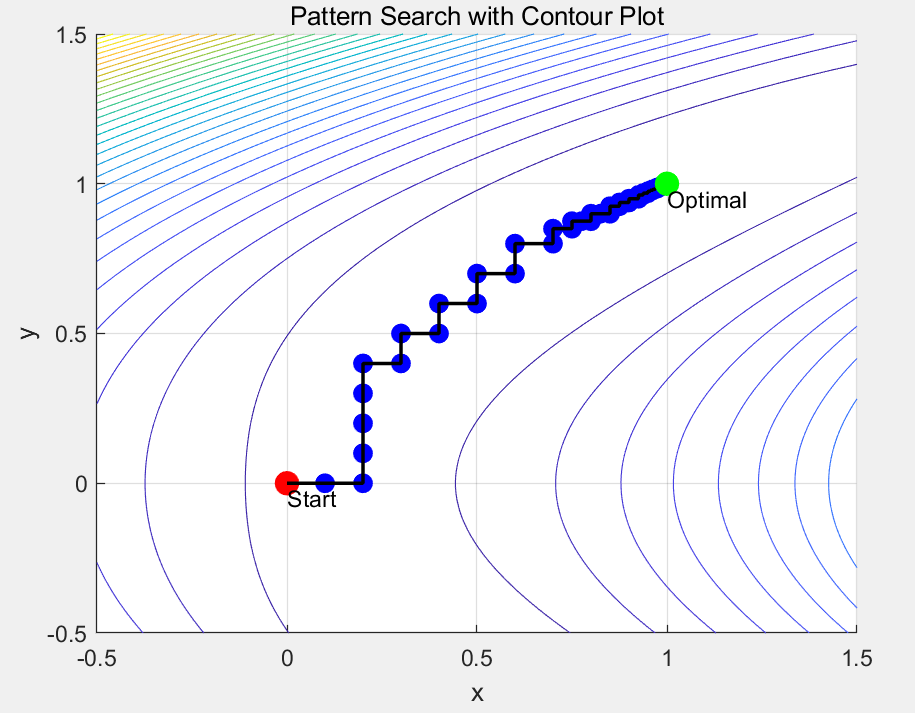
Minimum function value: -0.2



**Case2:**

Optimal point: (0.99999, 0.99999)

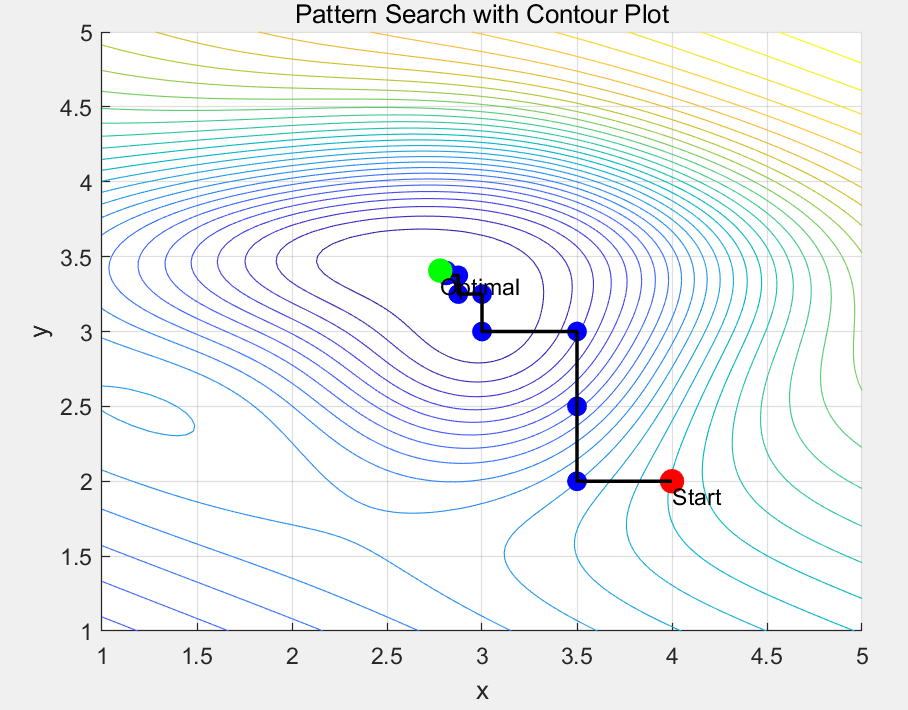
Minimum function value: 1.2573e-10



**Case3:**

Optimal point: (2.7802, 3.4065)

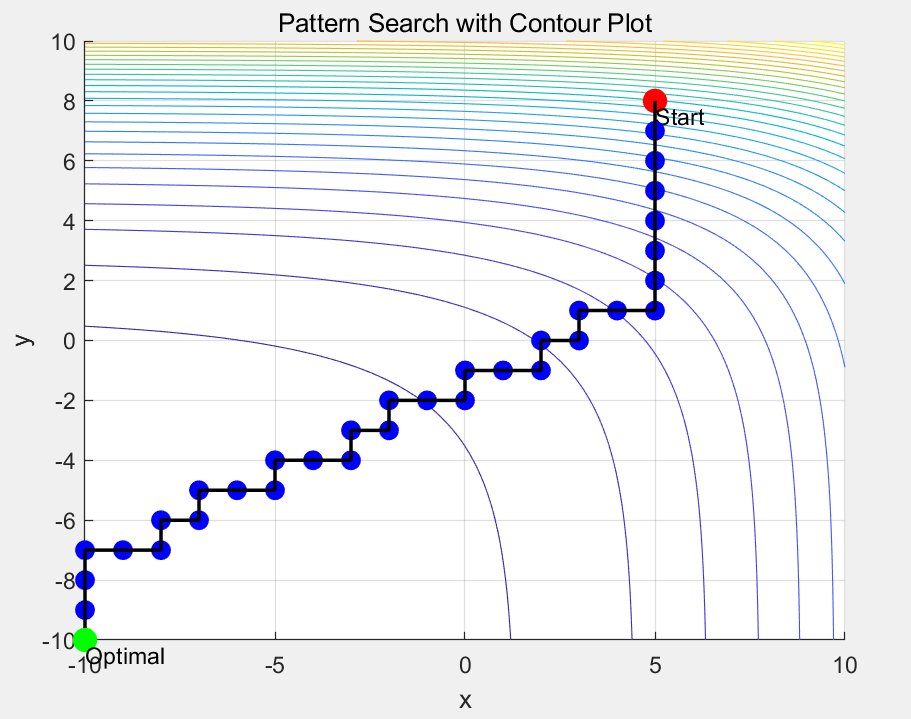
Minimum function value: 0.35675



**Case4:**

Optimal point: (-10, -10)

Minimum function value: 0.17101

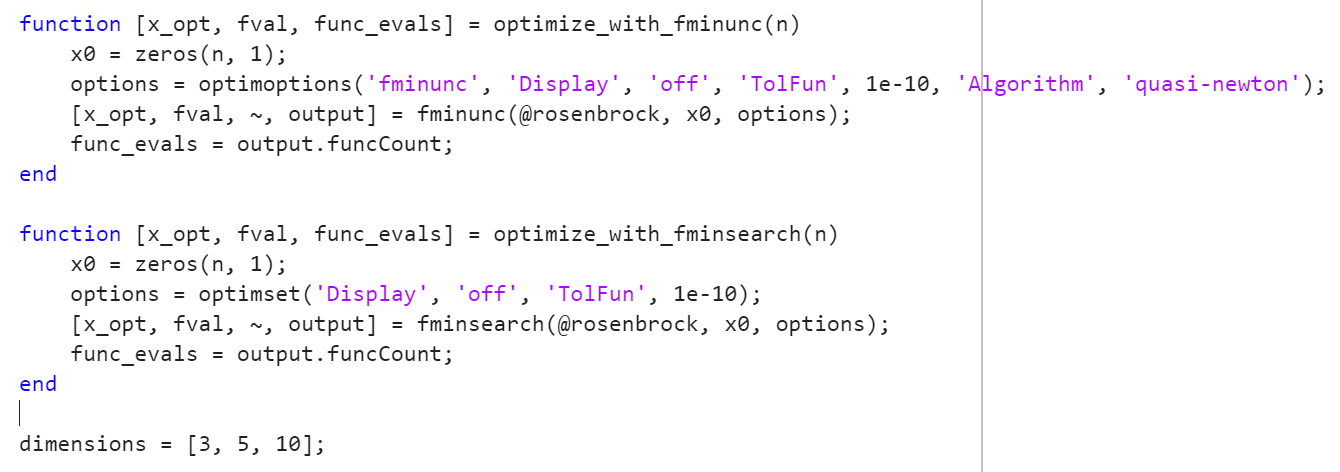


**Problem1B:**

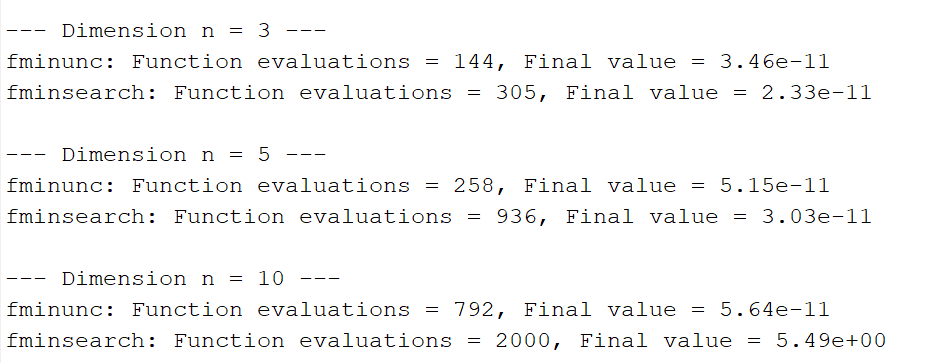
**Main ideas:**

fminunc and fminsearch are run in different dimensions.

**Main Codes:**



Result:

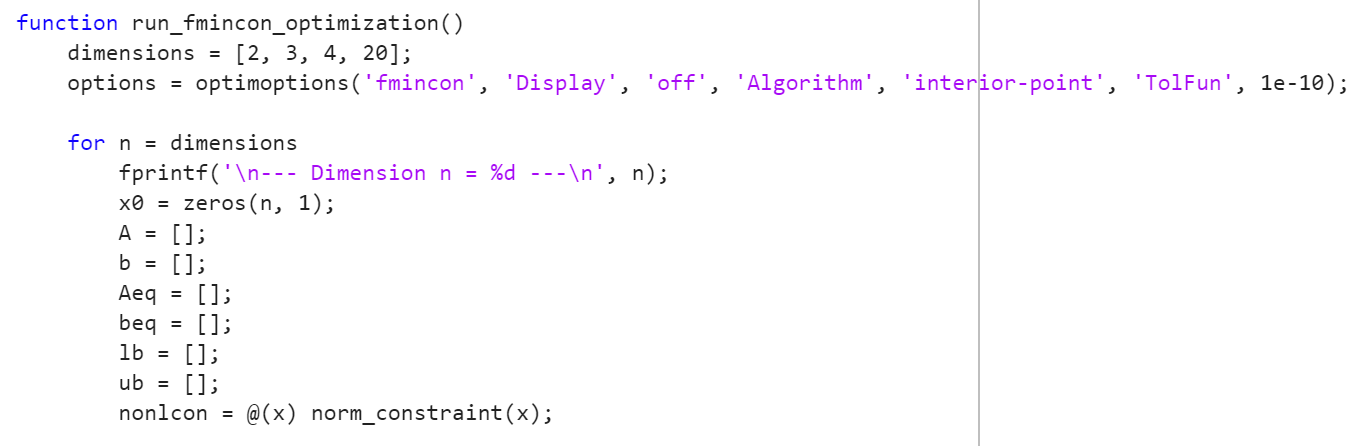


**Problem2:**

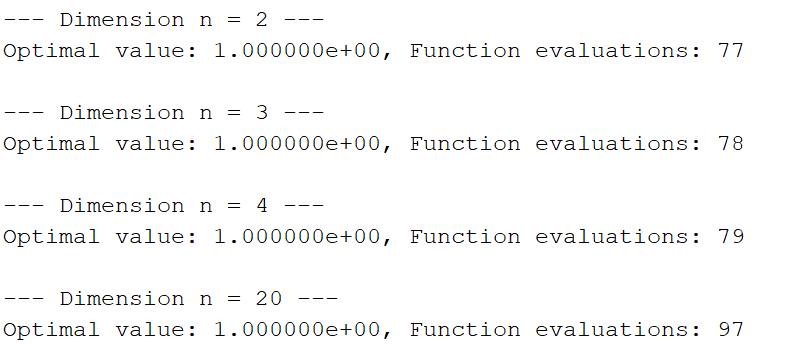
**Main ideas:**

Do like problem1b.

**Main codes:**



**Result:**

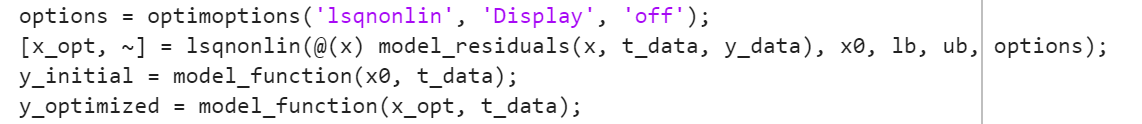


**Problem3:**

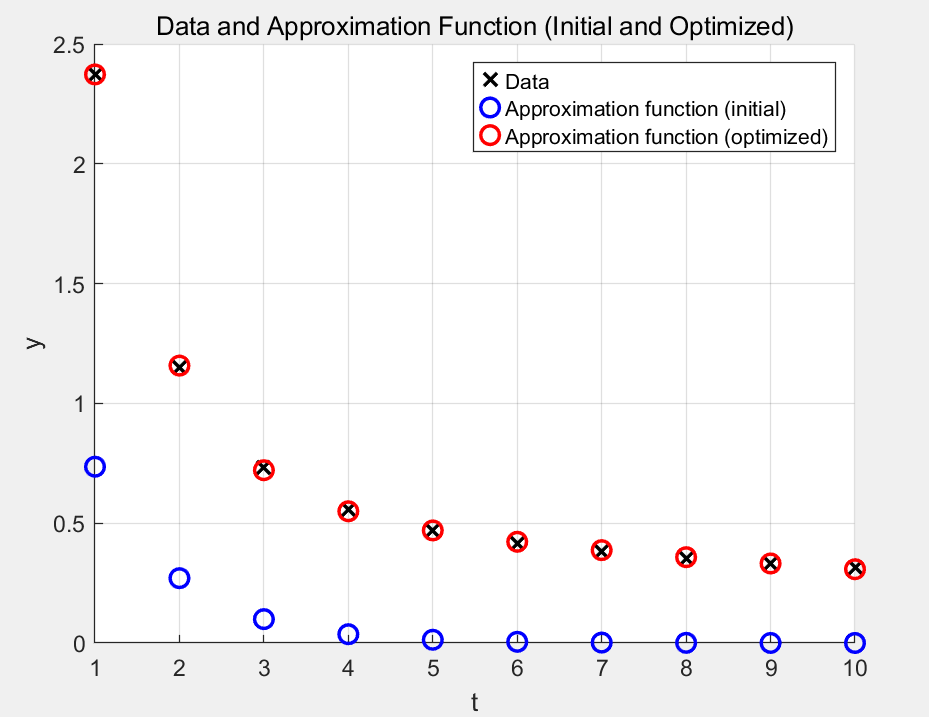
**Main ideas:**

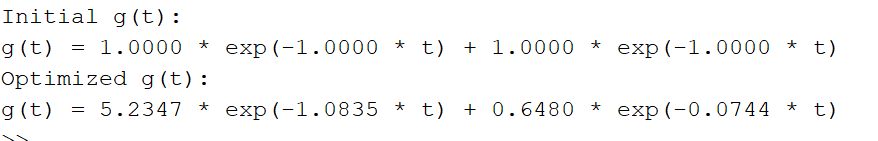
The code uses lsqnonlin to fit the function g(t) to given data points by optimizing parameters a, b, c and d. It prints the initial and optimized function expressions and plots the data along with the initial and optimized approximation functions.

**Main codes:**



**Result:**



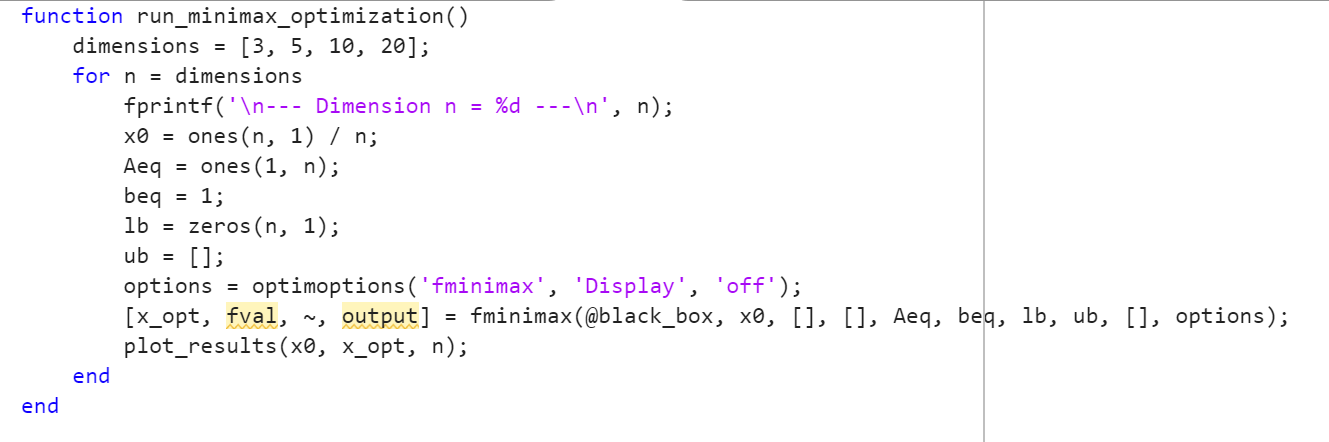


**Problem4:**

**Main ideas:**

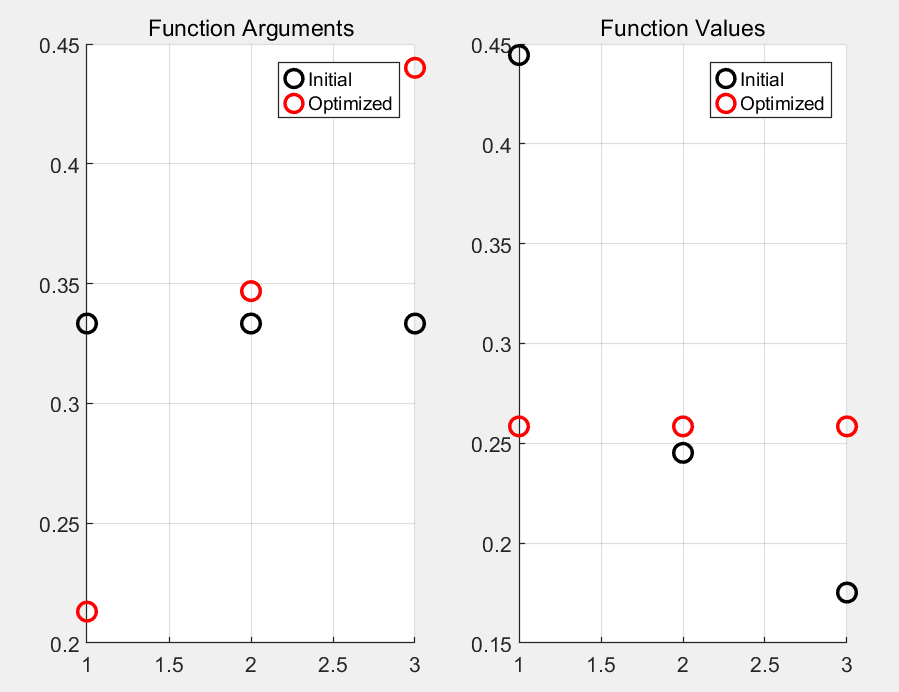
The code uses `fminimax` to minimize the maximum value of the `black\_box` function under equality and non-negativity constraints, visualizing the results for different dimensions.

**Main codes:**

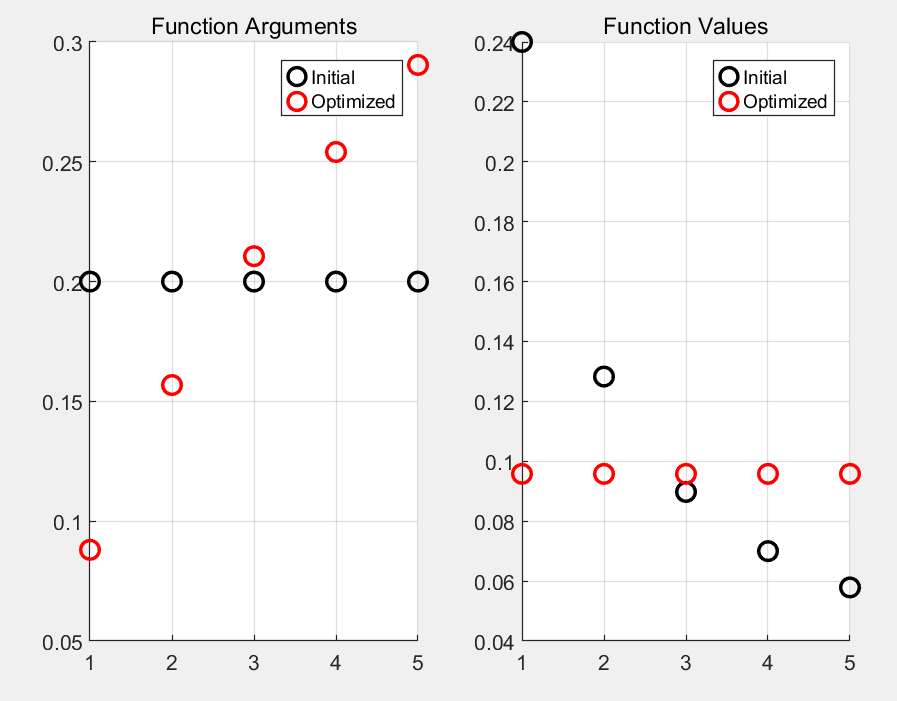


**Result:**

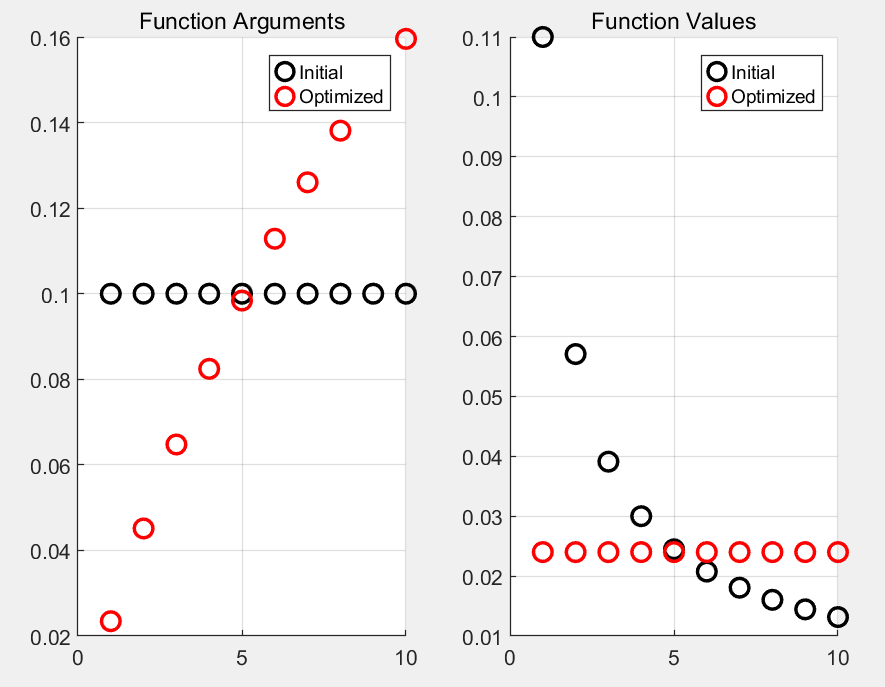
**n=3:**



**n=5:**



**n=10:**



**n=20:**

