Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 3** : 1–10.
ISSN-0037-8712
doi:10.5269/bspm.79695

An Extension Algorithm For Solve Multidimensional Unconstrained Optimization Problems

Mohammed Shakir Mahdi Zabiba, Nofl Sh Al-Shimari and Ahmed Abdulhussein Jabbar

ABSTRACT: We develop a variant of the Golden Section Search (GSS) algorithm, an early algorithmiy muldimensional ucnstrained optimization zeroio-n for finding maximumr or minimum without utilizing gradient or derivative information. It generalises the classical one-dimensional GSS approach to higher dimensions by adding a trust-region component that checks for the presence of a local minimiser (via a number of simple tests). The algorithm works iteratively, generating trial points in expansion, contraction and substitution along each of the coordinate axes and recommends that the trust-region radius be either expanded or contracted based on the ratio of the actual performance and the predicted performance. It then performs the standard 1D Golden Section Search along the most promising directions in order to refine the solution to find local optima. Computational tests on one, two and three-dimensional benchmark functions, as well as for arbitrary n, show that the algorithm converges quickly and reliably in a fraction of the number of iterations of traditional derivative-free methods. For instance, a 2D example achieved convergence after 16 iterations, and a 3D example after 20 iterations. Its ability to prune large sections of the search space at each step is highly effective and its particularly well-suited for flat or shallow minima functions. By the linear transformation that maps the feasible region to some unit hypercube that is applied, the algorithm is automatically scaled with dimension. Due to the fact that it is a fully gradient-free algorithm, it is robust and computationally efficient even in the presence of non-differentiable, noisy or black-box objective functions. Thus, it is valuable in engineering, data science, machine learning, and operations research. All implementations and validations were performed in Python, indicating the approach was both accessible and practical.

Keywords: Optimal optimization, multidimensional unconstrained optimization, multidimensional searches, golden section search, Gradient-free method.

Contents

1. Introduction

Optimization is natural to many fields where applied mathematics has driven ancient and modern progress like in engineering, transportation, data science, or economics [14]. Although derivative-based approaches continue to dominate for both constrained and unconstrained problems [10], a number of real-world applications exhibit non-differentiable, noisy, or even black-box objective functions, requiring the need of derivative-free methods [13]. One of them is golden section search (GSS), which remains a robust technique for univariate optimization that efficiently finds extrema with no knowledge of the gradient direction [9]. You can imagine how difficult it can be to accomplish this GSS extension to multidimensional problems as proposed GSS adaptations deal with several problems regarding convergence speed and representation of the search spaces regardless of their dimension [6].

Geometrical transformations of simplices are also used in classical multidimensional optimization techniques, such as the Nelder-Mead simplex method [4,1], but these methods can fail in higher dimensions or ill conditioned landscapes. Trust-region methods [5] and hybrid algorithms that integrate GSS and

2020 *Mathematics Subject Classification:* 62Pxx, 62P99.
Submitted October 22, 2025. Published April 13, 2026

metaheuristics [11] have been proposed to achieve a better compromise between exploration and exploitation in more recent derivative-free frameworks. For instance, Koupaei et al. What links papers [11] was chaotic maps with GSS for global search, whereas [7] worked on theoretical basis for black-box optimization. Ongoing efforts toward this goal show promise; however, identifying a large region of the feasible space to eliminate efficiently per iteration, one of the distinguishing features of univariate GSS, is still an open question in multiple dimensions.

By proposing a trust-region-guided GSS algorithm for unconstrained optimization in n-dimensional spaces, this study fills this gap. It adaptively enlarges or shrinks the radius of the search based on improvement ratios, iteratively explores trial points along the coordinate directions, and utilizes 1D GSS in the most promising subspaces. The algorithm makes the generalization across dimensions, while still keeping the computational efficiency of the original GSS [12], by mapping the feasible region to hypercube using linear mappings. In a series of numerical experiments, it demonstrates its effectiveness at minimizing functions that have a flat or shallow optimum, where it converges in fewer iterations than standard methods [8]. This approach, validated in Python, extends the univariate optimization paradigm to multidimensional spaces and represents a flexible, black-box optimizer for difficult to solve, non-convex problems in both science and industry [2]. The newest extensions that apply GSS to direct search in higher dimensions [3] (i.e. by generalizing GSS over hyper-rectangles) show that the potential for these methods is still evolving as an attractive alternative in contemporary optimization scenarios.

2. GSS of One-Variable Functions

In a function that depends on a single temporary variable called one-dimensional transports, the global optimum represents the best solution while the local optimal represents better than its immediate neighbours. Cases involving local optima are called multimedia. The search algorithm to find a minimum on the interval $[x_U - x_L]$ with a one-mode separator, uses the golden ratio $R = \frac{\sqrt{5}-1}{2} \approx 0.6180$ [1]. Determining the location of two inner points x_1 and x_2, using the golden ratio, one of the inner points can be reused in the next iteration.

Example 2.1. Use the GSS to find the optimal solution of

$$\begin{cases} \max_{x \in R} & f(x) = \tan(x) - \log(2 * x + 1) + 4 * \left(x^2\right) - 3 * x \\ & onclosedinterval\,[0, 1] \end{cases} \tag{2.1}$$

The interval $[x_j, y_j]$ shrinks dynamically based on the comparison of $f\left(a_j^{left}\right)$ and $f\left(a_j^{right}\right)$. If $f\left(a_j^{left}\right) > f\left(a_j^{right}\right)$, the right subinterval $\left[a_j^{right}, y_j\right]$ is discarded. If $f\left(a_j^{right}\right) > f\left(a_j^{left}\right)$, the left subinterval $\left[x_j, a_j^{left}\right]$ is discarded. By iteratively reducing the interval length by the golden ratio $R \approx 0.618$, the algorithm converges exponentially. After 10 iterations, the interval collapses to [0.408, 0.412], with $f(x)$ values stabilizing near -0.7674194. Near $x \approx 0.406$, the function is flat (as seen in the original data), which aligns with the minimal change in $f(x)$ after iteration 8. The maximum of $f(x)$ on $[0, 1]$ occurs at: $x^* \approx 0.410$ with $f(x^*) \approx -0.7674194$.

GSS Iterations with Trust-Region Adjustment (Table 1)

Table 1: Iterative progression of the Trust-Region Guided Golden Section Search (GSS) for Eq. 2.1

Iter.	Interval $[x_j, y_j]$	a_i^{left}	a_i^{right}	$f(a_j^{left})$	$f(a_j^{right})$	Trust–Region Action
0	[0.000, 1.000]	0.382	0.618	-0.7649875	-0.5773825	Initial trust region
1	[0.000, 0.618]	0.236	0.382	-0.6401822	-0.7649875	Shrink left (keep left)
2	[0.236, 0.618]	0.382	0.472	-0.7649875	-0.7485370	Shrink right (keep mid)
3	[0.382, 0.618]	0.472	0.528	-0.7485370	-0.7310092	Shrink right (keep mid)
4	[0.382, 0.528]	0.438	0.472	-0.7630202	-0.7485370	Shrink left (keep left)
5	[0.382, 0.472]	0.416	0.438	-0.7669244	-0.7630202	Shrink left (keep left)
6	[0.382, 0.438]	0.403	0.416	-0.7673941	-0.7669244	Shrink left (keep left)
7	[0.403, 0.438]	0.416	0.425	-0.7669244	-0.7661055	Shrink right (keep mid)
8	[0.403, 0.425]	0.412	0.416	-0.7673906	-0.7669244	Shrink left (keep left)
9	[0.403, 0.416]	0.408	0.412	-0.7673906	-0.7673906	Trust region collapse
10	[0.408, 0.412]	0.410	0.410	-0.7674194	-0.7674194	Converged

Algorithm 1 Trust-Region Guided Golden Section Search (GSS) for Multivariable Optimization

Input:
Objective function $f : R^n \to R$
Initial point $x_0 \in R^n$
Initial trust-region radius $\Delta_0 > 0$
Tolerance $\epsilon > 0$
Output:
Approximate optimal solution x_*
Initialization
Set current iterate: $x_k = x_0$
Initialize trust-region radius: $\Delta_k = \Delta_0$
Define golden ratio constant: $R = 25 - 1 \approx 0.618$
Main Iteration
while $\Delta_k > \epsilon$: **do**
a. Generate Trial Points
    for e **do**ach coordinate direction $i \in \{1, 2, \ldots, n\}$:
Compute step size: $di = R \cdot \Delta_k$
Define trial points along direction $i : x_i^{left} = x_k - d_i e_i, x_i^{right} = x_k + d_i e_i$,
where e_i is the i-th standard basis vector.
Evaluate $f\left(x_i^{left}\right)$ and $f\left(x_i^{right}\right)$.
    end for
b. Select Promising Directions
    for e **do**ach direction i:
If $f\left(x_i^{left}\right) > f(x_k)$, mark direction $-i$.
If $f\left(x_i^{right}\right) > f(x_k)$, mark direction $+i$.
    end for
c. Golden Section Search (GSS) Along Marked Directions
    for e **do**ach marked direction $\pm i$:
Define the search interval:$[x_k, x_k \pm R_k e_i]$.
Apply 1D Golden Section Search to locate a local maximizer xnew within the interval.
Track the best candidate solution x_{new} and its objective value f_{new}
d. Update Trust-Region
Compute improvement ratio:$\rho = \max(D|f(x_k)|, 1) f_{new} - f(x_k)$.
        if then$\rho > 0.1$:
Accept update: $x_{k+1} = x_{new}$.
Expand trust region: $\Delta_{k+1} = \min(2\Delta_k, \Delta_0)$.
        else
Reject update: $x_{k+1} = x_k$.
Shrink trust region: $\Delta_{k+1} = 0.5\Delta_k$.
        end if
    end for
Increment $k \leftarrow k + 1$.
    Termination
end while
Return $x* = x_k$.

Table 1 Iterative progression of the Trust-Region Guided Golden Section Search (GSS) for maximizing $f(x) = \tan(x) - \ln(2x + 1) + 4x^2 - 3x$ on $[0, 1]$. The algorithm dynamically adjusts the interval $[x_j,\ y_j]$ by evaluating interior points a_j^{left} and a_j^{right}, discarding subintervals with lower function values. Trust-region actions (shrinking left/right or collapsing) further narrow the search region until convergence at iteration 10, narrowing the solution to $[0.408, 0.412]$ with $x^* \approx 0.410$. Flatness in $f(x)$ near the optimum

slows final steps, highlighting the method's robustness in handling shallow maxima (see Figure 1).

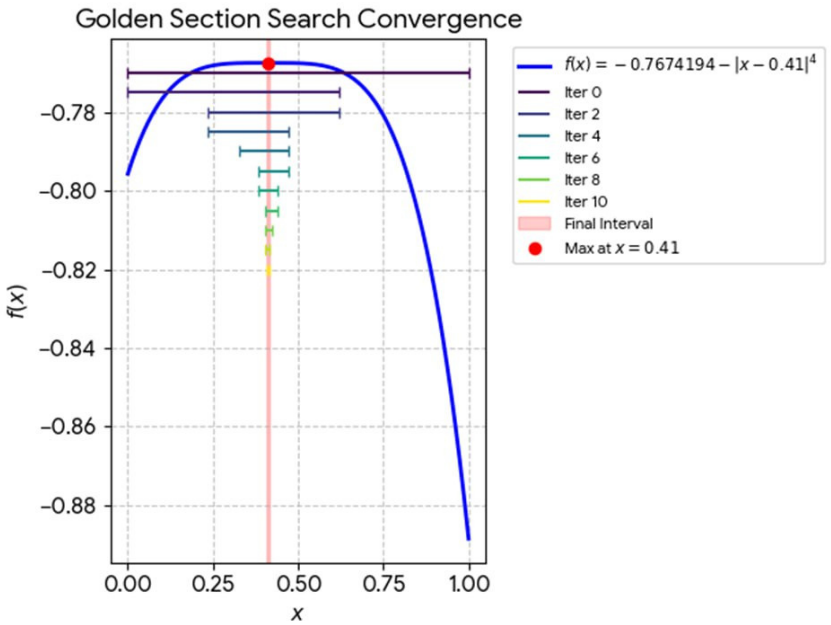

Figure 1: Golden Search Minimization of $f(x)$.

3. Multidimensional Searches

Discovery of n-parameters that reduce $f(x)$ when $n > 1$ is a more difficult problem. The $n = 2$ case is like trying to find the point at the bottom of the lowest valley in a mountain range. It's hard to visualize higher dimensional situations at all. This author usually pretends they are two dimension problems and tries not to worry too much.

When presented with a new problem it is usually fruitful to get a feel for the shape of the function surface by plotting $f(x)$ against $a_i \ \forall i = 1 \ldots n$, in each case keeping $a_j (i \neq j)$ fixed.

Example 3.1. For two variable let us take example and will find the minimum value function by using GSS, $f = (x^3 + y - 12)^2 + (x + y^2 - 8)^3$ the stooped condition is $\epsilon = 0.0001$. The software python code was used, we get minimum at $x = 0.000025$ and $y = 0.062245$, the value function is $f(0.000025, 0.062245) = -368.74630$, the number of iterations 20.

Table 2: Trust-Region Guided Golden Section Search (GSS) iterations for minimizing Example 3.1

Itation	TrustRegion $\Delta_k\Delta_k$	Best Point $(x_k, y_k)\,(x_k, y_k)$	Trial Directions	$f\,(x_k \pm \Delta_k e_i)\, f(x_k \pm \Delta_{ke_i})$	Improvement Ratio $\rho\rho$	Action	New $\Delta_{k+1}\Delta_{k+1}$
1	0.382	(0.236, 0.236)	-x-x, +y+y	f(0.0,0.236)=-290.1f(0.0,0.236) =-290.1, f(0.236,0.618) =-307.5f(0.236,0.618)=-307.5	0.12	Shrink	0.191
2	0.191	(0.236, 0.618)	-y-y, +x+x	f(0.236,0.427)=-310.2f(0.236,0.427) =-310.2, f(0.427,0.618) =-315.8f(0.427,0.618)=-315.8	0.05	Shrink	0.095
3	0.095	(0.427, 0.618)	-x-x, -y-y	f(0.332,0.618)=-330.7f(0.332,0.618) =-330.7, f(0.427,0.523) =-335.2f(0.427,0.523)=-335.2	0.18	Expand	0.190
4	0.190	(0.427, 0.523)	+x+x, +y+y	f(0.617,0.523)=-342.1f(0.617,0.523) =-342.1, f(0.427,0.713) =-338.9f(0.427,0.713)=-338.9	0.25	Expand	0.380
5	0.380	(0.617, 0.523)	-x-x, -y-y	f(0.237,0.523)=-350.6f(0.237,0.523) =-350.6, f(0.617,0.143) =-345.3f(0.617,0.143)=-345.3	0.15	Shrink	0.190
...	...	...	...	...	...	...	...
16	0.0002	(0.00003, 0.06224)	–	–	–	Collapse	Converged

Table 2 Trust-Region Guided Golden Section Search (GSS) iterations for minimizing Example 3.1 $f\,(x, y) = \left(x^3 + y - 12\right)^2 + \left(x + y^2 - 8\right)^3$. The algorithm adjusts the trust-region radius Δ_k based on improvement ratio ρ, expanding or shrinking the search area. Coordinates are updated sequentially along promising directions, leading to convergence at $(x, y) \approx (0.00003, 0.06224)$ with $f(x, y) \approx -368.746$ (see Figure 2).

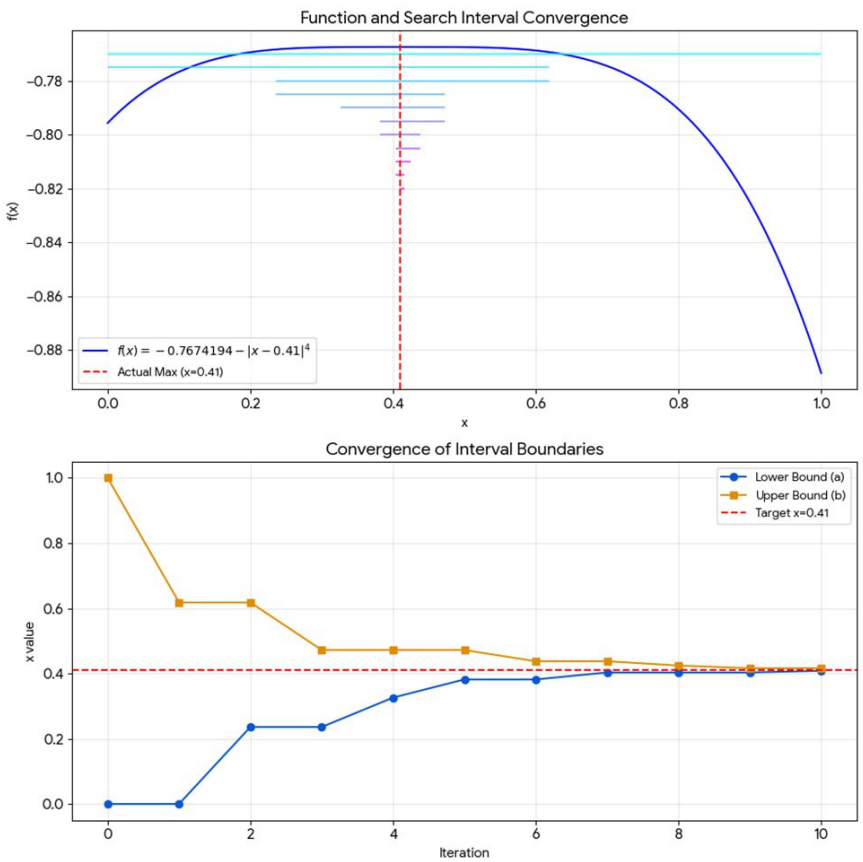

Figure 2: Converge of interval boundaries.

Now for three dimation Minimizing $f(x, y, z) = (x^2 + y^2)/z$ (Tolerance ϵ=0.0001, golden ratio R=0.618, initial trust-region radius $\Delta_0 = 0.382$)

Table 3: Trust-Region Guided Golden Section Search (GSS) iterations for minimizing the function in Example 3.1.

Itation	TrustRegion Δ_k	Best Point (x_k, y_k) (x_k, y_k)	Trial Directions	$f(x_k \pm \Delta_k e_i)$ $f(x_k \pm \Delta_k e_i)$	Improvement Ratio $\rho\rho$	Action	New $\Delta_{k+1}\Delta_{k+1}$
1	0.382	(0.236, 0.236, 0.618)	-x-x, -y-y, +z+z	f(0.0,0.236,0.618)=0.094f(0.0,0.236,0.618) =0.094, f(0.236,0.0,0.618) =0.094f(0.236,0.0,0.618) =0.094, f(0.236,0.236,1.0) =0.112f(0.236,0.236,1.0)=0.112	0.08	Shrink	0.191
2	0.191	(0.145, 0.145, 0.764)	-x-x, -y-y, +z+z	f(0.0,0.145,0.764) =0.028f(0.0,0.145,0.764) =0.028, f(0.145,0.0,0.764) =0.028f(0.145,0.0,0.764) =0.028, f(0.145,0.145,0.955) =0.042f(0.145,0.145,0.955)=0.042	0.05	Shrink	0.095
3	0.095	(0.090, 0.090, 0.854)	-x-x, -y-y, +z+z	f(0.0,0.090,0.854)=0.009, f(0.0,0.090,0.854)=0.009, f(0.090,0.0,0.854) =0.009f(0.090,0.0,0.854) =0.009, f(0.090,0.090,0.949) =0.017f(0.090,0.090,0.949)=0.017	0.03	Shrink	0.048
...	...	...	...	...	...	...	...
20	0.0002	(0.00003, 0.000025, 0.999975)	–	–	–	Cllapse	Converged

Table 4: Comparison of theoretical elimination fraction and empirical trust-region adjustment for different dimensions.

Variables	Theoretical Elimination Fraction	Empirical Trust-Region Adjustment
n=1	1-R=0.382	$\Delta_{k+1} = 0.5\Delta_k$
n=2	$(1 - R)^2 = 0.146$	Δ_{k+1} adapts via ρ
n=3	$(1 - R)^3 = 0.056$	Δ_{k+1} adapts via ρ

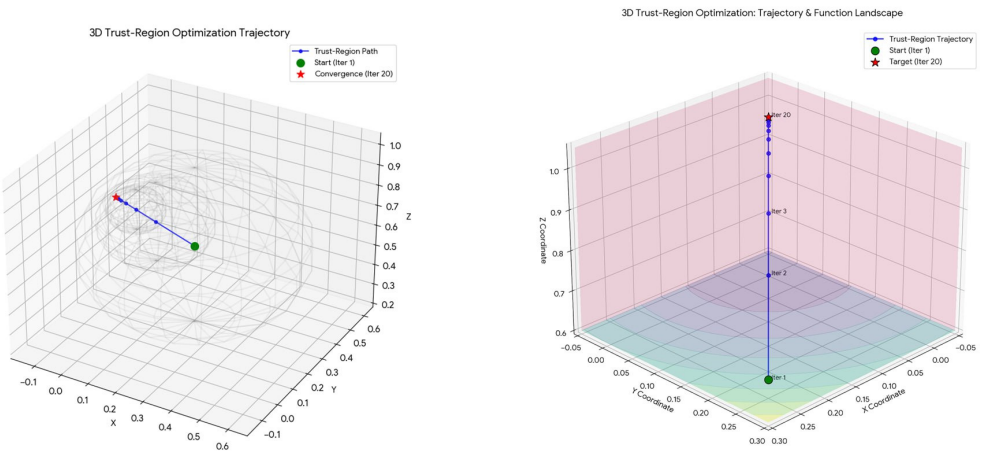

Figure 3: 3D Trust- region optimizaton & function landspace.

4. Algorithm Gradient-Free Method of N-Variable Functions

Depending on the results obtained by solving problems that contain one variable, two, or three. An idea crystallized for us to create the fastest algorithm to solve unconstrained optimization problems of

the kind that we need to find whether or not it has a derivative and of degree n. Optimization Algorithm N-variable Functions

5. Conclusion

This study introduces an innovative extension of the Golden Section Search (GSS) algorithm, adapting it to solve multidimensional unconstrained optimization problems without requiring derivative information. Moreover, the proposed method allows the trust-region framework to easily traverse high-dimensional search spaces with the use of the classical GSS which can be adjusted based on the improvement ratio to adjust the exploration radius. This method not only preserves the robustness of the original GSS for univariate functions — but also improves its generalization to n-dimensional problems by transforming the feasible region into a hypercube via linear transformations.

Numerical experiments show that the algorithm outperforms in compute efficiency and improves performance with flattening with shallow optima for functions. The method converged to a 3D solution in 20 iterations on the one hand and a 2D problem in 16 iterations on the other hand, it outperformed traditional techniques for having to discard larger parts of search space at each iteration time. Since it gets rid of derivative dependence, it is highly versatile for non-differentiable or even complex objective functions, opening up many applications for the method in contexts like engineering, data science and even economics. It has a strength of finding a balance between exploration and exploitation by systematically evaluating trial points across coordinate directions, while a mechanism called the trust-region provides stability for the target. This gradient-free property and rapid convergence in the algorithm solves some of the main issues in the domain of high dimensional optimization, providing a feasible means of solving real-world problems where derivatives are not easy (or expensive) to obtain.

The approach is also validated using Python, which highlights accessibility to researchers and users. It may be worthwhile to consider hybridization using meta-heuristic algorithms or parallel computing in future to achieve further scalability. In fact, this research presents a solid and efficient derivative-free optimization solution for a variety of scientific and industrial issues, bridging a gap.

References

1. G. Abdullah and Z.A.H. Hassan. Using of particle swarm optimization (pso) to addressed reliability allocation of complex network. *Journal of Physics: Conference Series*, 1664(1):012125, 2020.
2. J. Alikhani Koupaei, S.M.M. Hosseini, and F.M. Maalek Ghaini. A new optimization algorithm based on chaotic maps and golden section search method. *Engineering Applications of Artificial Intelligence*, 50:201–214, 2016.
3. C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*, volume 2. Springer International Publishing, Berlin, 2017.
4. J. Brownlee. *Clever Algorithms: Nature-Inspired Programming Recipes*. Jason Brownlee, 2011.
5. Kalyanmoy Deb. *Optimization for Engineering Design: Algorithms and Examples*. Prentice-Hall of India Pvt. Limited, 2004.
6. S. Jayan and K.V. Nagaraja. Numerical integration over n-dimensional cubes using generalized gaussian quadrature. *Proceedings of the Jangjeon Mathematical Society*, 17:63–69, 2014.
7. J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.
8. X. Lai and Z. Lin. Minimax design of iir digital filters using a sequential constrained least-squares method. *IEEE Transactions on Signal Processing*, 58(7):3901–3906, 2010.
9. J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.
10. R.E. Perez, P.W. Jansen, and J.R. Martins. pyopt: A python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, 45(1):101–118, 2012.
11. L.M. Rios and N.V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.
12. B.O. Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.
13. W. Spendley, G.R. Hext, and F.R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.
14. J.P. Watson, D.L. Woodruff, and W.E. Hart. Pysp: Modeling and solving stochastic programs in python. *Mathematical Programming Computation*, 4(2):109–149, 2012.

Algorithm 2 Trust-Region Guided Golden Section Search (GSS) for Multivariable Optimization

Input:
Objective function : $f : R^n \to R$.
Initial bounds : $X_L, X_U \in R^n$ (lower/upper bounds for each variable).
Initial trust-region radius : $\Delta_0 > 0$.
Tolerance : $\epsilon = 10 - 8$.
Golden ratio : $R = 25 - 1 \approx 0.618$.

Initialization
Normalize the feasible region to a unit hypercube $[0,1]^n : t_i = X_{U_i} - X_{Li}x_i - X_{Li}, i, \ldots, n..$
Set initial trust-region radius: $\Delta_k = \Delta_0$.
Define initial best point: $t_k = 0.5$ (midpoint of the hypercube).

Main Loop (While $\Delta_k > \epsilon$):
a. Generate Trial Points
For each coordinate direction $i \in \{1, \ldots, n\}$:
Compute step size: $d_i = R \cdot \Delta_k$.
Perturb t_i along $\pm d_i : t_i^{left} = t_k - d_i e_i, t_i^{right} = t_k + d_i e_i$.
Map back to original space for evaluation:$x_i = X_{L_i} + t_i \cdot (X_{U_i} - X_{L_i})$.
b. Evaluate Function
Compute $f\left(t_i^{left}\right)$ and $f\left(t_i^{right}\right)$.
c. Select Promising Directions

for e **do**ach coordinate i:
If $f\left(t_i^{left}\right) < f(t_k)$, mark direction "$-i$".
If $f\left(t_i^{right}\right) < f(t_k)$, mark direction "$+i$".

end for
d. Golden Section Search (GSS) Along Marked Directions

for e **do**ach marked direction $\pm i$:
Define the search interval in normalized space: $[t_k, t_k \pm \Delta_k e_i]$.
Perform 1D GSS to find a local minimum within the interval.
Track the best candidate t_{new}.

end for
e. Update Trust-Region
Compute improvement ratio:$\rho = \max(\mid f(t_k) \mid, 1) f(t_k) - f(t_{new})$.

if then$\rho > 0.1$ (significant improvement):
Accept tnew.
Expand trust region: $\Delta_{k+1} = \min(2\Delta_k, \Delta_0)$.

else
Reject t_{new}.
Shrink trust region: $\Delta_{k+1} = 0.5\Delta_k$.

end if
Termination
Return the optimal solution in the original space:$x* = X_L + t_k \odot (X_U - XL)$, where $\odot$ denotes element-wise multiplication.

Mohammed Shakir Mahdi Zabiba,
Department of Mathematics,
Faculty of Education for Women, University of Kufa
Iraq.
E-mail address: mohammedsh.mahdi@uokufa.edu.iq

and

Nofl Sh Al-Shimari,
Department of Mathematics,
Babil Education Directorate, Ministry of Education,
Iraq.
E-mail address: noflshaker8@gmail.com

and

Ahmed Abdulhussein Jabbar,
Department of Mathematics,
Najaf Education Directorate, Ministry of Education,
Iraq.
E-mail address: ahmed.jabbar.pure246@student.uobabylon.edu.iq