Optimization Techniques (McCormick Problem)

Mükrime Nur Gümüş
Dept. of Mathematical Engineering
Yıldız Technical University
İstanbul, Türkiye
mukrimenurgumus@gmail.com

Prof. Hale KÖÇKEN
Dept. of Mathematical Engineering
Yıldız Technical University
İstanbul, Türkiye
hgonce@yildiz.edu.tr

Assoc. Prof. Gökhan GÖKSU
Dept. of Mathematical Engineering
Yıldız Technical University
İstanbul, Türkiye
gokhan.goksu@yildiz.edu.tr

***Abstract*—This paper uses MATLAB codes and optimization techniques to solve the McCormick Problem [1].**

Keywords—optimization, McCormick Problem

I. Introduction

This paper focuses on the application of various optimization techniques. Following the guidelines outlined in the assignment paper, we will solve the McCormick problem with following methods;

- Newton-Raphson,
- Hestenes-Stiefel,
- Polak-Ribiere,
- Fletcher-Reeves,
- Dai-Yuan.

The Dai–Yuan algorithm is also incorporated alongside the other optimization methods mentioned in the assignment paper. Objective functions play a crucial role in evaluating and comparing the performance of optimization algorithms. Although numerous benchmark or test functions have been proposed in the literature, there is no universally accepted standard set of such functions. In this study, the selection of the benchmark function is determined based on the student identification number, where a = 8 and b = 3. Applying the formula ab (mod 50), we obtain the value 24. Accordingly, benchmark function number 24, McCormick Problem, as specified in the assignment document, will be used for analysis [1].

The McCormick optimization problem is a well-known benchmark problem used to evaluate the performance of nonlinear optimization algorithms. It is characterized by its non-convex nature and the presence of multiple local minima, making it a suitable test case for assessing the robustness and efficiency of various optimization methods. The problem involves minimizing a specific two-variable function that poses challenges due to its complex landscape [2]. In this study, we apply several gradient-based optimization techniques—including Newton-Raphson, Hestenes-Stiefel, Polak–Ribiere, Fletcher–Reeves, and Dai–Yuan algorithms—to solve the McCormick problem and compare their effectiveness in finding the global minimum.

II. Numeric Solve

A. First-Order Necessary Conditions

The objective function is defined as:

$$f(x_1, x_2) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1 \quad (1)$$

subject to the constraints:

$$-1.5 \le x_1 \le 4, -3 \le x_2 \le 3 \quad (2)$$

To obtain the numerical solution of the optimization problem, we first examine the first-order necessary conditions. This involves setting the gradient of the objective function equal to zero and solving for the variables x_1 and x_2.

The gradient of the function is defined as follows:

$$\nabla_f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (3)$$

To identify the critical points, the gradient of the function must be computed and set equal to zero. Using the third equation:

$$\begin{bmatrix} \cos(x_1 + x_2) + 2(x_1 - x_2) - 1.5 \\ \cos(x_1 + x_2) - 2(x_1 - x_2) + 2.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (4)$$

From fourth equation we find the equations:

$$x_1 = 1 + x_2 \quad (5)$$

$$\cos(x_1 + x_2) = -0.5 \quad (6)$$

The cosine values equal to -0.5 correspond to angles $\frac{2\pi}{3}$ and $\frac{4\pi}{3}$. Therefore, two distinct cases arise from this condition:

- **Case 1**: $x_1 + x_2 = \frac{2\pi}{3} + 2k\pi$
- **Case 2**: $x_1 + x_2 = \frac{4\pi}{3} + 2k\pi$

B. Finding Candidate Points for Optimization

a. Case 1

$$x_1 = {}^{\pi}/_{3} + 0.5 + k\pi \quad (7)$$

$$x_2 = {}^{\pi}/_{3} - 0.5 + k\pi \quad (8)$$

For $k = 0$, we obtain our first candidate point:

$$A(1.5471, 0.5471)$$

b. Case 2

$$x_1 = {}^{2\pi}/_{3} + 0.5 + k\pi \quad (9)$$

$$x_2 = {}^{2\pi}/_{3} - 0.5 + k\pi \quad (10)$$

For $k = 0$, we obtain second candidate point:

$$B(2.5943, 1.5943)$$

For $k = -1$, we obtain third candidate point:

$$C(-0.5471, -1.5471)$$

All candidate points within $-1.5 \le x_1 \le 4, -3 \le x_2 \le 3$ have been identified.

C. Second-Order Necessary Conditions (Hessian Matrix)

The Hessian matrix $H(x_1, x_2)$ is:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial f}{\partial x_2^2} \end{bmatrix} \quad (11)$$

Using the eleventh equation we obtain:

$$H(x_1, x_2) = \begin{bmatrix} -\sin(x, +x_2) + 2 & -\sin(x, +x_2) - 2 \\ -\sin(x_1 + x_2) - 2 & -\sin(x_1 + x_2) + 2 \end{bmatrix} \quad (12)$$

a. Point A

If we put the point A to the Hessian matrix:

$$H(1.5471,0.5471) = \begin{bmatrix} 1.1339 & -2.886 \\ -2.886 & 1.1339 \end{bmatrix}$$

And the $\det(H) \approx -6.9282 < 0$ therefore point A is a saddle point.

b. Point B

If we put the point A to the Hessian matrix:

$$H(2.5943,1.5943) = \begin{bmatrix} 2.2660 & -1.1339 \\ -1.1339 & 2.2660 \end{bmatrix}$$

And the $\det(H) \approx 6.92820 > 0$ and $H_{11} = 2.8660 > 0$ therefore point B is a local minimum.

c. Point C

$$H(2.5943,1.5943) = \begin{bmatrix} 2.2660 & -1.1339 \\ -1.1339 & 2.2660 \end{bmatrix}$$

And the $\det(H) \approx 6.92820 > 0$ and $H_{11} = 2.8660 > 0$ therefore point C is a also local minimum.

III. Implementation of Iterative Optimization Techniques

In accordance with the assignment specifications, initial guesses for the optimization algorithms were generated using a uniform distribution over the defined variable bounds. Specifically, for each problem, the initial point x_0 was sampled from the uniform distribution:

$$x_0 \sim U(x_{0,\min}, x_{0,max})$$

where $x_{0,\min}$ and $x_{0,\max}$ denote the lower and upper bounds for each decision variable, respectively, as specified for the function in the benchmark set. In this case, since the domain for each variable is $-1.5 \leq x_1 \leq 4$ and $-3 \leq x_2 \leq 3$, the initial guess vectors were generated using the rand function in MATLAB.

Using this method, three random initial guess points were obtained:

- **Initial Guess 1**: $[2.9810,\ 2.4348]$
- **Initial Guess 2**: $[-0.8016,\ 2.4803]$
- **Initial Guess 3**: $[1.9780,\ -2.4148]$

These initial points serve as the starting values for all optimization algorithms applied in this study. Additionally, the absolute error bound for convergence was taken as $\varepsilon = 10^{-4}$ for each algorithm, as instructed in the assignment guidelines.

A. Initial Guess 1

For the first initial guess, all optimization methods converged to the local minimum denoted as point B, which had previously been identified through analytical methods. While each algorithm reached the same solution, they differed in terms of the number of iterations and computation time required.

a. Newton-Raphson Method

Converged after 4 iterations. The computation time was approximately 0.299 seconds, and the algorithm converged to the point $x^* = [2.5944,\ 1.5944]$.

TABLE I. Newton-Raphson convergence for initial point 1

Iteration	Newton-Raphson	
	f(x)	*|| grad(f(x)) ||*
0	2.1510	2.0681
1	1.2595	0.3076
2	1.2289	0.0441
3	1.2283	0.0004
4	1.2283	0

b. Hestenes-Stiefel Method

Converged after 47 iterations. The total computation time was approximately 0.0260 seconds, and the algorithm reached the point $x^* = [2.5944,\ 1.5944]$.

TABLE II. Hestenes-Stiefel convergence for initial point 1

Iteration	Hestenes-Stiefel	
	f(x)	*|| grad(f(x)) ||*
0	2.1510	2.0681
1	1.4371	1.2873
2	1.3701	0.5892
3	1.3084	0.7954
4	1.2642	0.4606
10	1.2305	0.0997
20	1.2283	0.0059
30	1.2283	0.0012
40	1.2283	0.0002
47	1.2283	0

c. Polak-Ribiere Method

For the first initial guess, the Hestenes–Stiefel method converged after 15 iterations. The total computation time was approximately 0.0113 seconds, and the algorithm reached the point $x^* = [2.5944,\ 1.5944]$.

TABLE III. POLAK-RIBIERE CONVERGENCE FOR INITIAL POINT 1

Iteration	Polak-Ribiere Method	
	f(x)	*|| grad(f(x)) ||*
0	2.1510	2.0681
1	1.4371	1.2873
2	1.3956	1.0243
3	1.3431	0.9492
4	1.3395	0.9428
10	1.2283	0.0004
15	1.2283	0

d. Fletcher-Reeves Method

Converged after 24 iterations. The total computation time was approximately 0.0095 seconds, and the algorithm reached the point point $x^* = [2.5944,\ 1.5944]$.

TABLE IV. FLETCHER-REEVES CONVERGENCE FOR INITIAL POINT 1

Iteration	Fletcher-Reeves Method	
	f(x)	*|| grad(f(x)) ||*
0	2.1510	2.0681
1	1.4371	1.2873
2	1.3126	0.5350
3	1.2416	0.3247
4	1.2372	0.2375
10	1.2285	0.0365
20	1.2283	0.0005
24	1.2283	0

e.Dai-Yuan Method

Converged after 12 iterations. The total computation time was approximately 0.0087 seconds, and the algorithm reached the point point $x^* = [2.5944,\ 1.5944]$.

TABLE V. DAI-YUAN CONVERGENCE FOR INITIAL POINT 1

Iteration	Dai-Yuan Method		
	f(x)	*|| grad(f(x)) ||*	*β*
0	2.1510	2.0681	-
1	1.4371	1.2873	0.288
2	1.3069	0.6477	0.218
3	1.2418	0.2375	0.204
4	1.2283	0.0076	0.0010
10	1.2283	0.0003	0.244
12	1.2283	0	-

B. Initial Guess 2

For the second initial guess, not all optimization methods converged to the local minimum denoted as point B or C.

a. Newton-Raphson Method

Converged after 3 iterations. The computation time was approximately 0.0054 seconds, and the algorithm converged to the point $x^* = [1.5472, 0.5472]$ which is a saddle point. This result demonstrates a known characteristic of the pure Newton-Raphson method: its ability to converge to any type of stationary point (minima, maxima, or saddle points) where the gradient is zero, due to its reliance on finding the stationary point of its local quadratic model.

TABLE VI. NEWTON-RAPHSON CONVERGENCE FOR INITIAL POINT 2

Iteration	Newton-Raphson	
	f(x)	*|| grad(f(x)) ||*
0	2.0167	1.2123
1	1.9130	0.02599
2	1.9132	0.0001
3	1.9132	0

.

b. Hestenes-Stiefel Method

Converged after 37 iterations. The total computation time was approximately 0.0029 seconds, and the algorithm reached the point C, $x^* = [-0.5472, -1.5472]$ which is global minimum

TABLE VII. HESTENES-STIEFEL CONVERGENCE FOR INITIAL POINT 2

Iteration	Hestenes-Stiefel	
	f(x)	*|| grad(f(x)) ||*
0	20.1675	12.1235
1	19.9370	12.1610
2	17.3885	12.3406
3	1.5403	1.2256
4	-0.5782	0.1991846
10	-1.90341	0.0020
20	-1.9132	0.0002
30	-1.9132	0.0002
37	-1.9132	0

c. Polak-Ribiere Method

Converged after 26 iterations. The total computation time was approximately 0.0029 seconds, and the algorithm reached the point C, $x^* = [-0.5472, -1.5472]$ which is global minimum.

TABLE VIII. POLAK-RIBIERE CONVERGENCE FOR INITIAL POINT 2

Iteration	Polak-Ribiere Method	
	f(x)	*|| grad(f(x)) ||*
0	20.1675	12.1235
1	19.9370	12.1610
2	19.0713	12.2659
3	17.1357	12.2187
4	16.4236	12.1112
10	-1.9131	0.0116
20	-1.9132	0.0005
26	-1.9132	0

d. Fletcher-Reeves Method

Converged after 26 iterations. The total computation time was approximately 0.0025 seconds, and the algorithm reached the point $x^* = [-6.8304, -7.8304]$. At this point, the gradient norm and change in function value met the specified stopping criteria. This result highlights that for bounded problems, unconstrained algorithms might converge to optima or stationary points outside the desired domain. For practical application to such problems, methods incorporating constraint handling (e.g., penalty methods, barrier methods, active set methods, or projection onto the feasible set) would be necessary.

TABLE IX. FLETCHER-REEVES CONVERGENCE FOR INITIAL POINT 2

Iteration	Fletcher-Reeves Method	
	f(x)	*|| grad(f(x)) ||*
0	20.1675	12.1235
1	19.9370	12.1610
2	17.8476	12.4890
3	17.2686	12.2722
4	11.1537	11.0633
10	-8.1963	0.0185
20	-8.19640	0.0007
26	-8.19640	0

e. Dai-Yuan Method

Converged after 15 iterations. The total computation time was approximately 0.0026 seconds, and the algorithm reached the point point $x^* = [-9.9720, -10.9720]$. This result, along with the Fletcher-Reeves result, strongly reinforces the point that unconstrained algorithms, when applied to problems with implicit or explicit bounds they don't handle, can and will find stationary points outside those bounds if the function's landscape leads them there.

TABLE X. DAI-YUAN CONVERGENCE FOR INITIAL POINT 2

Iteration	Dai-Yuan Method		
	f(x)	*|| grad(f(x)) ||*	*β*
0	20.1675	12.1235	-
1	19.9370	12.1610	0.505
2	16.6971	12.0219	1.01
3	14.4951	11.4782	0.963
4	12.3774	11.8583	1.05
10	-11.3292	0.1815	0.0481
12	-11.3380	4.8659	-

C. Initial Guess 3

For the third initial guess, not all optimization methods converged to the local minimum denoted as point B or C.

a. Newton-Raphson Method

Converged after 4 iterations. The computation time was approximately 0.0008 seconds, and the algorithm converged to the point C, $x^* = [-1.5944, -2.5944]$.

TABLE XI. NEWTON-RAPHSON CONVERGENCE FOR INITIAL POINT 3

Iteration	Newton-Raphson	
	f(x)	*|| grad(f(x)) ||*
0	10.8692	9.7999
1	-1.3000	0.4445
2	-1.2340	0.1434
3	-1.2283	0.0037
4	-1.2283	0

b. Hestenes-Stiefel Method

Converged after 75 iterations. The total computation time was approximately 0.0020 seconds, and the algorithm reached the point $x^* = [-3.6888, -4.6888]$.

TABLE XII. HESTENES-STIEFEL CONVERGENCE FOR INITIAL POINT 3

Iteration	Hestenes-Stiefel	
	f(x)	*|| grad(f(x)) ||*
0	10.8692	9.7999
1	9.62598	9.6016
2	8.10868	8.8882
3	-4.9716	0.7301
4	-4.9766	0.6171
10	-5.0514	0.1355
20	-5.0547	0.0093
30	-5.0548	0.0093
40	-5.0548	0.0001
50	-5.0548	0.0001
60	-5.0548	0.0001
70	-5.0548	0.0001

Iteration	Hestenes-Stiefel	
	f(x)	*|| grad(f(x)) ||*
75	-5.0548	0

c. Polak-Ribiere Method

Converged after 19 iterations. The total computation time was approximately 0.0006 seconds, and the algorithm reached the point C, $x^* = [-0.5472, -1.5472]$.

TABLE XIII. POLAK-RIBIERE CONVERGENCE FOR INITIAL POINT 3

Iteration	Polak-Ribiere Method	
	f(x)	*|| grad(f(x)) ||*
0	10.8692	9.7999
1	9.6259	9.6016
2	9.5975	9.5961
3	-1.9131	0.01404
4	-1.9131	0.01030
10	-1.9132	0.00158
19	-1.9132	0

d. Fletcher-Reeves Method

Converged after 28 iterations. The total computation time was approximately 0.0007 seconds, and the algorithm reached the point $x^* = [-3.6888, \ -4.6888]$.

TABLE XIV. FLETCHER-REEVES CONVERGENCE FOR INITIAL POINT 3

Iteration	Fletcher-Reeves Method	
	f(x)	*|| grad(f(x)) ||*
0	10.8692	9.7999
1	9.6259	9.6016
2	6.5722	8.1351
3	5.2017	8.8041
4	4.2224	8.6124
10	-5.0497	0.1328
20	-5.0548	0.0072
28	-5.0548	0

e. Dai-Yuan Method

For the first initial guess, the Fletcher-Reeves method converged after 21 iterations. The total computation time was approximately 0.0026 seconds, and the algorithm reached the point point $x^* = [-0.5472, -1.5472]$.

TABLE XV. DAI-YUAN CONVERGENCE FOR INITIAL POINT 3

Iteration	Dai-Yuan Method		
	f(x)	*|| grad(f(x)) ||*	*β*
0	10.8692	9.7999	-
1	9.6259	9.6016	0.492
2	-1.7468	0.6285	0.00810
3	-1.8871	0.3871	0.271
4	-1.8976	0.3020	0.552
10	-1.9131	0.0130	0.193
20	-1.9132	0.0002	0.805
21	-1.9132	0	-

IV. CONCLUSION

Benchmark table is given below:

TABLE XVI. BENCHMARK TABLE

Initial Guess	Algorithm	Iterations	Time (s)	x^*
1	Newton-Raphson	4	0.0299	$[2.5944, 1.5944]$
1	Hestenes-Stiefel	47	0.0260	$[2.5944, 1.5944]$
1	Polak-Ribiere	15	0.0113	$[2.5944, 1.5944]$
1	Fletcher-Reeves	24	0.0095	$[2.5944, 1.5944]$
1	Dai-Yuan	12	0.0087	$[2.5944, 1.5944]$
2	Newton-Raphson	3	0.0054	$[1.5472, 0.5472]$
2	Hestenes-Stiefel	37	0.0029	$[-0.5472, -1.5472]$
2	Polak-Ribiere	26	0.0029	$[-0.5472, -1.5472]$
2	Fletcher-Reeves	26	0.0025	$[-6.8304, -7.8304]$
2	Dai-Yuan	15	0.0026	$[-9.9720, -10.9720]$
3	Newton-Raphson	4	0.0008	$[-1.5944, -2.5944]$
3	Hestenes-Stiefel	75	0.0020	$[-3.6888, -4.6888]$
3	Polak-Ribiere	19	0.0006	$[-0.5472, -1.5472]$
3	Fletcher-Reeves	28	0.0007	$[-3.6888, -4.6888]$
3	Dai-Yuan	21	0.0006	$[-0.5472, -1.5472]$

Figures below show the contour plot of the McCormick function $f\_24(x_1, x_2)$ along with the iteration paths taken by different algorithms from three different initial starting points.

The starting points are indicated by larger markers (blue, orange, yellow circles). The subsequent iteration points are color-coded by iteration number, where points corresponding to the same iteration number across different paths share the same color/marker style (as per the legend 'dataX'). Lines connect successive iterations for each path, colored by the initial starting point's path color.

The background contours represent lines of constant function value, with warmer colors (e.g., yellow on the color bar) indicating higher function values and cooler colors (e.g., dark blue) indicating lower function values.

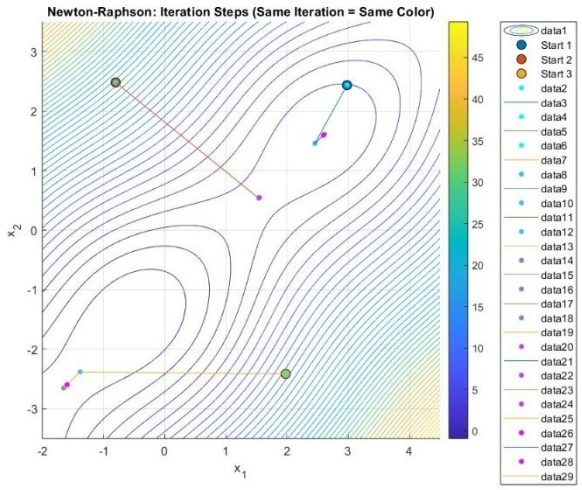

Fig. 2. Newton-Raphson Method

The most striking observation is the strong dependence of the Newton-Raphson method's convergence outcome on the initial starting point. Each of the three paths converges to a different stationary point of the function

Because of the nature of Newton-Raphson method, steps can be quite large, , especially in the initial iterations when far from a solution (e.g., the first step from Start 3). These steps directly target the minimum of the local quadratic approximation of the function.

When converging to a local minimum (like paths from Start 1 and Start 2), the method demonstrates its power in finding solutions with high precision in a very small number of iterations, provided the starting point is within the basin of attraction of that minimum and the Hessian is well-behaved in that region.

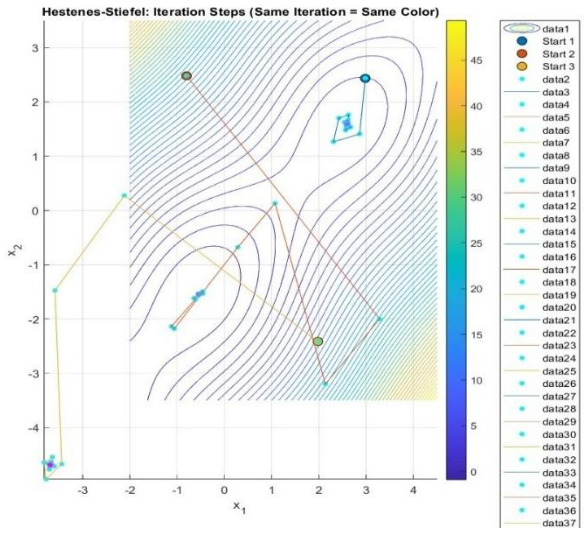

Fig. 3. Hestenes-Stiefel Method

Similar to the Newton-Raphson method, the Hestenes-Stiefel method demonstrates sensitivity to the initial starting conditions, with different starting points leading to different basins of attraction and potentially different outcomes.

This visualization for the Hestenes-Stiefel method (figure 3) illustrates its typical conjugate gradient behavior: paths that are not strictly steepest descent, reliance on line searches, greater number of iterations and much more time compared to second-order methods. It also highlights its sensitivity to initial conditions and its potential to converge to saddle points or, if unconstrained, to stationary points outside the desired problem domain.

The performance from Start 3, in particular, underscores the importance of considering constraint-handling mechanisms when applying such algorithms to bounded problems.

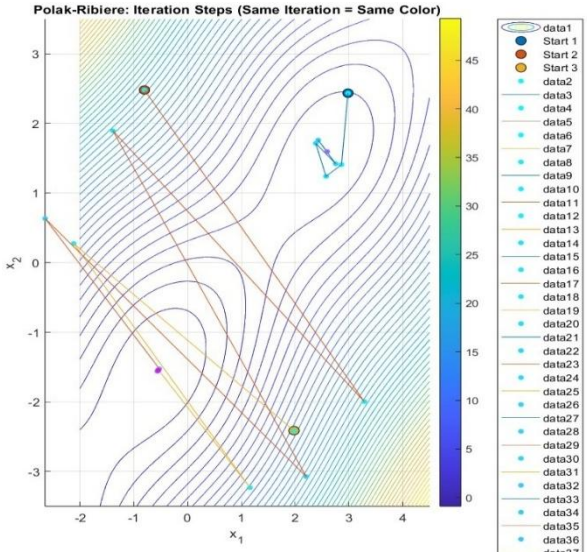

Fig. 4. Polak-Ribiere Method

The Polak-Ribiere method, as visualized, demonstrates its characteristic behavior on the McCormick function. It shows a strong sensitivity to initial conditions, with the potential for large, exploratory initial steps. Its ability to converge to different types of stationary points (local minimum B, saddle point A, and global minimum C from these starts) is evident.

The performance, particularly the aggressive initial steps from some starting points, suggests that the formulation for β can lead to a more dynamic search compared to some other variants, which can be beneficial in some cases but might also lead to overshooting if the line search is not carefully tuned or if the function landscape is particularly challenging.

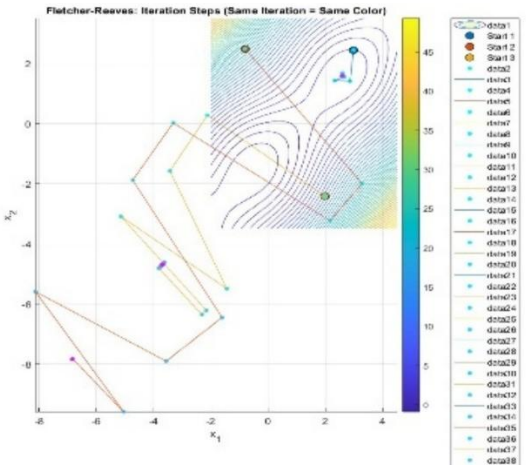

Fig. 5. Fletcher-Rieves Method

The Fletcher-Reeves method's performance on the McCormick function, as shown in Figure 5, highlights its sensitivity to initial conditions. While one starting point led to a known local minimum within the typical domain, two other starting points resulted in convergence to stationary points located far outside this region.

This behavior, particularly the tendency to explore and settle in distant, potentially flatter regions of the unconstrained function, might be attributed to the specific formulation of β in and its known susceptibility to less robust performance on certain non-quadratic problems compared to other variants.

The results strongly suggest that for problems with defined operational bounds, using the Fletcher-Reeves method (and other unconstrained algorithms) without appropriate constraint-handling mechanisms can lead to solutions that are mathematically valid for the unconstrained function but practically undesirable for the bounded problem.

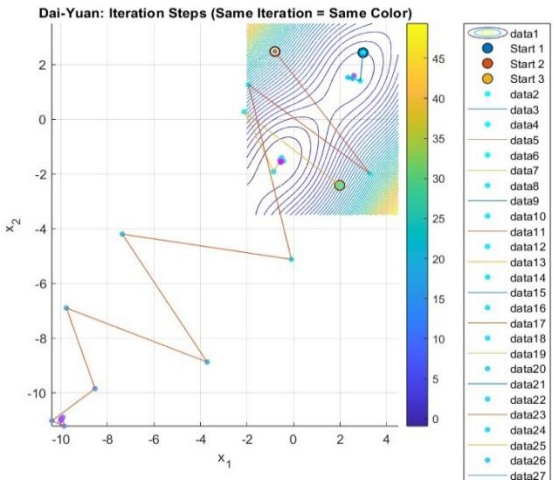

Fig. 6. Dai-Yuan Method

The Dai-Yuan method's performance on the McCormick function, illustrated in Figure 6, shows varied behavior depending on the starting location. While some initial points lead to convergence to a local minimum or a saddle point within the primary region of interest, others can trigger an extensive exploration of the unconstrained function space, leading to convergence at stationary points far outside the problem's defined bounds.

The trajectory of Path 2 is particularly illustrative of how the specific β formulation of the Dai-Yuan method can interact with the function landscape to produce wide-ranging search paths. While this could be advantageous for some global exploration tasks, it highlights the method's unconstrained nature.

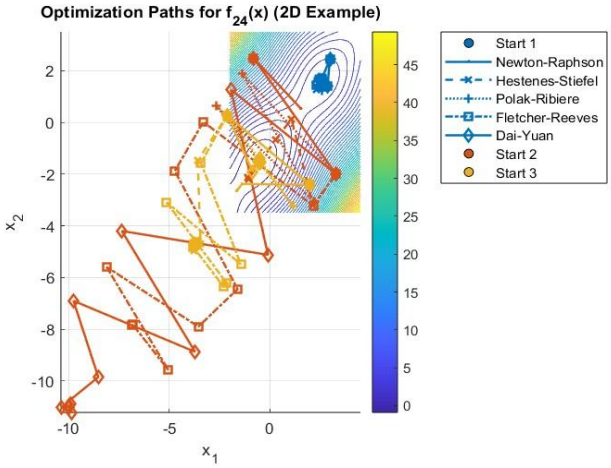

Fig. 7. Optimization Paths of different algorithms

Figure 7 provides a comparative visualization of the optimization paths taken by all implemented algorithms (Newton-Raphson, Hestenes-Stiefel, Polak-Ribière, Fletcher-Reeves, and Dai-Yuan) on the McCormick function. Paths originate from three distinct starting points.

Each algorithm is represented by a unique line style and marker (as detailed in the legend). The paths are colored according to their originating start point.

A consistent observation across all algorithms is their sensitivity to the initial starting point. For this multi-modal McCormick function, different starting locations lead the algorithms to converge to different stationary points, including the global minimum, a local minimum, a saddle point, or even stationary points far outside the intended feasible region.

The Newton-Raphson paths (solid lines) are generally characterized by very few, often large, direct steps. This reflects its quadratic convergence and reliance on second-order (Hessian) information. Its paths are visually distinct from the more iterative, often zig-zagging paths of the first-order Conjugate Gradient methods.

The Conjugate Gradient methods generally take more iterations and more time than Newton-Raphson. Their paths often involve more intermediate steps and changes in direction as they iteratively build up information about the curvature of the function.

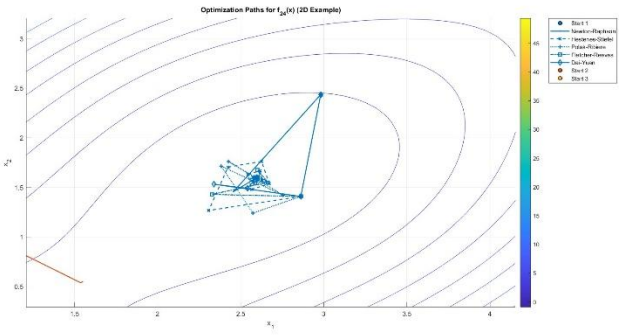

Fig. 8. Zoomed in graph for Start 1.

From Start 1 in Figure 8, all algorithms, including the Conjugate Gradient methods, eventually converge to the local minimum B. However, their paths to get there differ. For instance, Newton is very direct.

From Start 3 in Figure 7, Newton converges to the saddle point A. The CG methods (as seen in the inset for Start 3, though some paths quickly leave this zoomed region) also seem to initially head towards or explore points A or C before their paths fully develop.

These paths highlight the unconstrained nature of these algorithms. They are exploring and converging to stationary points of the unconstrained McCormick function that lie outside the implicitly defined feasible region of the benchmark problem.

Polak-Ribière (dotted-+) and Hestenes-Stiefel (dashed-x) from Start 2 also show significant exploration, though perhaps not as extremely as Fletcher-Rieves and Dai-Yuan.

This comparative plot underscores that while Newton-Raphson is very fast when it converges to a desirable point, its outcome is highly start-dependent and it can find saddle points.

The Conjugate Gradient methods, while generally requiring more iterations, also show strong start-point dependency. Their unconstrained nature can be a significant drawback for bounded problems, as demonstrated by several paths converging far outside the feasible region.

This strongly suggests that for reliable optimization of the McCormick function within its specified bounds, either very careful selection of initial guesses (if prior knowledge exists) or, more practically, the use of algorithms designed for constrained optimization would be necessary.

REFERENCES

[1] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," Journal of Global Optimization, vol. 31, no. 4, p. 635, 2005.

[2] McCormick, S.F. An iterative procedure for the solution of constrained nonlinear equations with application to optimization problems. *Numer. Math.* **23**, 371–385 (1975). https://doi.org/10.1007/BF01437037K. Elissa, "Title of paper if known," unpublished.