A hybrid GA-MHW algorithm for solving nonlinear equations

Adithya P Devanandan[1], Sreedeep C D[2],

Abstract

This work introduces a hybrid method combining the search method Genetic Algorithm (GA) and the derivative-free iterative method MHW for solving nonlinear equations with high accuracy. The algorithm for the hybrid method to solve nonlinear equations with multiroots is provided. The algorithm is implemented using fitness and cost functions. Using eight test functions, hybrid Steffensen-genetic algorithm and hybrid GA-MHW (multiroot) are compared, and the results are analysed. The results shows that the proposed hybrid method yields the best results with high accuracy.

AMS Subject Classification: 65H04, 65H05
Keywords: Genetic Algorithm, MHW

1 Introduction

Numerical analysis plays a significant role in a wide range of disciplines, including engineering, physical sciences, medicine, biomedical research, and applied mathematics, where reliable computational techniques are essential for modelling and analysis. One of its fundamental components is root-finding, as many real-world problems can be formulated in the form

$$d(x) = 0,$$

where $d: D \subseteq \mathbb{R} \to \mathbb{R}$ is a continuous function defined on the domain D. The solutions to such problems are obtained by determining the roots of these functions. A variety of iterative methods have been developed to solve nonlinear equations [1, 2, 4, 6, 8, 11,13]. Some of these approaches require the computation of derivatives, which can be computationally expensive or difficult to evaluate for certain nonlinear models, especially those arising in medical and biological applications. To address this challenge, derivative-free iterative methods have been introduced as efficient alternatives, allowing nonlinear equations to be solved without the explicit use of derivatives while preserving accuracy and robustness.

One such approach is the MHW method proposed by Hassan in 2005 [5]. This method attains sixth-order convergence with an efficiency index of 1.565. Instead of relying on analytical derivatives, it employs divided differences as effective approximations of derivatives. The iterative scheme of the method is given by

$$w_k = x_k - \beta d(x_k),$$

[1]Department of Mathematics, Amrita School of Physical sciences, Amritapuri, India,
[2]Department of Mathematics, Amrita School of Physical sciences, Amritapuri, India,

$$b_k = x_k - \frac{d(x_k)}{d[w_k, x_k]},$$

$$v_k = b_k + \frac{d(b_k)}{d[b_k, x_k]},$$

$$x_{k+1} = v_k + \frac{d(v_k)}{d[b_k, v_k] + d[x_k, b_k] - d[x_k, v_k]},$$

where $\beta = 0.5$.

This study proposes a hybrid GA-MHW algorithm (Hybrid MGA) that integrates the global exploration ability of the Genetic Algorithm (GA) [10,16] with the high-order convergence of the sixth-order MHW method introduced by Hassan in 2005 [5]. The procedure begins with the initialisation of a population, after which genetic operators are applied to generate successive populations of candidate solutions. Based on a predefined fitness criterion, a selected subset of these candidates is chosen and employed as starting points for the MHW iteration. If the solution obtained through the MHW process satisfies the prescribed tolerance level, it is retained as a candidate root. A distinctness criterion is then applied to the refined solutions to cluster numerically close root approximations and filter out duplicates, with one representative solution from each cluster added to the set of final roots.

The subsequent section presents the detailed methodology along with the algorithm employed for solving nonlinear equations. This is followed by a section containing numerical experiments, where the performance of two different methods is evaluated and compared in terms of iteration count, accuracy, and computational efficiency, together with a discussion of the observed results.

2 Hybrid GA-MHW algorithm

In the proposed framework, the Genetic Algorithm is first employed to explore the search domain, and a selected set of solutions generated by GA is subsequently used as initial approximations for the MHW method. The GA process starts by creating a random initial population within the prescribed domain, with the boundary points explicitly included to enhance search coverage. A population consists of a fixed number of candidate solutions, which are randomly sampled from the domain to match the specified population size.

Each individual in the population is evaluated using the cost function $c = |d(x)|$, and its corresponding fitness value is computed as $fitness = \exp\left(\frac{1}{c+\epsilon}\right)$, where ϵ is a small positive constant introduced to prevent division by zero. At the initial stage, the solution with the minimum cost is identified as the best solution. During subsequent generations, this best solution and its cost are updated whenever a candidate with a lower cost is encountered.

To generate the next population, tournament selection is employed [6]. In this process, three individuals are randomly chosen from the current population, and the one with the highest fitness is selected for inclusion in the mating pool. This procedure is repeated until the mating pool reaches the desired population size. Pairs of individuals from the mating pool, denoted by p_1 and p_2, undergo linear crossover with a fixed probability cp, producing two offspring defined as $c_1 = \alpha p_1 + (1-\alpha)p_2$ and $c_2 = (1-\alpha)p_1 + \alpha p_2$, where $\alpha = 0.7$. If crossover does not occur, the parent solutions are directly passed to the next generation.

Following crossover, mutation is applied to each offspring with probability mp. The mutation operator perturbs the offspring by adding a random value selected from the interval $\left(-0.1(b_u - b_l), 0.1(b_u - b_l)\right)$, after which the resulting solution is clipped to ensure it remains within the predefined domain bounds.

After the offspring population is formed, a fixed number of the top-performing individuals (up to 95) are selected for refinement using the MHW method. Each selected individual is iteratively refined until either a specified tolerance is satisfied or a maximum number of iterations is reached. In cases where numerical instability arises during the MHW process, the iteration is halted and the current estimate is retained as the refined solution. These refined solutions then replace their corresponding individuals in the offspring population, and the global best solution and its cost are updated whenever an improvement is achieved.

The refined solutions obtained across all generations are accumulated throughout the GA execution. The algorithm then proceeds to subsequent generations, using the refined offspring population as the current population and repeating the steps of selection, crossover, mutation, refinement, and root collection. Upon completion of all generations, a clustering procedure is applied to the collected refined solutions. Only those solutions whose cost satisfies the prescribed tolerance are retained, while the remaining solutions are grouped using a distinctness threshold. Within each cluster, the solution with the smallest cost is selected as the representative root.

This hybrid strategy effectively merges the global search capability of the Genetic Algorithm with the high-order convergence of the MHW method, resulting in a robust and accurate approach for identifying multiple roots of nonlinear equations.

Algorithm 1 Hybrid GA-MHW Algorithm (multiroot)

Input: Function $d(x)$, domain bounds $[b_l, b_u]$, population size N, maximum generations G, crossover probability cp, mutation probability mp, tolerance tol, MHW parameters β and maximum iterations max_iter, distinct tolerance $distinct\_tol$.
Initialize: Generate a random population pop of size N within the domain and explicitly include the boundary points b_l and b_u.
Set $best\_solution = None$, $best\_cost = \infty$, $all\_roots = [\,]$.
for $gen = 1$ to G **do**
  Compute cost for all solutions: $costs = |d(pop)|$.
  Compute fitness: $fitness = exp\big(1/(costs + 10^{-12})\big)$.
  Update best solution:
  if $min(costs) < best\_cost$ **then**
    $best\_cost \leftarrow min(costs)$
    $best\_solution \leftarrow$ corresponding solution
  end if
  Select parents using tournament selection ($size = N$).
  Apply linear crossover with probability cp to produce offspring.
  Apply mutation with probability mp to each offspring, and clip to domain bounds.
  MHW refinement:
  Sort offspring by $|d(x)|(ascending)$ and select the top $min(95, N)$ solutions.
  for each selected solution **do**
    Apply MHW with fixed parameter β for up to max_iter iterations.
    if numerical instability is detected **then**
      Terminate MHW and accept the current iterate as the refined solution.
    end if
  end for
  Replace the selected offspring with their MHW-refined values.
  if $min(|d(refined)|) < best\_cost$ **then**
    $best\_cost \leftarrow min(|d(refined)|)$
    $best\_solution \leftarrow$ corresponding refined solution
  end if
  Update population: $pop \leftarrow$ offspring.
  Append all refined solutions to all_roots.
end for
Retain solutions in all_roots satisfying $|d(x)| < tol$.
Sort retained solutions.
Cluster solutions using $distinct\_tol$.
Accept the solution with the lowest cost value in each cluster.
Return: $best\_solution$, $best\_cost$, $final\_roots$.

3 Numerical Examples

In this section, eight test functions $[3, 5, 7, 9, 11, 12, 14, 15]$ are evaluated using the proposed algorithm to obtain their corresponding roots, followed by the comparison of the performance of hybrid Steffensen-genetic algorithm (SGA) and Hybrid MGA.

For MHW, we define the parameter $\beta = 0.5$, tolerance 10^{-8}, and maximum iterations 5. For GA, we use Tournament Selection, Linear crossover with probability $cp = 0.7$, Random mutation with probability $mp = 0.1$, problem-based domain, an initial population size of 90,

and a parents-to-offspring ratio of 1: 1. Fitness is calculated by; $fitness = e^{\frac{1}{|c|+10^{-12}}}$, where c is the cost $c = |d(x)|$.

3.1 Multiroot

The sets of roots of the eight test functions $[3, 5, 7, 9, 11, 12, 14, 15]$ obtained using the Hybrid MGA are presented in the table.

Function	No. of Iter/Cycles	Domain	Roots	Time
$x^2 + \sin\left(\frac{x}{5}\right) - \frac{1}{4}$	5	$[-1,1]$	-0.6096058960527, 0.4099920179891	19.5801877975463 ms
$\sin^2 x - x^2 + 1$	5	$[-5,3]$	-1.4044916482153, 1.4044916482153	26.1129617691040 ms
$\sin(2\cos x) - 1 - x^2 + e^{\sin(x^3)}$	5	$[-1,2]$	-0.7848959876612, 1.3061752018468	33.5825204849243 ms
$10xe^{-x^2} - 1$	5	$[0,2]$	0.1010258483156, 1.6796306104284	21.0745906829834 ms
$x^{10} - 2x^3 - x + 1$	5	$[0,2]$	0.5914480933407, 1.1103391853581	16.5972185134887 ms
$x^2e^x - \sin x + x$	5	$[-2,2]$	-1.4993930969014, $1.2441332976267 \times 10^{-06}$	38.3791875839233 ms
$x + 1 - e^{\sin x}$	5	$[-2,2]$	$5.9568639235638 \times 10^{-06}$, 1.6968123868097	40.8413076400756 ms
$(2x\lvert\cos x\rvert + x^2 - 1)^2 - 2$	5	$[-4,4]$	-2.7047848565256, -1.8211663327381, -1.5622967227141, -0.8409530969929, -0.2441531671132, 1.3507869552842	124.6291494369506 ms

3.2 Comparative Study

The comparison of the performance of SGA and Hybrid MGA is given below.

Example 1:

$$d(x) = x^2 + \sin\left(\frac{x}{5}\right) - \frac{1}{4}. \qquad x \in [-1,1]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	0.409992017	$1.613339315 \times 10^{-17}$	19.580187797 ms
Hybrid SGA (multiroot)	10	0.409992017	$6.217248937 \times 10^{-15}$	148.535728454 ms

Example 2:

$$d(x) = \sin^2 x - x^2 + 1. \qquad x \in [-5,3]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	−1.404491648	$2.750291543 \times 10^{-16}$	26.112961769 ms
Hybrid SGA (multiroot)	10	−1.404491648	$1.776356839 \times 10^{-15}$	188.740658760 ms

Example 3:

$$d(x) = \sin(2\cos x) - 1 - x^2 + e^{\sin(x^3)}. \qquad x \in [-1,2]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	−0.784895987	$1.250121952 \times 10^{-16}$	33.582520484 ms
Hybrid SGA (multiroot)	10	−0.784895987	$2.553512956 \times 10^{-15}$	466.346526145 ms

Example 4:

$$d(x) = 10xe^{-x^2} - 1. \qquad x \in [0,2]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	1.679630610	$7.017556902 \times 10^{-17}$	21.074590682 ms
Hybrid SGA (multiroot)	10	1.679630610	$2.220446049 \times 10^{-16}$	139.860453605 ms

Example 5:

$$d(x) = x^{10} - 2x^3 - x + 1. \qquad x \in [0,2]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	0.591448093	$5.005929481 \times 10^{-17}$	16.597218513 ms
Hybrid SGA (multiroot)	10	0.591448093	$4.440892098 \times 10^{-16}$	100.168600082 ms

Example 6:

$$d(x) = x^2e^x - \sin x + x. \qquad x \in [-2,2]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	-1.499393096	$8.823518455 \times 10^{-17}$	38.379187583 ms
Hybrid SGA (multiroot)	10	-1.499393096	$2.486899575 \times 10^{-14}$	147.093033790 ms

Example 7:

$$d(x) = x + 1 - e^{\sin x}. \qquad x \in [-2,2]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	1.696812386	$2.676717951 \times 10^{-16}$	40.841307640 ms
Hybrid SGA (multiroot)	10	1.696812386	$1.953992523 \times 10^{-14}$	115.104446411 ms

Example 8:

$$d(x) = (2x|\cos x| + x^2 - 1)^2 - 2. \qquad x \in [-4,4]$$

Methods	No. of Iter/Cycles	Root	Cost	Time
Hybrid MGA (multiroot)	5	-1.821166332	$4.553970946 \times 10^{-16}$	124.629149436 ms
Hybrid SGA (multiroot)	10	-1.821166332	$7.993605777 \times 10^{-15}$	240.592160224 ms

4 Conclusion

The comparative analysis demonstrates that the proposed Hybrid MGA outperforms the SGA in terms of both solution accuracy and computational efficiency when solving nonlinear equations. The results show that the Hybrid MGA approach consistently delivers higher precision while requiring reduced computational time. In addition to enhanced accuracy, the proposed method exhibits increased robustness with respect to the choice of initial guesses, making it reliable even in challenging problem settings. Furthermore, the algorithm is capable of successfully identifying multiple distinct roots of nonlinear functions. These characteristics make the proposed approach particularly suitable for complex real-world applications, including problems arising in engineering, medical and biological modelling, and other applied sciences, where accurate and efficient solution of nonlinear equations is essential.

References

[1] Abbasbandy, S. (2003). Improving Newton–Raphson method for nonlinear equations by modified Adomian decomposition method. *Applied Mathematics and Computation*, 145(2–3), 887–893.

[2] Argyros, I. K. (2008). *Convergence and applications of Newton-type iterations*. Springer Science & Business Media.

[3] Feng, X., & He, Y. (2007). High order iterative methods without derivatives for solving nonlinear equations. *Applied Mathematics and Computation*, 186(2), 1617–1623.

[4] Gouri M Krishna, & Sreedeep C D. (2026). An Improved Secant Type Method for Solving Nonlinear Optimization Problems.

[5] Hassan, M., Ali, M., & Waseem, M. (2025). Derivative Free Iterative Method for Solving Nonlinear Equations with Stability Analysis. *Southern Journal of Computer Science*, 1(02), 38–63.

[6] Haupt, R. L., & Werner, D. H. (2007). *Genetic Algorithms in Electromagnetics*. John Wiley & Sons.

[7] Hueso, J. L., Mart´ınez, E., & Teruel, C. (2015). Determination of multiple roots of nonlinear equations and applications. *Journal of Mathematical Chemistry*, 53(3), 880–892.

[8] Kumar, S., Kumar, D., Sharma, J. R., Cesarano, C., Agarwal, P., & Chu, Y. M. (2020). An optimal fourth order derivative-free numerical algorithm for multiple roots. *Symmetry*, 12(6), 1038.

[9] Parhi, S. K., & Gupta, D. K. (2008). A sixth order method for nonlinear equations. *Applied Mathematics and Computation*, 203(1), 50–55.

[10] Pavithra Krishna, & Sreedeep C D. (2026). An Enhanced Genetic Algo rithm for Root-Finding in Nonlinear Equation.

[11] Qiu, T., Hu, H., Chen, R., Zhou, Q., Guan, Q., & Li, X. (2021). A multiroot solver for discontinuous and non-differentiable equations by integrating genetic algorithm and derivative-free iterative methods. *Applied Soft Computing*, 109, 107493.

[12] Soleymani, F., Khattri, S. K., & Vanani, S. K. (2012). Two new classes of optimal Jarratt-type fourth-order methods. *Applied mathematics letters*, *25*(5), 847-853.

[13] Sreedeep, C. D., Argyros, I. K., Dinesh, K. G., Shrestha, N., & Argyros, M. (2025). Convergence analysis of a power series based iterative method having seventh order of convergence.*Matematychni Studii*, 64(2), 179-193.

[14] Wang, X., & Zhang, T. (2013). A family of Steffensen type methods with seventh-order convergence. *Numerical Algorithms*, 62(3), 429–444.

[15] Wu, X., & Fu, D. (2001). New high-order convergence iteration methods without employing derivatives for solving nonlinear equations. *Computers & Mathematics with Applications*, *41*(3-4), 489-495.

[16] Zafar, F., Cordero, A., Mujtaba, S., & Torregrosa, J. R. (2025). A Hybrid Steffensen–Genetic Algorithm for Finding Multi-Roots of Nonlinear Equations and Applications to Biomedical Engineering. *Algorithms*, 18(9), 582.