VERKFRÆÐILEGAR BESTUNARAÐFERÐIR

Day 4

T-423-ENOP

Arnar Gylfi Haraldsson
arnarh23@ru.is

Hafþór Árni Hermannsson
hafthorh20@ru.is

Ragnheiður Gná Gústafsdóttir
ragnheidurg@ru.is

May 5, 2024

Exercise 1 - Golden Ratio Search

Description

Write a Matlab function that implements a golden ratio search algorithm. Let f be a function of a single parameter x. Let the (unique) minimum of f be bracketed by two points x_1 and x_2 ($x_1 < x_2$). Let $t = (1+5^{1/2})/2$ (golden ratio). Find two points x_3 and x_4 so that $(x_2 - x_3) = (x_2 - x_1)/t$ and $(x_4 - x_1) = (x_2 - x_1)/t$. Eliminate the point that has the largest value of $f(x_j)$, $j = 1,2,3,4$. Set $x_1 = x_3$ (if x_1 is eliminated) or $x_2 = x_4$ (if x_2 is eliminated). Iterate the procedure until the minimum is located within the prescribed tolerance.

Test the function using the following two cases:

1. $f(x) = (x-2)^2$, $x_1 = 0$, $x_2 = 10$;
2. $f(x) = x^2 + 3\exp(-2x)$, $x_1 = 0$, $x_2 = 10$.

In both cases, the tolerance should be set to 10^{-3}. Display the function value after each iteration of the algorithm.

Implementation

The function was implemented as described and the search visualised with an animation that showed the search interval decreasing until the solution was found satisfying the tolerance (10^{-3}).

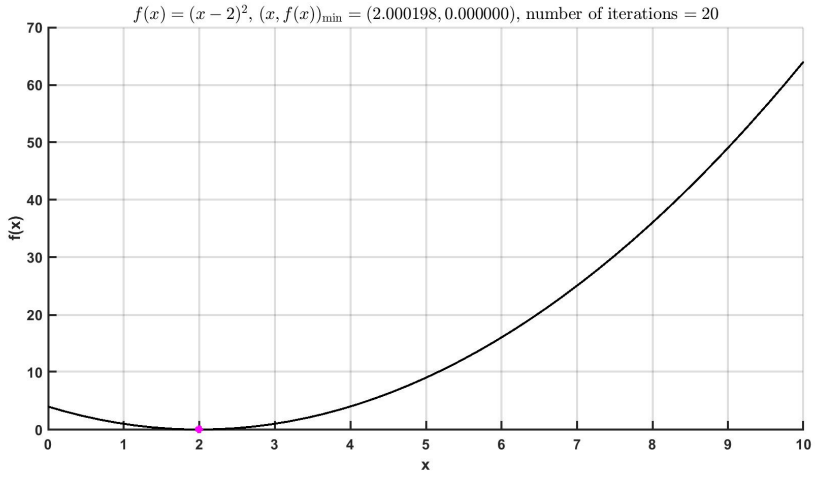


Figure 1: first function solution found with golden ratio search

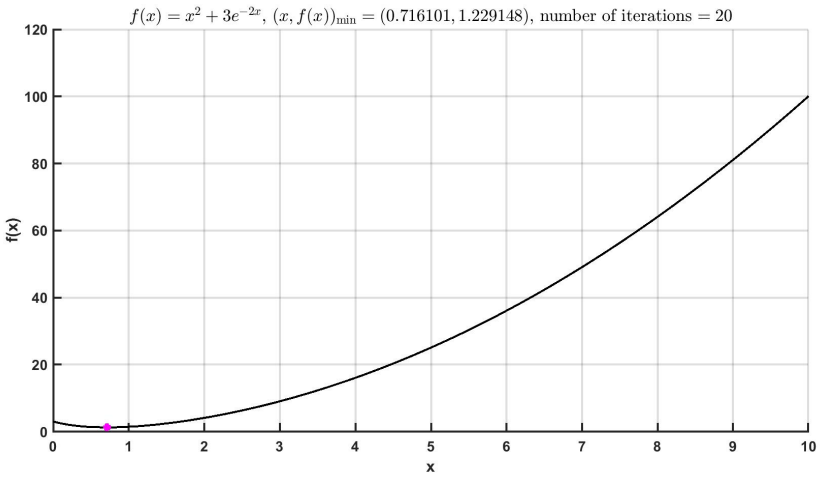


Figure 2: second function solution found with golden ratio search

For fun we also analyzed the effect of the ratio on the accuracy of the solution and the number of iterations needed to find the solution. The golden ratio function was tested with different ratios ranging from 1.35 - 1.7 and a step size of 5×10^{-4}

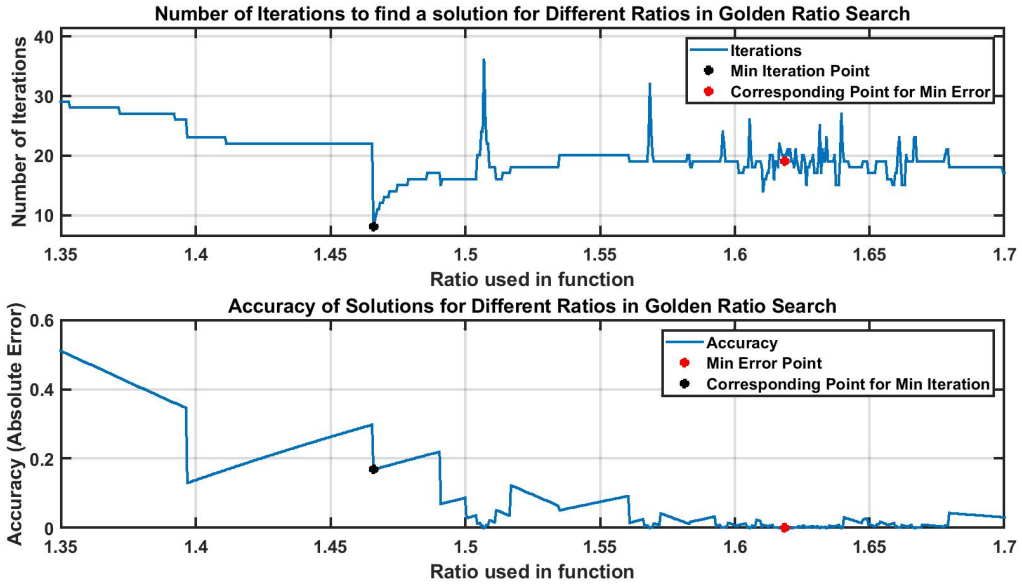


Figure 3: Comparison of the accuracy and number of itterations depending on ratio used in function

Interestingly, the most accurate solution was the one that used the golden ratio exactly in the function. But a lot of other ratios work similarly well in the range 1.593-1.679
The number of iterations fluctuated a lot and the golden ratio was not the one that would find the solution the fastest.

Exercise 2 - Newton's Method

Description

We implement the algorithm of Newton's method $\left(x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\right)$ on the four functions specified in the problem description.

x^2

$\ln(x)$

x^4

$\sqrt{x} - 2$

Our termination condition is met when the function value is smaller than 2^{-52}.

Results:

We compute the derivatives for each of the functions analytically.

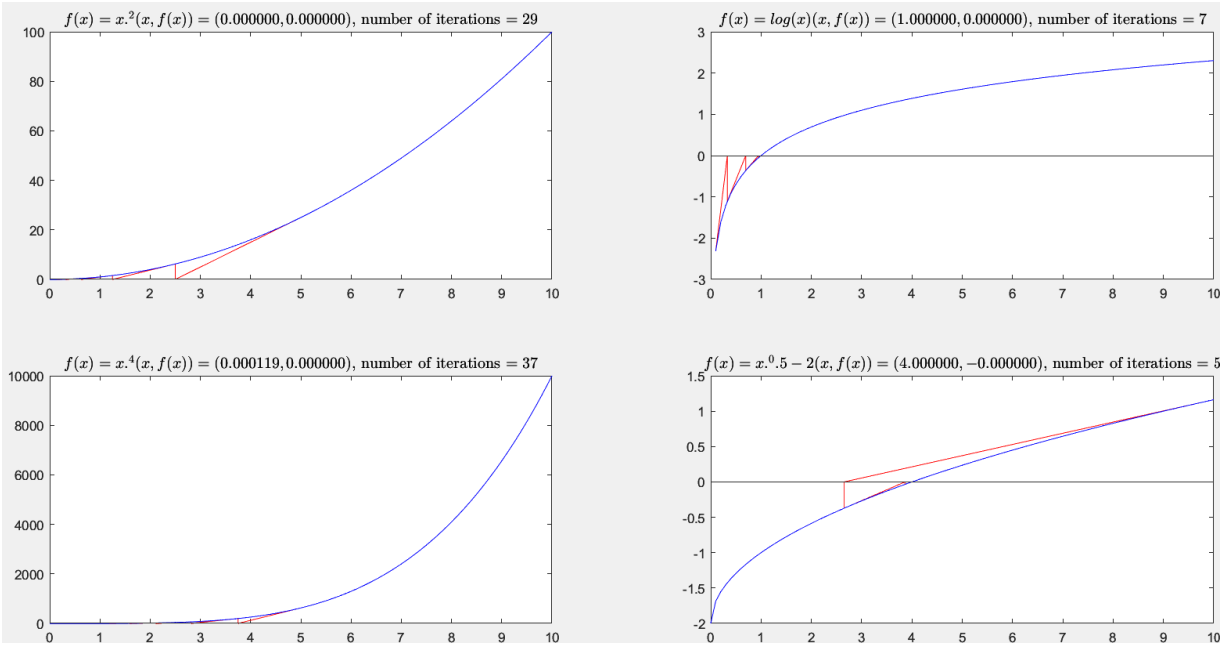
$$\frac{d}{dx}x^2 = 2x$$

$$\frac{d}{dx}\ln(x) = \frac{1}{x}$$

$$\frac{d}{dx}x^4 = 4x^3$$

$$\frac{d}{dx}\sqrt{x} - 2 = \frac{1}{2\sqrt{x}}$$

Using the same initial values x_0 as in the problem description, we get the following result:



Exercise 3 - Descent Method with Trust Region

Description

Implement a descent method with trust region that uses a first-order model of the objective function. The model should be obtained by estimating the objective function gradient using finite differences. The model can be optimized using any available method (e.g., golden ratio search).
Test your function using:

1. $f(x) = x_1^2 + ... + x_n^2$ for different n (use [1 2 ... n] T as a starting point)
2. $g(x) = sin(x)$ (use 2 as a starting point)
3. $h(x) = exp(x_1/5 + x_2/2) + x_1^2 + x_2^2$(use [5 3]T as a starting point).
4. Rosenbrock function (use [0 0]T as a starting point).

Functionality

The trust region descent algorithm is designed to find a local minimum of a given objective function $f : \mathbb{R}^n \to \mathbb{R}$. The function `trust_region_descent` accepts several parameters: the objective function f, an initial guess x, an initial trust region radius δ, a maximum number of iterations `kmax`, and a convergence tolerance `tol`.

Algorithm Overview

The algorithm iterates up to a maximum number of iterations or until a convergence criterion is met. Each iteration involves several key steps:

1. **Gradient Calculation:** The gradient of the function f at the current point x is approximated using central finite differences. This gradient informs the direction in which the function has the steepest ascent, and the opposite direction is considered for the descent step.
2. **Trust Region Subproblem:** A trial step h_{tr} is calculated by scaling the negative gradient by the current trust region radius δ. This step is constrained to lie within a ball defined by δ, ensuring that the modifications to x are not too large.
3. **Model Evaluation:** A first-order Taylor expansion is used to approximate the change in the function value due to the trial step. This model, denoted as $q(h)$, predicts the new function value and is compared against the actual function evaluation at $x + h_{tr}$.
4. **Trust Region Adjustment:** The ratio r of the actual reduction to the predicted reduction is calculated. If r is close to 1, the model is accurate, and the trust region may be expanded. If r is very low, the model is not accurate, leading to a reduction in the size of the trust region.
5. **Convergence Check:** The algorithm checks if the norm of the gradient or the norm of the trial step h_{tr} is below the specified tolerance, indicating potential convergence to a local minimum.

Plotting Convergence

To aid in the analysis of the algorithm's performance, the function records the values of x at each iteration and plots these after the completion of the iterations. This visual representation shows the convergence behavior of each component of x, providing insights into the dynamics of the optimization process.

Results

This Matlab function using descent method with trust region was then applied to the functions given with the starting points given, $\delta = 0.1$, tolerance = 1e-8 and kmax = 100 (kmax = 10.000 for Rosenbrock for a closer result to the analytical one). The results of are displayed in table 1 and figures 4 to 7 show the convergence of the local minima against the iterations. The analytical result for the h function

was obtained visually as we had a hard time finding the actual result, it may have a small error that we deemed insignificant.

Function	Iterations	Result	Analytical Result
f	39	$1 \times 10^{-8} \begin{bmatrix} -0.0112 & -0.0224 & -0.0335 & -0.0447 & -0.0559 \\ -0.0671 & -0.0783 & -0.0895 & -0.01006 & -0.01118 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
g	34	4.7124	4.71239
h	34	$\begin{bmatrix} -0.0880 & -0.2200 \end{bmatrix}$	approx. $\begin{bmatrix} -0.0880 & -0. \end{bmatrix}$
Rosenbrock	10,000	$\begin{bmatrix} 0.9999 & 0.9999 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \end{bmatrix}$

Table 1: Local minima of each function using the descent method with trust region.

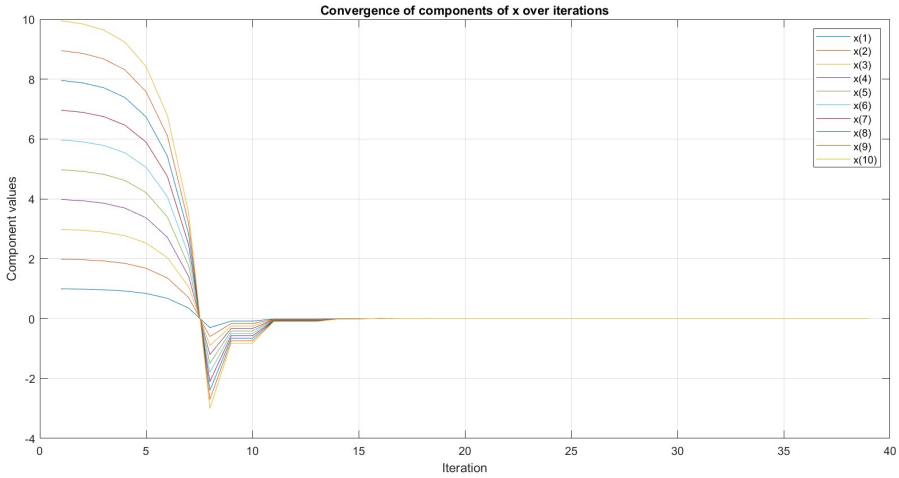


Figure 4: Convergence of the local minima with iterations for f(x)

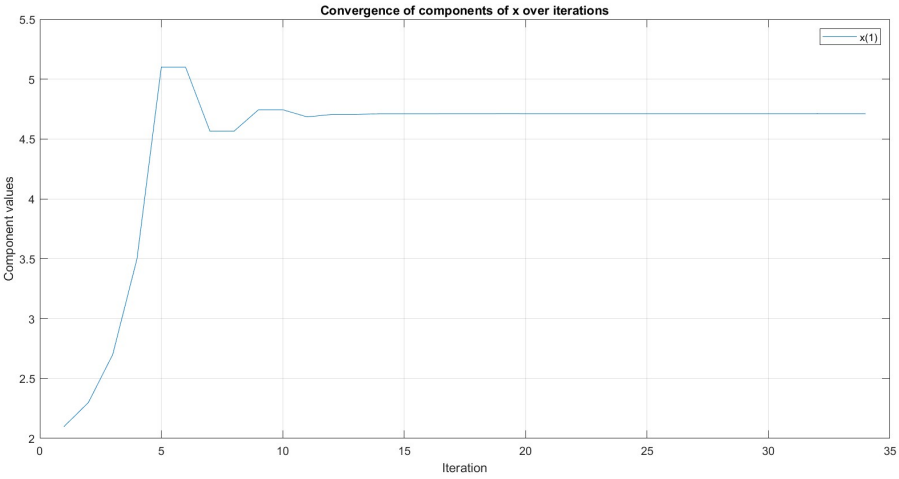


Figure 5: Convergence of the local minima with iterations for g(x)

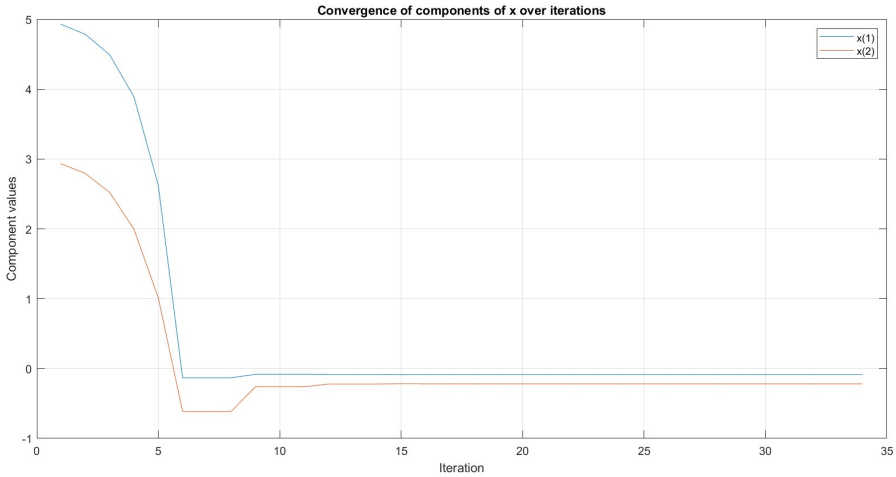


Figure 6: Convergence of the local minima with iterations for h(x)

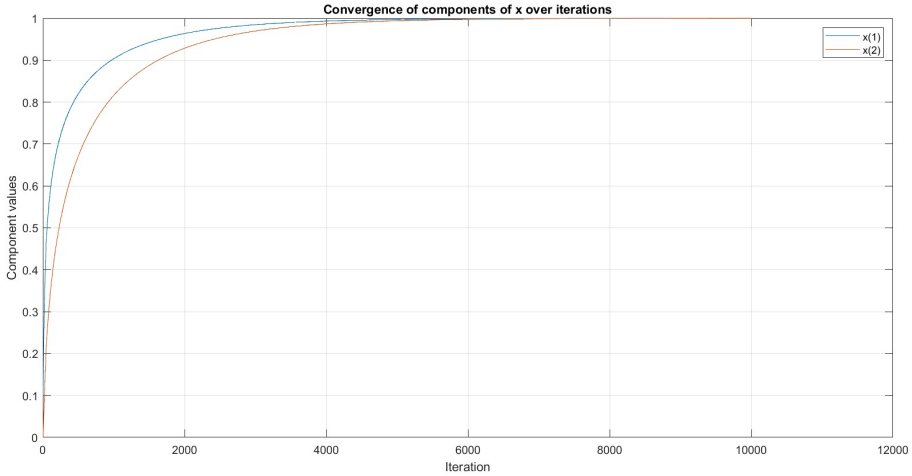


Figure 7: Convergence of the local minima with iterations for Rosenbrock

As so many iterations are needed to get close to the analytical solution for the Rosenbrock function we decided to try to optimize it with the golden ratio search. We did that by obtaining the htr using alpha_min from the golden ratio search algorithm developed in exercise 1 and multiplying that with the gradient. This produced a prefect result in fewer iterations for Rosenbrock but much worse for the other three, especially f and h. The results are displayed in table 2 and figures 8 to 11 show the convergence of the solutions.

Function	Iterations	Result	Analytical Result
f	101	$\begin{bmatrix} 0.9230 & 1.8461 & 2.7691 & 3.6921 & 4.6152 \\ 5.5382 & 6.4613 & 7.3843 & 8.3073 & 9.2304 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
g	101	4.7269	4.71239
h	101	$\begin{bmatrix} 4.5350 \\ 2.5686 \end{bmatrix}$	N/A
Rosenbrock	8641	$\begin{bmatrix} 1.0000 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \end{bmatrix}$

Table 2: Local minima of each function using golden ratio search and the descent method with trust region.

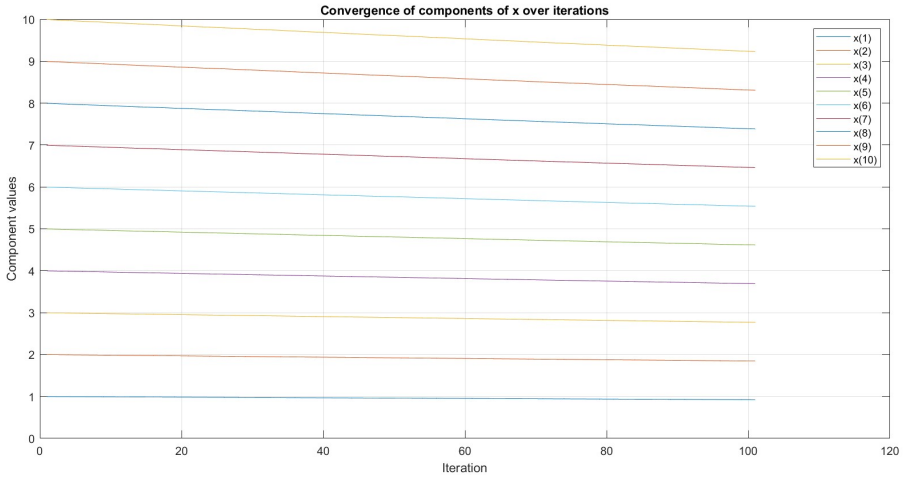


Figure 8: Convergence of the local minima with iterations for f(x) with golden ratio search

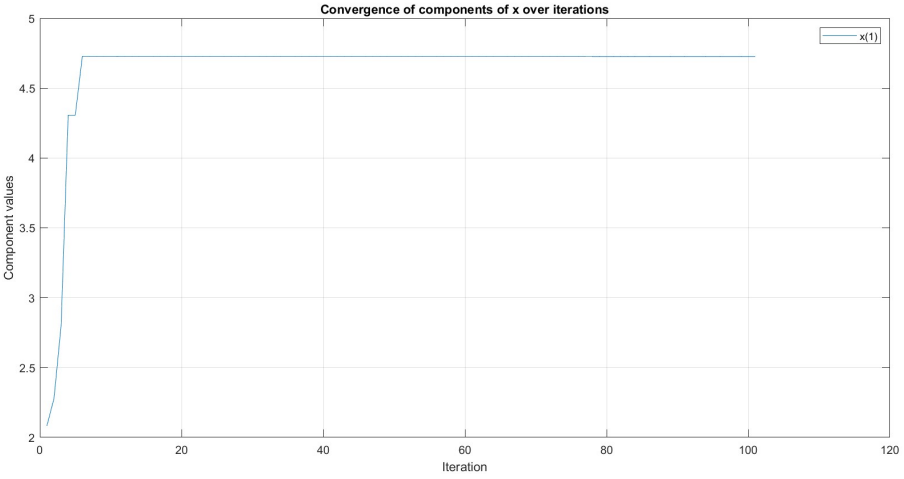


Figure 9: Convergence of the local minima with iterations for g(x) with golden ratio search

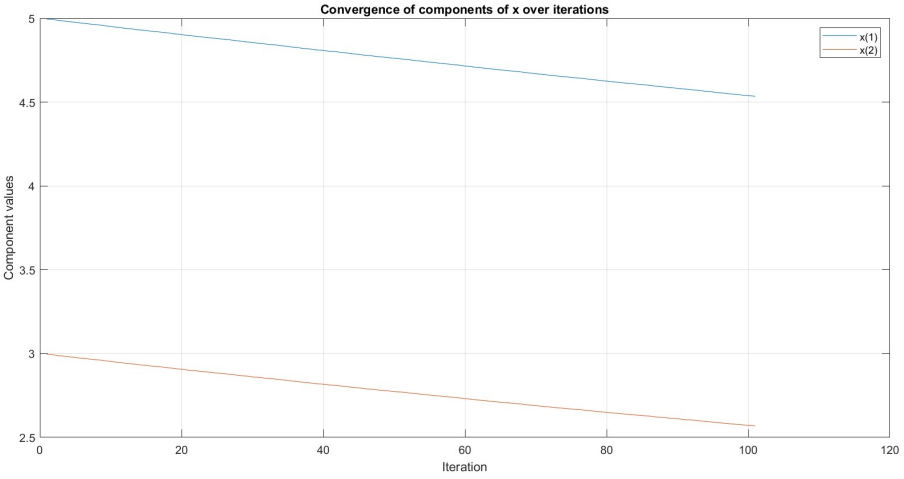


Figure 10: Convergence of the local minima with iterations for h(x) with golden ratio search

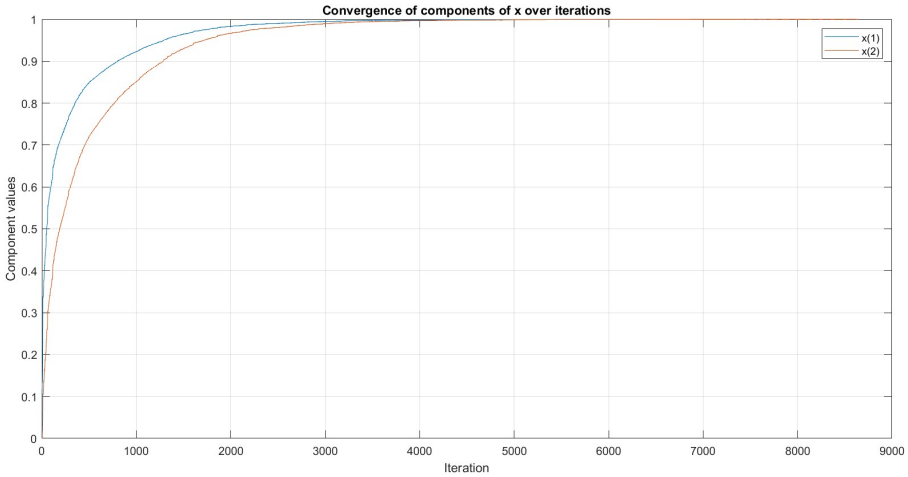


Figure 11: Convergence of the local minima with iterations for Rosenbrock with golden ratio search