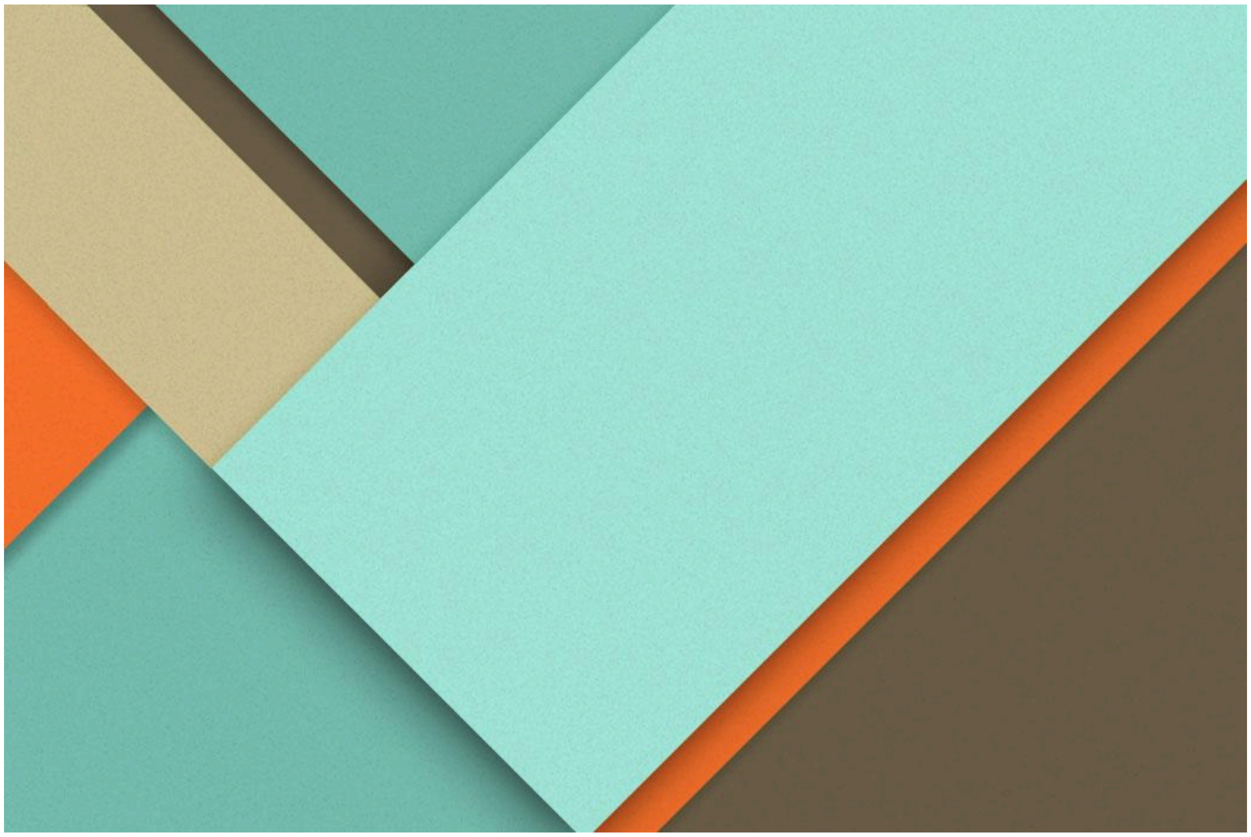
Report 3



# Comparative Study of Classical Optimization Algorithms in 1D and Multidimensional Spaces

Team :

| | |
|---|---|
| Moahmed Ehab Yousri | 202201236 |
| Hamza Abdelmoreed | 202201508 |
| Amin Gamal | 202202219 |

## Overview

The report examines the algorithms developed for unconstrained nonlinear optimization and applies them in problems.

The most important thing to mention is how the efficiency of these optimization methods was assessed on benchmark problems in terms of, for instance, the amount of iterations, rate of convergence, and time complexity.

---

## Section 1:

### **Algorithms** :

### Fibonacci Method:

The Fibonacci method is a recursive optimization algorithm that shrinks the search interval through Fibonacci numbers. It guarantees an effective downsizing of the interval size for a set tolerance.

Steps:

Generate the Fibonacci sequence until it satisfies the condition

Calculate two intermediary points c and d in the interval.

Assess the function at c and d to find the new interval.

Continue until interval width is below the accuracy tolerance.

## Advantages:

Efficient for bracketed minimization problems.

Guarantees convergence in the interval by gradually shrinking the search space.

No derivative information is required.

## Disadvantages:

Suffers from the precomputing need of the (Fibonacci) sequence, which can be computationally intensive with regard to small tolerances.

Slower than some methods includes derivatives , especially for smooth functions.

## Conclusion:

Most suitable for cases in which it is not possible or economical to compute the derivatives. Nevertheless, it can be less efficient than methods such as Newton’s or Golden Section for continuous functions.

---

## Golden Section Method:

The Golden Section Method applies the golden ratio to divide the search interval and progressively narrow the interval size. It is efficient and does not require derivative information.

### Steps :

Define the golden ratio constant.

Compute intermediate points c and d within the interval.

Assess the function at c and d and shrink the interval thereof.

Keep on doing this until the interval width is less than or equal to the tolerance value.

Advantages:

Simpler, since it employs a constant golden ratio rather than Fibonacci numbers.

Efficient and does not require derivatives.

Works well for a wide range of unimodal functions.

Disadvantages:

Slightly less efficient than Fibonacci for large tolerances.

It may be slower than methods based on derivatives when a gradient is available.

Conclusion:

A strong and clean approach suited when the function is unimodal and derivative information is not available.

---

## Newton's Method:

Newton's Method:

Newton's method, i.e., exploiting derivative and second derivative information, is used to solve the minimum.

Steps:

Compute the gradient and Hessian

Update x using the Newton-Raphson formula.

Repeat until convergence or until the maximum number of iterations is reached.

.Advantages:

Very fast convergence for smooth and reasonably behaved functions (quadratic convergence).

Very efficient when derivatives and second derivatives are easily accessible.

Disadvantages:

Uses both the first and second order derivatives of the function, which may be costly or difficult to compute.

It may fail if the second derivative is zero or if an initial guess is very far away from the solution.

Conclusion:

Best for smooth functions with easily computable derivatives. Not suitable for non-smooth or noisy functions.

---

## Quasi-Newton Method:

The Quasi-Newton method is an approach where in the approximation of the second derivative (Hessian) gradient differences are used and the position is iteratively updated. It is also effective for problems for which the Hessian is not easy to compute.

Steps:

Initialize the approximation H=1.0.

Adopt a line search to obtain the step size optimal.

Update x and the Hessian approximation.

Repeat until convergence.

### Advantages:

But it does not require the exact second derivative (Hessian), and thus is more general than Newton's method.

Converges more rapidly than for techniques that do not make use of gradient information (e.g., Fibonacci or Golden Section).

### Disadvantages:

Slower than Newton’s method due to approximate Hessian updates.

Needs computation of gradients, which can be expensive for certain functions.

### Conclusion:

Strikes a balance between efficiency and computational complexity. Suitable for problems where second derivatives are unavailable or expensive.

---

## Secant Method:

The Secant method is used to estimate the derivative through a finite difference formula. It is employed to determine the roots of the derivative f′(x)=0 for the minimization of f(x).

### Advantages:

Does not require explicit derivative computation, unlike Newton's method.

Faster than gradient-free methods like Fibonacci and Golden Section.

### Disadvantages:

Baser than Newton’s method, because it provides an approximation of derivatives based on finite differences.

Can diverge if initial guesses are poorly chosen.

---

## Section 2:

### Fletcher-Reeves Conjugate Gradient Method :

Frechert and Reeves suggested the Conjugate gradient method which is widely known as CG. It is used for the optimization of unconstrained nonlinear problems. Of more importance is the fact that CG does not require the evaluation of the cost function unless an iteration is being performed. Also this method employs both the steepest descent and a conjugate direction in order to reduce the time taken for convergence.

Fletcher-Reeves CG method estimates are obtained in a repetitive process through a sequence of smooth and differentiable objective functions, hence it almost entirely satisfies the quadratic convergence of quadratic functions.

**Pseudocode** :

Input: objective function and its gradient, initial point, tolerance and maximum iterations respectively.

```
1. Initialize: x ← x0, grad ← grad_f(x), d ← -grad
```

```
2. For k = 1 to max_iter: apply these steps :

    - Check to see if convergence is achieved : If ||grad|| < tol, return x.

    -Determine the line search by solving for α: α ← argmin f(x + α * d).

    - Update position: x_new ← x + α * d.

    - Calculate the updated gradient: grad_new ← grad_f(x_new).

    - Compute β: β ← ||grad_new||^2 / ||grad||^2.

    - Change the vectors: d ← -grad_new + β * d.

    - Alter the parameters: x ← x_new, grad ← grad_new.

3.  Then Return x.
```

Output: Optimal solution x*.

**Advantages:**

-Modular and suitable for large-scale applications.

-Good for a diverse range of scale operations.

-Only requires gradients to be evaluated or Hessian matrices to be calculated.

## Limitations:

Vulnerable to the accuracy of the line search.

Might be ineffective on non-convex or badly scaled problems.

Necessitates f(x) to be twice continuously differentiable.

## Results:

Number of Iterations: Tolerance and step size method-dependent.

Convergence Behavior: The gradient norms decrease rapidly for a few iterations. For quadratic functions, convergence is quadratic.

Optimal Solution x* : (1.0 , 1.0)

---

## Marquardt Method:

The Marquardt method is a very powerful optimization algorithm that minimizes nonlinear functions. It combines the advantages of the steepest descent method, which works well when the optimum is far away, and Newton's method, which converges fast near the optimum. This method modifies the Hessian matrix by adding a regularization parameter to it. This makes the method work in cases where the Hessian could be singular or ill-conditioned.

Pseudocode:

Input: objective function and its gradient,Hessian, initial regularization,initial point, tolerance and maximum iterations respectively..

```
1. Initialize: x ← x0, λ ← lambda_init.
2. For k = 1 to max_iter:  apply these steps :
    - Compute gradient: grad ← grad_f(x).
    -Check to see if convergence is achieved: If ||grad|| < tol, return x.
    - Compute Hessian: H ← hessian_f(x).
    - Regularize Hessian: H_reg ← H + λ * I.
    - Solve for step direction: δ ← -H_reg$^{-1}$ * grad.
    - Update x: x_new ← x + δ.
    - If f(x_new) < f(x):
        x ← x_new.
        λ ← λ / 10.
      Else:
        λ ← λ * 10.
    - Check step size: If ||δ|| < tol, return x.
3. Return x.
```

Output: Optimal solution x*

**Advantages:**

-Combines the advantages of steepest descent and Newton's method.

-Robust for ill-conditioned Hessians by regularization.

-Dynamically adjusts step size for efficiency in convergence.

**Limitations:**

-Computationally expensive to solve if the dimensionality of the Hessian is high.

-Sensitive to the choice of the initial regularization parameter λ\\lambdaλ.

**Results:**

Optimal solution: [0.99999998 0.99999996]

Function value at optimum: 4.866187419097994e-16

Number of iterations: 41

---

## Quasi-Newton Method:

Quasi-Newton methods represent the most popular iterative optimization schemes, approximating the Hessian matrix of the objective to achieve superlinear convergence. Of all the Quasi-Newton methods, the Broyden-Fletcher-Goldfarb-Shanno update rule is widely used because it's robust and efficient. The BFGS update rule does not explicitly calculate the Hessian matrix but instead uses successive gradient updates to iteratively refine an approximation of the inverse Hessian.

**Pseudocode**:

Input: objective function and its gradient, initial point, tolerance and maximum iterations respectively.

```
1. Initialize: x ← x0, H ← I, grad ← grad_f(x).
2. For k = 1 to max_iter:
    a Check to see if convergence is achieved : If ||grad|| < tol, return x.
    b.Change the vectors: d ← -H * grad.
    c. Line search: Find α that minimizes f(x + α * d).
    d. Update x: x_new ← x + α * d.
    e. Calculate gradients: grad_new ← grad_f(x_new).
    f. Calculate differences: s ← x_new - x, y ← grad_new - grad.
    g. Update H using BFGS formula.
    h. Update x and the grad: x ← x_new, grad ← grad_new.
3. Return x.
```

Output: Optimal solution x*.

**Advantages**

-Yields superlinear convergence near the optimum.

-Does not require explicit computation of the Hessian matrix.

-Maintains positive definiteness in the approximation of the Hessian.

### Limitations

-Computational cost increases with dimensionality due to matrix-matrix and matrix-vector operations.

### Results:

```
Optimal solution: [1. 1.]

Function value at optimum: 1.0095170856086831e-18

Number of iterations: 18
```

## Section 3:

**a) Rosenbrock's Parabolic Valley Function**

The Rosenbrock function is a non-convex function widely used to test optimization algorithms due to its challenging narrow valley shape.

**f(x1,x2) = 100(x2−x1^2)^2  + (1−x1)^2**

**Initial Point**: X0=(−1.2,1.0)

- **Results for Rosenbrock Function**

| Method | Iterations | Optimal Solution | Optimal Value | CPU Time (s) |
|---|---|---|---|---|
| Fletcher-Reeves CG | 41 | [1.00000291, 1.00000585] | 8.563×10^−12 | 0.018179 |
| Marquardt Method | 41 | [0.99999998, 0.99999996] | 4.866×10^−16 | 0.003407 |
| Quasi-Newton (BFGS) | 18 | [1.00000000, 1.00000000] | 1.010×10^−18 | 0.007212 |

**Analysis:**

**Number of Iterations:** Concerning the quasi-Newton method, it suffices to mention here that the iterative solution for the BFGS case demonstrates fewer iterations and has faster convergence with only 18 iterations rather than 41 iterations that were required by other methods.

**Optimal Solution:** All achieved optimal points closely approximate the minimum value of the theoretical function given by (1,1) (1,1).

**Optimal Value:** Among all optimization techniques, the quasi-Newton technique is said to be the best since it produces the smallest value of the function.

**CPU Time:** The Marquardt Method is the quickest of them all since it does not take long since it is able to make adjustments of the steps to be taken quickly.

**b) Powell’s Quartic Function**

The Powell’s Quartic function tests algorithms on a complex surface with multiple variables and steep gradients.

Function:

$f(x1,x2,x3,x4)=(x1+10x2)^2+5(x3-x4)^2+(x2-2x3)^4+10(x1-x4)^4$

Initial Point: $X0=(3.0,-1.0,0.0,1.0)$

**Results for Powell’s Quartic Function**

| Method | Iterations | | Optimal Value | CPU Time (s) |
|---|---|---|---|---|
| **Fletcher-Reeves CG** | **1000** | | **9.061×10^−10** | **0.341768** |
| **Marquardt Method** | **100** | | **2.093×10** | **0.004395** |
| **Quasi-Newton (BFGS)** | **32** | | **1.167×10^−10** | **0.008371** |

**Analysis:**

**The Total Number Of Iterations**: With respect to the number of iterations required, the BFGS and quasi Newton method beats Fletcher-Reeves CG by 68 iterations taking them down to only 32 iterations.

**The Optimal Solution**: There is some difference in the solutions because of exponentials in Powells's complex structure, but Quasi Newton is observed to show the best performance in the area of convergence.

**The Optimal Valu**e: Out of all methods, the one which comes closest to achieving a result that is most accurate is the quasi Newton method with some of its values attaining zero.

**CPU Time**: In relation to calculation cost doing Fletcher-Reeves CG is costly, venturing with the Marquardt method works to be the quickest instead.

## Comparison of Methods

### Fletcher-Reeves Conjugate Gradient Method

- To begin with, its modular structure makes it appropriate for large-scale use.
- In addition, it does not involve the explicit computation of the Hessian matrix: only the gradient evaluation is involved.
- Of course, it has certain drawbacks as well. First of all, it is extremely susceptible to the precision of the line search.
- It is also able to show optimal results for quadratic functions, as it manages to achieve quadratic convergence.
- Last but not least, it requires a smooth objective function that is continuously twice-differentiable.

### Marquardt Method

- In my view, the greatest benefit is that, while relying solely on the limitations of steepest descent and Newton‘s method, it brings the two together.
- Also noteworthy, when faced with invalid or ill-conditioned Hessians, the author makes use of regularization.
- Likewise, it allows tailoring the step to the size required and therefore enhances convergence.
- Nonetheless, it certainly has all the flaws noted above in the case of the Fletcher-Reeves Conjugate Gradient Method.
- Furthermore, they are extraordinarily meticulous due to a degree of conditionality of the Hessian matrix.

### Quasi-Newton (BFGS) Method

- This is certainly one of the most intense forms of the BFGS graph - near the optimal solution it achieves superlinear convergence.
- On the contrary, it is common for them to utilize some iteration process whereby either the BFGS or simply gradient increases and progressively adjusts itself to better meet the expected outcomes.
- More importantly, various classes of problems appear to retain their reliability and efficiency.
- Finally, in regard to scaled functions, the above figure makes it obvious that they are not ideal for many functions.
- Remembering that the greater dimensional aspects are combined with matrix operations, they can very easily inflate computation costs.

**Rosenbrock's Function**

**Features**

1. **Dimensionality:**
    - **A 2-dimensional function involving two variables**
2. **Shape:**
    - **Forms a narrow, curved valley with the global minimum inside the valley. This shape often slows the convergence of optimization algorithms.**
3. **Global Minimum:**
    - **The function reaches its global minimum at x=(1,1) where f(x)=0**
4. **Complexity:**
    - **Although the function is nonlinear and non-convex, it is less complex than Powell's Quartic function.**
5. **Testing Use:**
    - **Used as a benchmark to test how well optimization algorithms navigate narrow regions and converge to minima.**

**Powell's Quartic Function**

**Features**

1. **Dimensionality:**
    - **A 4-dimensional function involving four variable**

2. **Shape:**
   - **Exhibits a more complex surface with steep gradients and quartic terms, making it harder for optimization algorithms to converge efficiently.**
3. **Global Minimum:**
   - **The function achieves its global minimum at x=(0,0,0,0)**
4. **Complexity:**
   - **Significantly more complex than the Rosenbrock function due to the combination of quadratic, quartic, and interaction terms.**
5. **Testing Use:**
   - **Used to evaluate optimization algorithms' performance on higher-dimensional and highly nonlinear problems.**