Optimization Techniques
Electrical Computer and Engineering Dept.
Aristotle University of Thessaloniki
7th Semester - November 2025

Anastasia Loukeri
Student ID:10975
aloukeri@ece.auth.gr

General Problem Description

Goal: The objective is to find the minimum of a given convex function $f(x)$ where $x \in [a, b]$. More specifically, with an initial interval $[-1, 3]$, we are asked to calculate a new interval $[a_k, b_k]$ of a predetermined accuracy $\ell > 0$, so that $b_k - a_k \leq \ell$, in which the minimum x^* of the functions we are looking for is contained:

- $f_1(x) = 5^x + (2 - cos(x))^2$
- $f_2(x) = (x-1)^2 + e^{x-5} \cdot sin(x+3)$
- $f_3(x) = e^{-3x} - (\sin(x-2) - 2)^2$

The search will use minimum search methods **without the use of derivatives** (Bisection Method, Golden Section Method, Fibonacci Method), and also with one method that **uses derivatives** (Bisection Method with the use of Derivatives).

All the above methods can be applied since they are based on Theorem 5.1.1 (of the book), a basic prerequisite of which is that the functions f_i must be **strictly almost convex** in the interval $[a, b]$ where we study them. The plots of the functions f_i in $[-1, 3]$ are as follows:

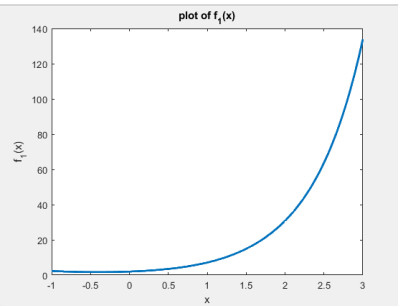

Figure 1: Plot of f_1

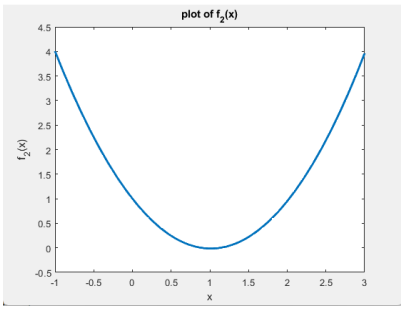

Figure 2: Plot of f_2

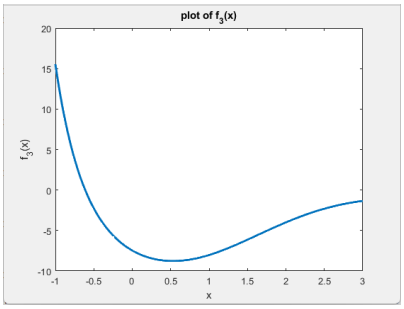

Figure 3: Plot of f_3

Therefore, the prerequisite mentioned holds (our functions are strictly convex, thus also strictly almost convex, in $[-1, 3]$).

Finally, the algorithm using derivatives is also applicable to all f_i we have, as they are **differentiable** in the interval we study (with a known computable derivative).

The Bisection - Dichotomous Method without the use of Derivatives

The method for finding the minimum of a function $f(x)$ (when referring to a function $f(x)$, we always assume that it satisfies the prerequisites previously mentioned for the corresponding method) starts from the initial interval $[a, b]$ – in which we seek to find the minimum x^* – and in each iteration calculates a new interval $[a_k, b_k]$, smaller than the previous one (where k is the iteration), in which x^* is again present.

The method stops at iteration n for which the termination criterion holds:

$$b_n - a_n \leq \ell$$

for a given ℓ (i.e., for a given accuracy of the final interval).

For the calculation of the next interval $[a_{k+1}, b_{k+1}]$ in each iteration:

- We take two points $x_{1,k}, x_{2,k} \in [a_k, b_k]$ around the bisector of this interval at a distance ϵ from it:
$$x_{1,k} = \frac{a_k + b_k}{2} - \epsilon \quad \text{and} \quad x_{2,k} = \frac{a_k + b_k}{2} + \epsilon$$
- We calculate the values of f at these points, $f(x_{1,k}), f(x_{2,k})$, and compare:
  - If $f(x_{1,k}) < f(x_{2,k}) \implies a_{k+1} = a_k, b_{k+1} = x_{2k}$
  - Otherwise $f(x_{1,k}) > f(x_{2,k}) \implies a_{k+1} = x_{1k}, b_{k+1} = b_k$
- The algorithm stops when the termination criterion is satisfied.

For this algorithm to work correctly, ϵ must satisfy:

$$2 \cdot \epsilon < \ell$$

From the above, we understand that the objective function f performs 2 calculations in each iteration. Therefore, since k expresses the current iteration, we will have $2 \cdot k$ calculations up to that moment. (We consider $k = 0$ before the first iteration, i.e., $a_0 = a, b_0 = b$).

Requirement 1 - Constant final search range $\ell = 0.01$

Taking various values for ϵ, we calculate, for the constant ℓ, the number of calculations of the objective function for each of our 3 functions, taking into account the above setup regarding the relationship between the number of iterations and the number of calculations of the objective function.

The plots we obtain for each function are shown below with the horizontal axis of ϵ being on a scale of 10^{-3}:

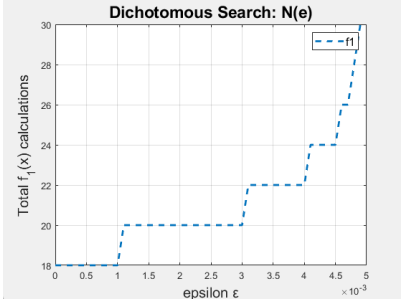

Figure 4: Plot for f_1

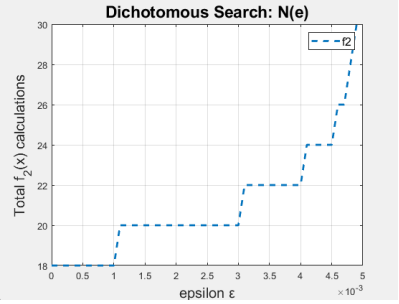

Figure 5: Plot for f_2

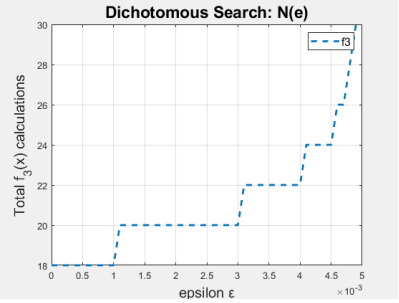

Figure 6: Plot for f_3

Computational Analysis

1. We observe that as ϵ increases and approaches the value $\ell/2 = 0.005$, more calculations of the objective function are required, which makes the algorithm slower (more computational cost).

2. Also, we observe that for all functions the plots are identical.That was expected since the number of iterations of the algorithm depends only on ℓ, ϵ, and the initial interval $[a, b]$ and those parameters are the same for all of the three functions.

3. The fact that the plot shows plateaus (levels - like step function) is related to the fact that for very small changes in ϵ, there is no practical change for the given accuracy expressed by ℓ.

Requirement 2 - Constant $\epsilon = 0.001$

Similarly, taking various values for ℓ, we calculate the total number of objective function calculations for each of our 3 functions, keeping ϵ constant.

The plots we obtain for each function are shown below:

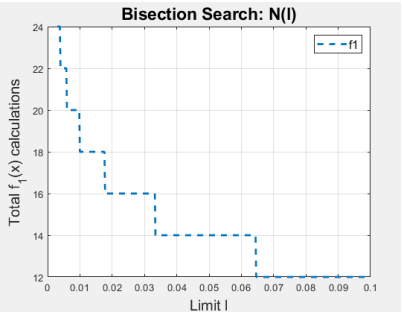

Figure 7: Plot for f_1

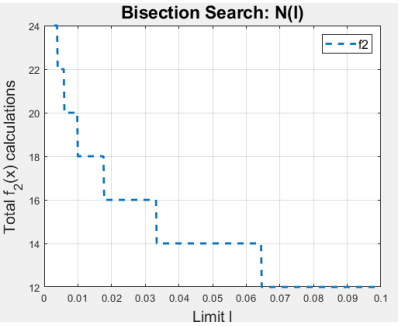

Figure 8: Plot for f_2

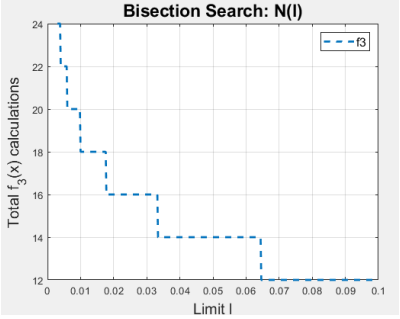

Figure 9: Plot for f_3

Computational Analysis

- It is evident that increasing the tolerance ℓ (i.e., relaxing the convergence criterion for the final interval width, $b_n - a_n \leq \ell$) leads to a reduction in the required number of objective function evaluations ($2n$). This inverse relationship is consistent with optimization theory, as a lower precision requirement necessitates fewer iterative steps, thereby enhancing computational efficiency.

- The computational cost, quantified by $2n$, remains invariant across all three objective functions (f_1, f_2, f_3). This is due to the inherent properties of the Dichotomous Search Method, where the number of iterations n is solely determined by the initial interval $[a, b]$ and the prescribed parameters ℓ and ϵ, independent of the specific functional form of $f(x)$.

- The observed step-function behavior (plateaus) in the graphical representation is a consequence of the discrete nature of the iteration count n. Minor, continuous variations in the tolerance ℓ do not result in a change in the integer value of n required to satisfy the termination condition for a fixed ϵ.

- To achieve higher resolution in the graphical representation, one must either increase the sampling density of the parameter ℓ across the domain or restrict the domain of ℓ under consideration.

Requirement 3 - Plot of Subintervals $[a_k, b_k]$ for Various ℓ

For various values of the tolerance ℓ, we calculate all the upper and lower bounds, a_k and b_k, of the subintervals for a constant ϵ.

For each function separately, we create a c plot for both bounds (a_k and b_k) for various values of ℓ. and iterations K:

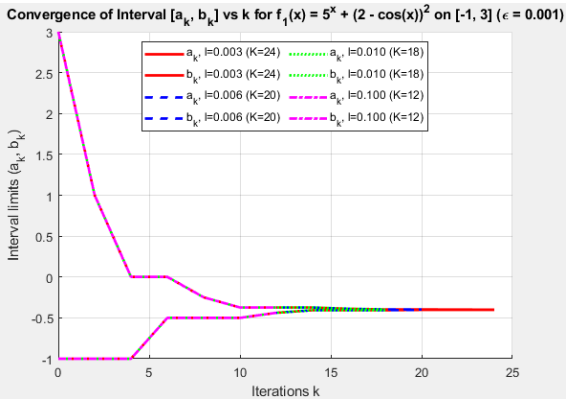

Figure 10: Plot of f_1

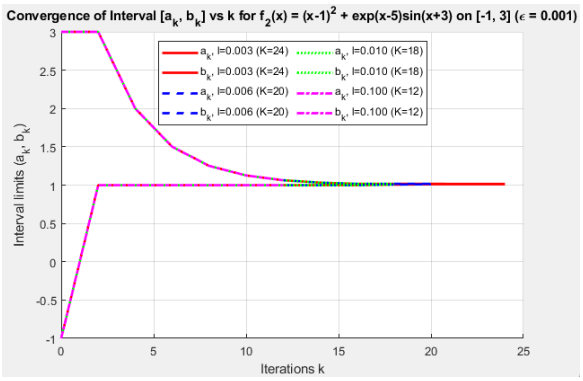

Figure 11: Plot of f_2

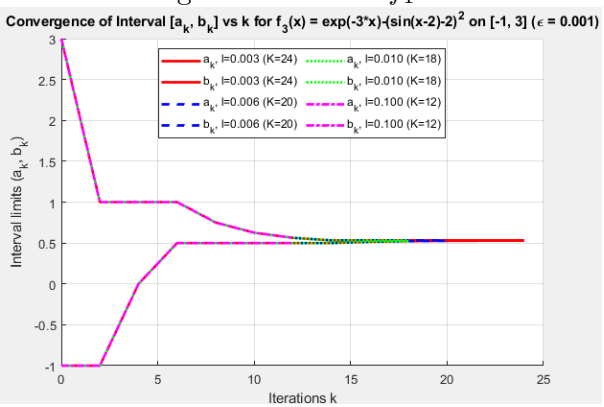

Figure 12: Plot of f_3

Analysis of Interval Convergence

Convergence of Subintervals

- The two interval bounds, a_k and b_k, exhibit asymptotic convergence, confirming the efficacy of the iterative interval reduction process.
- It is observed that decreasing the tolerance ℓ (i.e., requiring higher precision) necessitates a greater number of total iterations (n) and, consequently, more function evaluations. This outcome demonstrates the fundamental trade-off between algorithmic accuracy and computational effort.
- The identical graphical representation across all three objective functions is maintained. This is because the value of ℓ serves exclusively as a stopping criterion, determining the point at which the desired interval width is achieved. It does not influence the intermediate step-by-step calculations of the interval reduction process.

- The mechanism of interval reduction ensures that in each iteration k, only one boundary is updated: either the upper bound (b_k) decreases, or the lower bound (a_k) increases. Simultaneous modification of both bounds is precluded by the method's design.

Comments on Algorithmic Complexity - Efficiency

The Bisection Method is notable for its conceptual simplicity and ease of implementation in unconstrained optimization.

However, in terms of computational cost, the method is considered inefficient compared to alternative search techniques. The fundamental requirement of two objective function evaluations per iteration (i.e. $2n$ total operations) effectively doubles the algorithmic time complexity relative to methods that can achieve similar interval reduction with fewer evaluations. This characteristic places the Bisection Method at a computational disadvantage.

The Golden Section Search Method

The Golden Section Search Method is an iterative technique for locating the unconstrained minimum x^* of a unimodal objective function $f(x)$ over a specified closed interval $[a, b]$. At each iteration k, the method generates a new, progressively smaller subinterval $[a_k, b_k]$, which is guaranteed to contain the minimizer x^*.

The iterative process is terminated at the n-th step when the stopping criterion is satisfied: $b_n - a_n \leq \ell$, where ℓ is a predefined tolerance that dictates the precision of the final interval length.

The ratio of reduction in the interval length per iteration is governed by the golden ratio $\phi = \frac{1+\sqrt{5}}{2} \approx 1.6180$. Specifically, the length of the subsequent interval $[a_{k+1}, b_{k+1}]$ is given by the relation $b_{k+1} - a_{k+1} = \gamma(b_k - a_k)$, where $\gamma = \phi^{-1} = \frac{\sqrt{5}-1}{2} \approx 0.6180...$, which is the positive root of the characteristic equation $\gamma^2 + \gamma - 1 = 0$.

The core of the method involves defining two interior points $x_{1,k}$ and $x_{2,k}$ within the current bracketing interval $[a_k, b_k]$ such that they exhibit symmetry based on γ:

- Interior Points: $x_{1,k} = a_k + (1-\gamma)(b_k - a_k) = b_k - \gamma(b_k - a_k)$ and $x_{2,k} = a_k + \gamma(b_k - a_k)$, where $x_{1,k}, x_{2,k} \in (a_k, b_k)$.
- Interval Reduction: The values of the objective function are evaluated at $f(x_{1,k})$ and $f(x_{2,k})$, and their comparison is used to discard one-third of the current interval, thus commencing the next iteration.

Iterative Step (While Loop):

- Comparison of Function Values: The values $f(x_{1,k})$ and $f(x_{2,k})$ are compared:

$$\begin{aligned} \text{If } f(x_{1,k}) < f(x_{2,k}) &\Rightarrow [a_{k+1}, b_{k+1}] = [a_k, x_{2,k}] \\ &\Rightarrow x_{1,k+1} = x_{1,k}, \quad x_{2,k+1} = x_{1,k} \\ \text{Else if } f(x_{1,k}) > f(x_{2,k}) &\Rightarrow [a_{k+1}, b_{k+1}] = [x_{1,k}, b_k] \\ &\Rightarrow x_{1,k+1} = x_{2,k}, \quad x_{2,k+1} = b_k - (1-\gamma)(b_k - a_k) \end{aligned}$$

- Termination: The algorithm proceeds until the stopping criterion is met.

A crucial efficiency characteristic of the Golden Section Search is the management of function evaluations. Two function evaluations are required for the initial iteration ($k = 0$), i.e., $f(x_{1,0})$ and $f(x_{2,0})$. In all subsequent iterations ($k \geq 1$), only **one** new function evaluation is needed. This is because the γ-ratio ensures that one of the previous interior points becomes one of the new interior points ($x_{1,k+1}$ or $x_{2,k+1}$), and its function value is already known.

Consequently, if n represents the total number of iterations performed within the while loop, the total number of function evaluations N is given by $N = 2+n$. For instance, $n = 3$ iterations correspond to $N = 2 + 3 = 5$ total function evaluations. Upon termination, the final interval is defined by a_n and b_n.

The definition used to calculate the total number of function evaluations N based on the iteration count n is often expressed as $N = n+1$ in some contexts, where n represents the total number of new interior points calculated.

Requirement 1 - Parametric search range ℓ

Determine the range of the index ℓ utilized to calculate the number of objective function evaluations for each of the three functions, considering the observed relationship between the number of while loop iterations and the corresponding number of function evaluations.

The following diagrams illustrate the determination of the tolerance ℓ values for each function:

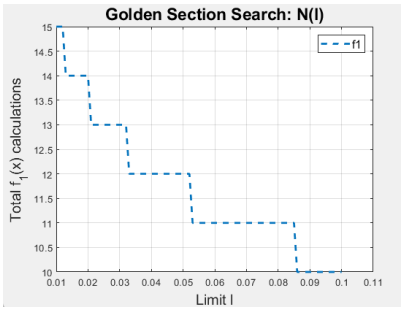

Figure 13: Plot of f_1

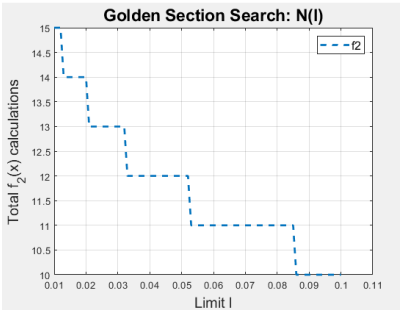

Figure 14: Plot of f_2

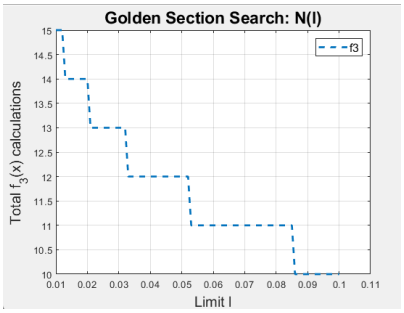

Figure 15: Plot of f_3

Observations on Precision and Efficiency

It is observed that as the stopping tolerance ℓ is increased, fewer total evaluations of the objective function are required. This relationship holds because a larger value for ℓ corresponds to a larger final interval width and consequently, a lower precision in locating the minimizer, necessitating fewer iterations.

For all functions considered, the convergence behavior, represented by the relationship between the number of iterations and the interval reduction, is **always the same**. The total number of iterations required by the algorithm depends exclusively on the precision ℓ and the initial interval $[a, b]$.

The fact that the graphical representation of the final interval width may exhibit plateaus suggests that, for very small changes in ℓ, there may be no practical alteration in the final interval width. If we wish to achieve greater resolution in the graphical representations, we must either compute more points or reduce the range on the horizontal axis.

Requirement 2 - Plotting Subintervals $[a_k, b_k]$ for various values of tolerance ℓ

For different values of the tolerance ℓ, we are asked to calculate and plot the lower and upper bounds, a_k and b_k, of the subintervals for the given functions. A common plot should be generated for each function separately, showing the two bounds across different values of ℓ.

The resulting diagrams for the bounds of every function are shown below:

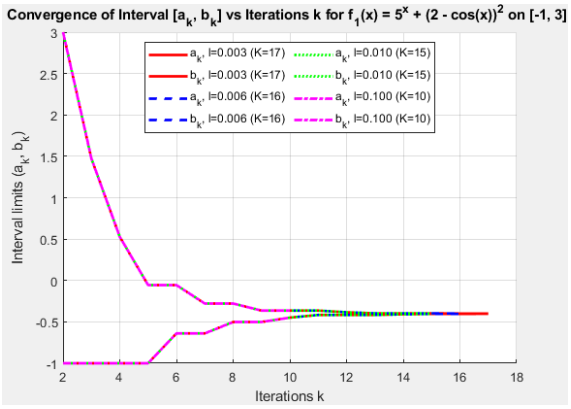

Figure 16: Plot of f_1

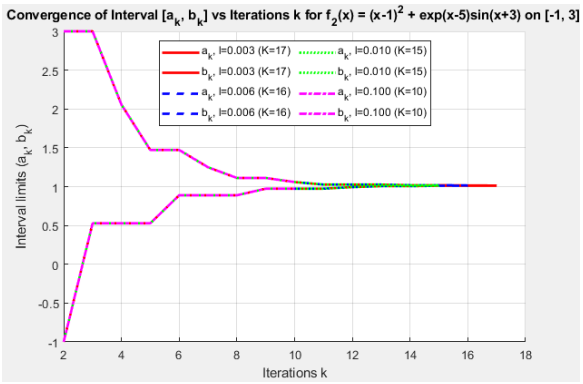

Figure 17: Plot of f_2

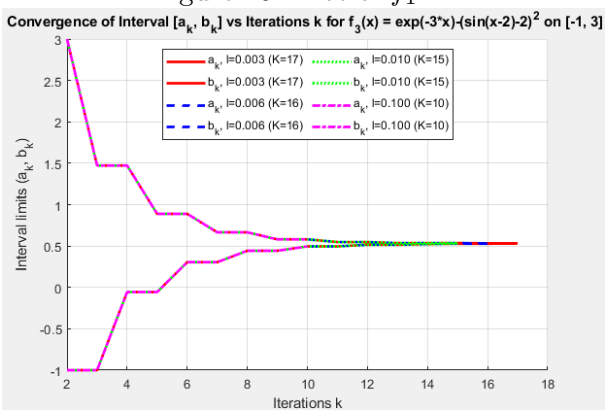

Figure 18: Plot of f_3

Convergence Analysis

It is observed that as the stopping tolerance ℓ is decreased, a proportionally greater number of total function evaluations/iterations n is required. This is a direct consequence of the method's objective: a smaller ℓ specifies a narrower final interval width, demanding higher precision in the minimizer's location, and thus necessitates more computational steps.

The convergence pattern remains consistent across all analyzed functions. Changes in ℓ only affect the total number of iterations required; specifically, a reduction in ℓ causes the algorithm to proceed for a few additional iterations. This consistency is upheld because the objective function $f(x)$ is only checked against the convergence criterion (target precision) to terminate the algorithm, regardless of the value of ℓ.

It is evident that at each iteration k, the boundaries of the interval $[a_k, b_k]$ are updated as follows: either the lower bound a_k is increased (moves up) or the upper bound b_k is decreased

(moves down). Critically, there is never a simultaneous change in both bounds within a single iteration.

Concluding Remarks - Algorithmic Efficiency

The fundamental efficiency of the Golden Section Search method lies in the mechanism by which the bracketing interval $[a_k, b_k]$ is reduced at each step. As detailed previously, this method enables the computational cost to be significantly minimized by requiring only one new objective function evaluation per iteration, following the initial two evaluations needed to commence the algorithm.

It should be noted that while the method is efficient, there is an inherent risk of requiring a substantial number of iterations if an overly strict tolerance ℓ is chosen, which may lead to a computationally unsatisfactory result given the total resources expended.

The Fibonacci Search Method

The Fibonacci Search Method is an iterative technique for locating the minimum x^* of a unimodal objective function $f(x)$ over an initial bracketing interval $[a, b]$. At each iteration k, the method generates a new subinterval $[a_k, b_k]$ guaranteed to contain the minimizer x^*, where k is the iteration index.

The process terminates when the interval length satisfies the condition: $(b_k - a_k)/\ell \leq F_n$, where ℓ is a predefined final interval length tolerance, n is the total number of iterations, and F_n is the n-th Fibonacci number (with the convention $F_0 = F_1 = 1$).

We employ the ratio criterion, known from the properties of Fibonacci numbers, for the interval reduction. This specific selection of the ratio is the fundamental computational advantage of the Fibonacci method.

Initial Step and Iterative Reduction

For the determination of the subsequent interval $[a_{k+1}, b_{k+1}]$ in the first iteration:

- We define two interior points: $x_1 = a + \frac{F_{n-2}}{F_n}(b - a)$ and $x_2 = a + \frac{F_{n-1}}{F_n}(b - a)$.
- We evaluate the function values at these two points, $f(x_1)$ and $f(x_2)$, and proceed to the iterative `for` loop.

Iterative Step (For Loop):

- Comparison of Function Values: The values $f(x_{1,k})$ and $f(x_{2,k})$ are compared:

$$\begin{aligned}
&\text{If } f(x_{1,k}) < f(x_{2,k}) \Rightarrow a_{k+1} = a_k, \quad b_{k+1} = x_{2,k} \text{ and} \\
&\qquad x_{1,k+1} = a_{k+1} + \frac{F_{n-k-2}}{F_{n-k}}(b_{k+1} - a_{k+1}) = x_{1,k}, \quad x_{2,k+1} = x_{1,k} \\
&\text{Else if } f(x_{1,k}) > f(x_{2,k}) \Rightarrow a_{k+1} = x_{1,k}, \quad b_{k+1} = b_k \text{ and} \\
&\qquad x_{1,k+1} = x_{2,k}, \quad x_{2,k+1} = a_{k+1} + \frac{F_{n-k-1}}{F_{n-k}}(b_{k+1} - a_{k+1})
\end{aligned}$$

- Termination: The algorithm proceeds until the `for` loop is completed (i.e., the termination criterion is met).

From the above, we deduce that the objective function f requires 2 initial evaluations in the first step. For every subsequent iteration, only one new evaluation is needed. This is because the Fibonacci ratio property ensures that one of the previous interior points becomes one of the new interior points ($x_{1,k+1}$ or $x_{2,k+1}$), and its function value is already known.

Assuming the index starts at $k = 0$ (i.e., $a = a_0, b = b_0$), the total number of function evaluations N upon entering the loop is $N = 2 + k$.

Let n be the total number of iterations executed by the `for` loop. The total number of function evaluations is $N = 2 + n$. For instance, if $n = 5$ iterations occur, the total count is $N = 2 + 5 = 7$ function evaluations.

The definition used for the function's output N, which expresses the total number of evaluations of $f(x)$ in the `for` loop, does not correspond to the theoretical criterion $(b_k - a_k)/\ell \leq F_n$ even if the same index is used.

Requirement 1 - Parametric search range ℓ

We determine the range of the tolerance ℓ utilized to calculate the number of objective function evaluations for each of the three functions, considering the observed relationship between the number of while loop iterations and the corresponding number of function evaluations.

The following diagrams illustrate the determination of the tolerance ℓ values for each function:

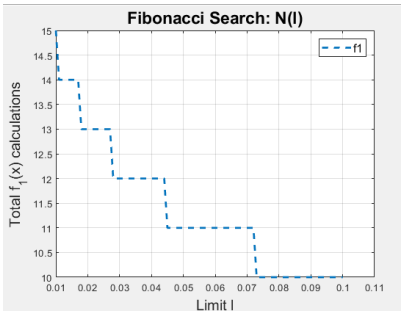

Figure 19: Plot of f_1

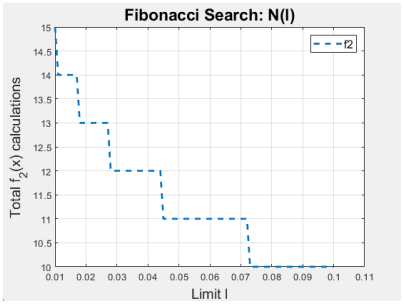

Figure 20: Plot of f_2

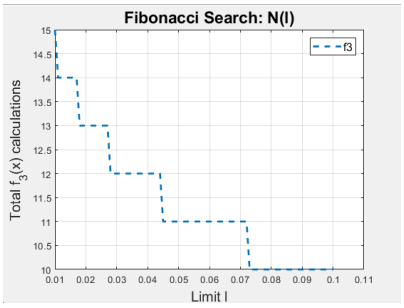

Figure 21: Plot of f_3

Observations on Algorithmic Efficiency and Precision

The Influence of Tolerance ℓ on Computational Cost

It is observed that an increase in the stopping tolerance ℓ results in a reduction in the total number of objective function evaluations required by the optimization algorithm. This inverse relationship is fundamental: a larger value for ℓ dictates a wider final search interval, thereby requiring lower positional precision for the minimizer, and consequently, fewer iterations and function evaluations are needed for the convergence criterion to be met.

Convergence Consistency

For all objective functions analyzed, the algorithm exhibits an identical convergence profile. The total number of iterations required is solely dependent on the specified precision tolerance ℓ and the dimensions of the initial search interval $[a, b]$. This indicates the algorithm's robustness, where its performance dependency lies only on the input parameters ℓ and $[a, b]$, independent of the specific function form (assuming unimodality).

The phenomenon where the graphical representation of the final interval width displays plateaus suggests that highly marginal perturbations in the tolerance ℓ may not induce a practical or measurable change in the resulting final interval width.

To achieve greater visual fidelity and resolution in the graphical representations, two methodological adjustments can be employed:

- Increased Sampling: Incorporate a larger number of data points (i.e., calculate the final interval for more values of ℓ).
- Axis Range Reduction: Decrease the display range on the horizontal axis to magnify the observed variations over a smaller parameter space.

Requirement 2 - Plotting Subintervals $[a_k, b_k]$ for various values of tolerance ℓ

For different values of the tolerance ℓ, we are asked to calculate and plot the lower and upper bounds, a_k and b_k, of the subintervals for the given functions. A common plot should be generated for each function separately, showing the two bounds across different values of ℓ.

The resulting diagrams for the bounds of every function $f(x)$ are shown below:

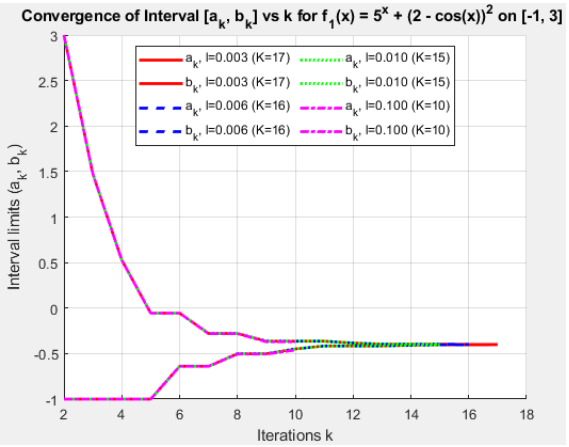

Figure 22: Plot of f_1

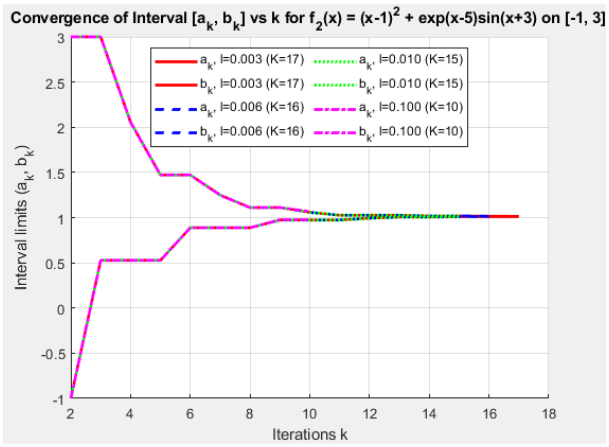

Figure 23: Plot of f_2

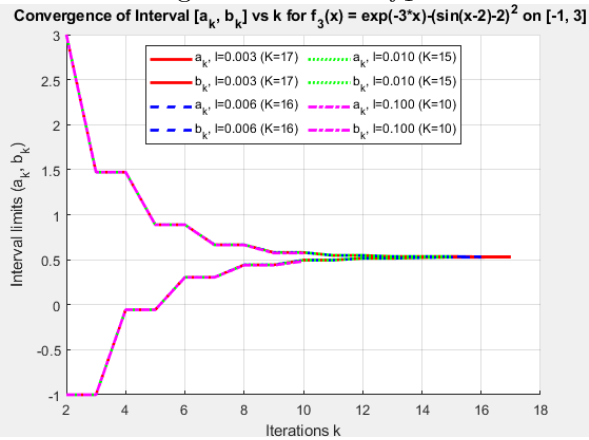

Figure 24: Plot of f_3

Relationship between Tolerance and Iterations

The two bounds converge, a result that is inherently expected. We observe that as the stopping tolerance ℓ is decreased, the total number of function evaluations, N, and iterations, n, required are increased. This relationship is due to the smaller ℓ demanding a significantly narrower final interval, which equates to higher localization precision for the minimizer, thus necessitating more computational steps.

The convergence profile remains consistent for every function. Only when the tolerance ℓ is reduced does the algorithm execute a few additional iterations. The key factor is that the given objective function $f(x)$ is checked solely against the desired precision to satisfy the termination criterion, and this process is robust against small parameter variations.

A critical observation is the systematic interval update: at each iteration k, either the upper bound (b_k) is decreased, or the lower bound (a_k) is increased.A simultaneous change in both bounds is never observed.

Concluding Comparative Remarks - Computational Efficiency

The Fibonacci Search Method possesses the distinct advantage of requiring only one new objective function evaluation after the initial two, which is the same superior efficiency demonstrated by the Golden Section Search Method. In fact, in practical application, the Fibonacci method often slightly outperforms the Golden Section Search by achieving the required reduction in fewer steps (iterations).

A primary prerequisite and disadvantage of the Fibonacci method, however, is the need to specify in advance the total number of iterations n required for a given precision. The basic inherent weakness is the necessity to calculate and store the entire sequence of Fibonacci numbers up to F_n. The recursive nature of the Fibonacci sequence makes its calculation a non-trivial programming task, necessitating an efficient computational technique to generate each number for calculation.

The Bisection Method with Derivative Utilization

The Bisection Method with Derivative Utilization (or Gradient-Based Bisection Search) is a gradient-based optimization algorithm designed to locate the unconstrained minimizer x^* of a differentiable function $f(x)$ over an initial search domain $[a, b]$. At each iteration k, the method iteratively generates a new, progressively smaller subinterval $[a_k, b_k]$ which is guaranteed to contain the minimizer x^*.

Termination Criterion

The iterative process is terminated at the n-th step when the final interval length satisfies the precision requirement: $b_n - a_n \leq \ell$, where ℓ is a predefined final tolerance. This precision constraint is equivalent to achieving a sufficient relative reduction factor:

$$\left(\frac{1}{2}\right)^n \leq \frac{\ell}{b-a}$$

The integer n represents the total number of iterations required to satisfy this inequality.

The prerequisite for applying this method is the analytical knowledge and computability of the first derivative $\frac{df(x)}{dx}$ within the domain under study, which is used to determine the subsequent interval $[a_{k+1}, b_{k+1}]$ at each step.

Iterative Step (While Loop)

The process involves the following sequence of operations at each iteration k:

- We compute the midpoint $x_k = (a_k + b_k)/2$ of the current interval $[a_k, b_k]$.
- We then evaluate the gradient at this midpoint, $\left.\frac{df(x)}{dx}\right|_{x=x_k}$.
- **Decision Rule (Based on Gradient Sign):**
  - If $\left.\frac{df(x)}{dx}\right|_{x=x_k} < 0$: The minimizer is in the right half-interval. Thus, $a_{k+1} = x_k$ and $b_{k+1} = b_k$.
  - If $\left.\frac{df(x)}{dx}\right|_{x=x_k} > 0$: The minimizer is in the left half-interval. Thus, $a_{k+1} = a_k$ and $b_{k+1} = x_k$.
  - If $\left.\frac{df(x)}{dx}\right|_{x=x_k} = 0$: The minimizer has been exactly located at x_k, and the `while` loop terminates immediately: $a_{k+1} = x_k$ and $b_{k+1} = x_k$.

Computational Cost

It is important to note that the objective function $f(x)$ itself is **not** explicitly evaluated within the iterative steps. The computational cost is exclusively measured by the number of evaluations of the **derivative** $f'(x)$. Therefore, if k expresses the iteration count from the moment the loop commences (assuming $k = 0$ at the start, $a = a_0, b = b_0$), we have k total derivative evaluations.

The output parameter n directly represents the total number of iterations executed by the `while` loop. For example, if $n = 3$ iterations occur, this implies 3 total evaluations of the derivative function $f'(x)$. The function returns the iteration count n.

Requirement 1 - Parametric search range ℓ

We determine the range of the tolerance ℓ utilized to calculate the number of the derivative of the objective function evaluations for each of the three functions, considering the observed relationship between the number of while loop iterations and the corresponding number of function evaluations.

The Effect of Tolerance ℓ on Computational Cost

It is consistently observed that an increase in the stopping tolerance ℓ results in a reduction in the total number of objective function evaluations required (N). This is because a larger ℓ dictates a wider final interval width, which translates to a lower required precision for the minimizer's location, consequently requiring fewer iterations and function evaluations.

The following diagrams illustrate the determination of the tolerance ℓ values for each function:

Convergence Invariance

For all objective functions analyzed, the algorithm exhibits an identical convergence profile. The total number of iterations required is solely dependent on the specified precision tolerance ℓ and the extent of the initial search interval $[a, b]$.

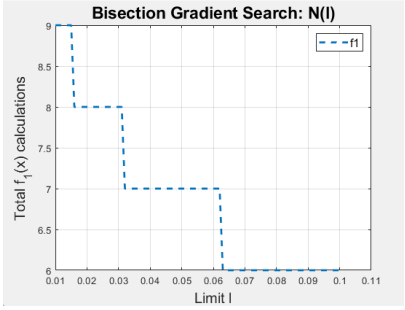

Figure 25: Plot of f_1

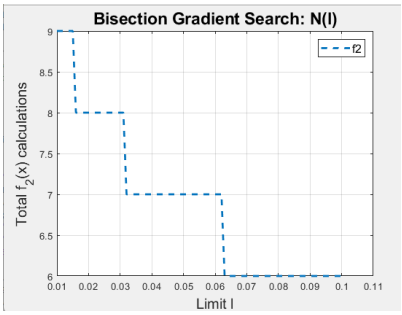

Figure 26: Plot of f_2

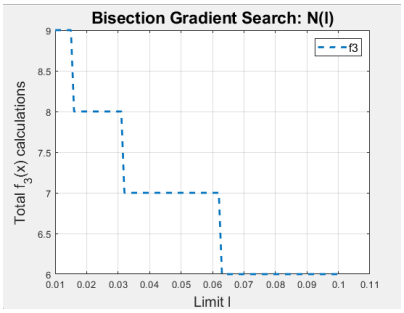

Figure 27: Plot of f_3

The phenomenon where the graphical representation of the final interval width displays plateaus suggests that highly marginal changes in the tolerance ℓ may not induce a practical or measurable change in the resulting final interval width.

Requirement 2 - Plotting Subintervals $[a_k, b_k]$ for various values of tolerance ℓ

For different values of the tolerance ℓ, we are asked to calculate and plot the lower and upper bounds, a_k and b_k, of the subintervals for the given functions. A common plot should be generated for each function separately, showing the two bounds across different values of ℓ.

Relationship between Tolerance ℓ and Computational Effort

It is observed that as the stopping tolerance ℓ is decreased, a proportionally greater number of total function evaluations/iterations n is required. This is an expected consequence, as a smaller ℓ necessitates a significantly narrower final interval width, demanding higher precision in the minimizer's location, and thus increasing the computational cost.

The convergence profile remains consistent for every function. Changes in the tolerance ℓ only affect the point of termination, causing the algorithm to execute a few additional iterations when ℓ is reduced. This robustness stems from the fact that the given objective function $f(x)$ is checked solely against the desired precision to satisfy the termination criterion.

A critical structural property is evident: at each iteration k, the interval boundaries are updated by either decreasing the upper bound (b_k) or increasing the lower bound (a_k). A simultaneous alteration of both bounds is never observed within a single step.

The resulting diagrams for the bounds of every function $f(x)$ are shown below:

Concluding Comparative Remarks on the Bisection Method with Derivative Utilization

Advantages

The Bisection Method with Derivatives is regarded as the most technologically sophisticated optimization method examined in this study for locating the minimum. Similar to the Fibonacci Search Method, it provides the significant advantage of **a priori knowledge** of the required number of iterations n for a specified precision ℓ. This characteristic is particularly valuable in cases demanding high precision where the computational time budget (or step count) needs to be known beforehand.

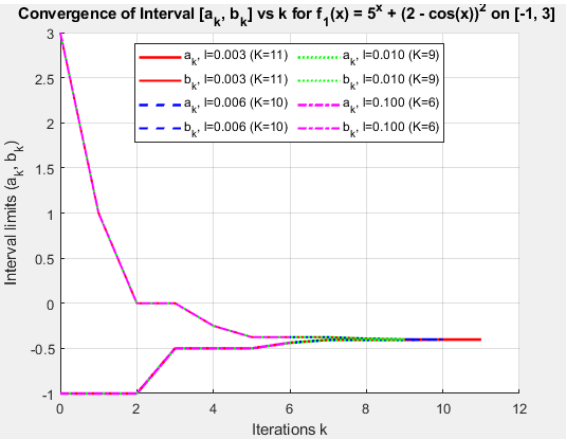

Figure 28: Plot of f_1

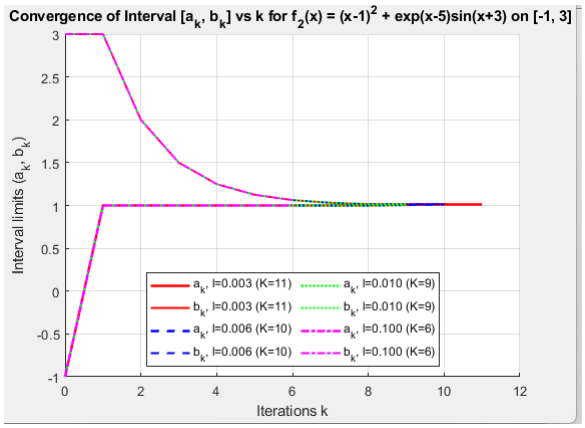

Figure 29: Plot of f_2

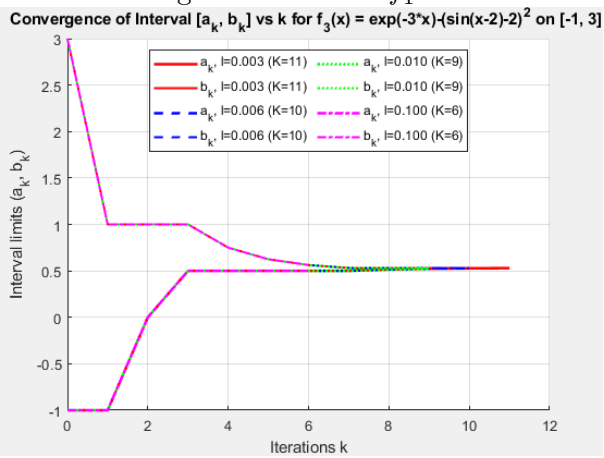

Figure 30: Plot of f_3

Disadvantages

The major drawback of this method is the requirement that the function must be **differentiable** throughout the entire studied domain - initial interval. Furthermore, it often implicitly assumes that the derivative can be analytically (or numerically) computed at every point within this interval. Since functions that satisfy these strict differentiability and computability conditions are not universally encountered in optimization problems, the applicability of the Bisection Method with Derivatives is inherently limited compared to non-gradient-based search methods.

General Synthesis and Comparative Analysis

Overall Efficacy of Optimization Methods

The comprehensive analysis confirms that all examined optimization algorithms are capable of efficiently converging towards the true optimal solutions in practical applications. The fastest converging algorithm, however, is the one that utilizes gradient information (the Bisection Method with Derivatives), which, in a generalized context, corresponds to the class of more sophisticated methods often computed at the end of code sections as an 'extra' utility.

Decision Matrix for Non-Gradient-Based Search

A critical decision point arises when the utilization of the first derivative is not feasible or desirable. This may be due to several factors:

- Analytical Complexity: The derivative $\frac{df(x)}{dx}$ may be excessively complex or cumbersome to derive analytically.
- Computational Cost: The evaluation of the derivative might be computationally prohibitive.
- Non-Differentiability: The objective function $f(x)$ may not be differentiable (or smooth) across the entire search domain $[a, b]$.

In such scenarios, the most suitable alternatives are the Fibonacci Search Method and the Golden Section Search Method.The Bisection Method without utilization of derivatives is excluded due to its higher inefficiency among all the other methods. The selection between Fibonacci and Golden Section Method depends on a careful consideration of two main factors:

1. **Spatial Complexity (Pre-Calculation Requirement):** The Fibonacci method necessitates the pre-calculation and storage of the entire sequence of Fibonacci numbers up to the required iteration F_n. This introduces a constraint on memory and temporal complexity, particularly if the required number of iterations n (and thus the size of F_n) is high.
2. **Precision Requirements and Iteration Count:** If the required precision is exceptionally high, leading to a substantial number of iterations n, the magnitude of the required Fibonacci numbers (F_n) will also be very large.

Final Recommendation on Choice of Method

While the Fibonacci method offers a marginal theoretical advantage in terms of the rate of interval reduction compared to the Golden Section method, its practical implementation is often hindered by the necessity to compute and store potentially very large Fibonacci numbers. This recursion-based calculation introduces computational delay and complexity.

Conclusion: If computational efficiency is paramount but derivative use is restricted, the Golden Section Search Method is the preferred choice over the Fibonacci Search Method when the anticipated iteration count n is high. This is because the Golden Section method operates using a single, constant factor ($\gamma = \phi^{-1}$), thereby avoiding the substantial overhead associated with the recursive calculation and storage of the large sequence of Fibonacci numbers required by its counterpart.

Code Reference and Instructions for execution

The uploaded file contains two function files per method and two or three related to every requirement (1, 2, 3) mentioned in every section.In order to build & run they must be on the same path.
The .m files are named (1_ 1),(1_ 2) etc. refer to a specific requirement for each method.