Optimization Project

Alice Devilder*, Kanupriya Jain†

The Steepest Descent Method

April 16, 2024

Abstract

This project focuses on the implementation and analysis of the steepest descent method for finding minima of two well-known mathematical functions: the Rosenbrock's function and the Himmelblau's function. The steepest descent method is a widely-used optimization algorithm that iteratively moves towards the direction of the negative gradient to find the local minima of a function.

The report will provide a valuable resource for understanding the behavior and performance of the steepest descent method in the context of optimization problems.

*alice.devilder@gmail.com

†kanupriya2406@gmail.com

Contents

1 Rosenbrock's and Himmelblau's functions and their gradients

As part of this project, we implemented the steepest descent method to minimize *Rosenbrock's function*

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

and *Himmelblau's function*

$$g : \mathbb{R}^2 \to \mathbb{R}, \quad g(x_1, x_2) = (x_2^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

For this purpose, we computed their gradients that are respectively:

$$\nabla f(x_1, x_2) = \begin{pmatrix} -2(1 - x_1) - 400x_1(x_2 - x_1^2) \\ 200(x_2 - x_1^2) \end{pmatrix}$$

and

$$\nabla g(x_1, x_2) = \begin{pmatrix} 4x_1(x_1^2 + x_2 - 11) + 2(x_1 + x_2^2 - 7) \\ 2(x_1^2 + x_2 - 11) + 4x_2(x_1 + x_2^2 - 7) \end{pmatrix}$$

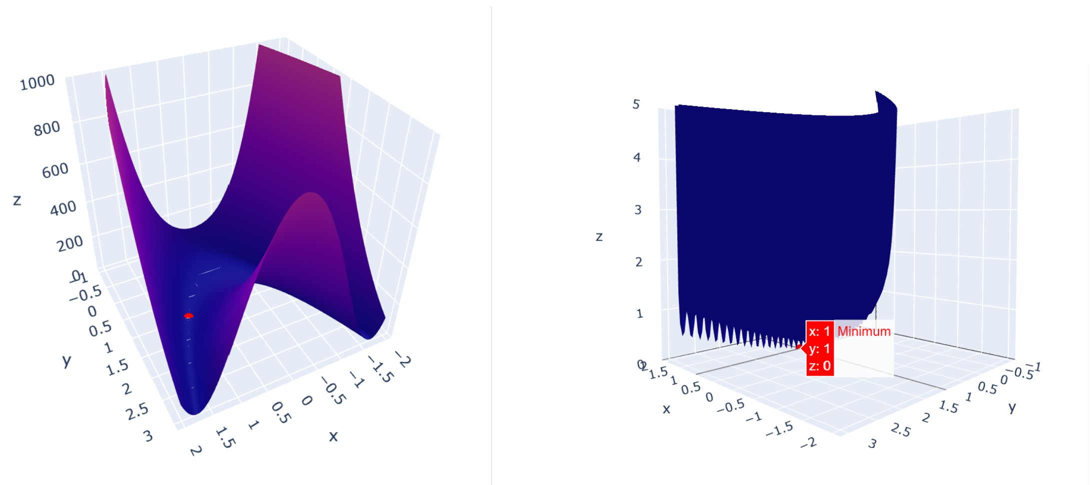
2 Intuition of the minima on the plots of the functions

In order to have an intuition of how many minima there are and their positions we made plots of the behaviour of the *Rosenbrock's* and the *Himmelblau's* functions.

2.1 *Rosenbrock's* function

For the *Rosenbrock's* function, we first plot the graph of the function with a wide range for $f(x_1,x_2)$. Then we noticed that it was a bit difficult to see where was the minimum of the function so as we knew that the minimum was attained at $f(\bar{x}) = 0$, we restrained the axis z between 0 and 5.

In this way, we see that there is a unique minimum attained for $x = (1, 1)$.



(a) Overview

(b) Zoom on the minimum

Figure 1: 3D plot of the *Rosenbrock's* function

Remark : We knew that the minimum of f was $f(\bar{x}) = 0$ because the function is strictly positive and there exists $\bar{x}$ such that $f(\bar{x}) = 0$ so $\bar{x}$ is the global minimum of the *Rosenbrock's* function.

2.2 *Himmelblau's* function

Concerning the *Himmelblau's* function, we observe on the plot that this function have 4 minima. Similar to the *Rosenbrock's* function, the *Himmelblau's* function is strictly positive so the minimum is 0 because it is attained. Here, we see that there are 4 differents values for $\bar{x} = (x_1, x_2)$ that satisfy the relation $g(\bar{x}) = 0$.

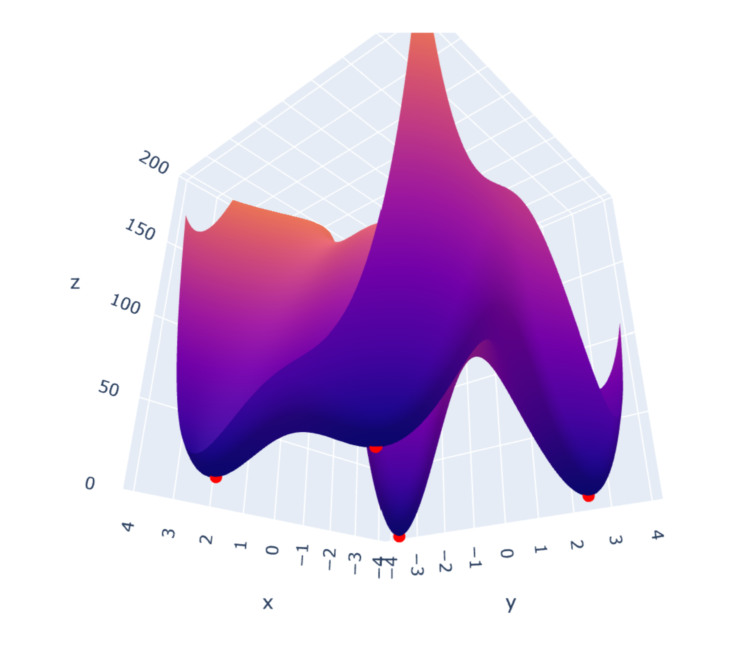


Figure 2: 3D plot of the *Himmelblau's* function

The minima are $x_0 = (3, 2)$, $x_1 = (-2.805118, 3.131312)$, $x_2 = (-3.779310, -3.283186)$, $x_3 = (3.584458, -1.848126)$.

3 The algorithms to find the optimal stepsize

3.1 The *Goldstein-Armijo* algorithm

Algorithm 1 The *Goldstein-Armijo* algorithm

Require: let $\sigma > 0$, $0 < \beta_1 \leq \beta_2 < 1$
  $\delta \leftarrow 0.001$ ▷ We can also take random value $\delta \in (0,1)$
  $\tau_0 \leftarrow -\sigma \frac{\nabla f(x)^T d}{||d||^2}$
  $j \leftarrow 0$
  while $j <$ max_iterations **do** ▷ max_iterations = 10000
    if $f(x + \tau_j d) \leq f(x) + \delta \tau_j \nabla f(x)^T d$ **then**
      $t \leftarrow \tau_j$
      break
    else
      $\tau_{j+1} \in [\beta_1 \tau_j; \beta_2 \tau_j]$
      $t \leftarrow \tau_{j+1}$
      $j \leftarrow j + 1$
    end if
  end while
  return t

At first, we took δ as a random value between 0 and 1 (we decided to call the function *random()* in the algorithm instead of choosing by ourselves a random value), but finally to have consistent results, we decided to take $\delta = 0.001$.

We decided to put default values for σ, β_1, β_2 and $max\_iterations$ that are $\sigma = 1e-3, \beta_1 = 1e-3, \beta_2 = 0.8, max\_iterations = 10000$ respectively.

3.2 The *Wolfe-Powell* algorithm

Algorithm 2 Wolfe Powell Step Size Algorithm

Require: f: Objective function, x: Current point, $grad$: Gradient function, $0 < \delta < \beta < 1$

Ensure: $stepsize$: Final step size

```
stepsize ← 1.0                                        ▷ Initial step size
d ← −grad(x)                                          ▷ Initial search direction
iteration ← 0
while iteration < max_iterations do
    f_x ← f(x)
    grad_x ← grad(x)
    x_next ← x + stepsize × d
    armijo_condition ← f(x_next) ≤ f_x + δ ×stepsize × ⟨grad_x, d⟩
    curvature_condition ← ⟨grad(x_next), d⟩ ≥ β × ⟨grad_x, d⟩
    i ← iteration
    if armijo_condition and curvature_condition then
        break
    end if
    result ← line search(f, grad, x, d)
    if not result[0] then
        break
    end if
    stepsize ← result[0]
    iteration ← iteration + 1
end while
    return stepsize
```

We decided to use the line search algorithm to adjust the step size. If it is evaluated to be false i.e. the line search did not find a suitable step size, then we decided to break out of the loop.

We decided to put default values for δ, β and $max\_iterations$ that are $\delta = 1e-4$, $\beta = 0.8$, $max\_iterations = 10000$ respectively.

4 The Steepest Descent method with a constant stepsize

4.1 The Steepest Descent algorithm

Algorithm 3 Steepest Descent Algorithm

Require: f: Objective function, *gradient*: Gradient function, *initial_point*: Initial point, *step_size*: Step size, *max_iterations*: Maximum number of iterations, ϵ: Convergence tolerance

Ensure: x: Final point

$x \leftarrow initial\_point$
for *iteration* **in** *range*(*max_iterations*) **do**
    $gradient\_at\_x \leftarrow gradient(x)$
    $x\_next \leftarrow x - step\_size \times gradient\_at\_x$
    $i \leftarrow iteration$
    if $\|x\_next - x\| < \epsilon\_$ or $\|gradient\_at\_x\| < \epsilon\_$ **then**
        break
    end if
    $x \leftarrow x\_next$
end for
      return x

In this algorithm, we are taking our descent direction as $-\nabla$f(x).
In practice, instead of taking ∇f(x) = 0 as stopping criterion, one uses $\|x^{k+1} - x^k\| < \epsilon$ or $\nabla f(x^k) < \epsilon$ for an admissible error $\epsilon > 0$ as the stopping criterion.

We initially took some random step sizes and calculated minimizers of the functions with initial points -

$$x_0 = [0,0]^T$$

$$\tilde{x} = [\pi + 1,\pi - 1]^T$$

Then we tried to compute optimal stepsizes using **Goldstein-Armijo algorithm** and **Wolfe-Powell algorithm** and applied the steepest descent method to obtain the minimizers of the functions with the same initial points x_0 and $\tilde{x}$.

4.2 Results with the *Rosenbrock's* functions

We are taking our constant step sizes as 0.001, 10 and 1e-6.

For Rosenbrock Function

Time taken	No. of iterations	Step Size	Initial point	Minimizer	Minimized Value
12.3933	8313	0.001	[0 0]	[0.98891964 0.97791742]	0.000122974
0.00608921	4	0.001	[4.14159265 2.14159265]	[2.02511517e+30 2.45208772e+06]	1.6819e+123
0.00500774	3	10	[0 0]	[1.31076424e+26 2.01587193e+18]	2.95188e+106
0.00368524	2	10	[4.14159265 2.14159265]	[6.15595349e+19 1.23743787e+14]	1.43609e+81
0.00127149	0	1e-06	[0 0]	[0 0]	1
2.94393	2249	1e-06	[4.14159265 2.14159265]	[1.61943175 2.60959641]	0.400499

Figure 3: Results for *Rosenbrock's Function*

- We observe that with initial point (0,0), the steepest descent algorithm converges without reaching the max iterations and gives us a minimizer close to the global minimizer of the function. However, when the initial point is $(\pi+1, \pi-1)$, the algorithm doesn't converge.
- With step size 10, the algorithm doesn't seem to converge with any given initial point. Indeed, this stepsize is way too large.
- For step size 1e-06 starting from (0,0), we can notice that the algorithm didn't do any iteration. Indeed, we took $x_{next} = x + stepsize * p$ and one of the stopping criterion is $||x_{next} - x|| < \epsilon$ but in our algorithm $\epsilon = 1e-05$ so this criterion is satisfied from the iteration 0. Therefore we got a minimizer equals to the initial point. In this case, the stepsize is too small.

  When initiated from $(\pi+1, \pi-1)$, it yields a minimizer that closely approximates the true minimum but it still stops too early because the stepsize is too small.

Then we computed stepsizes using the Goldstein-Armijo algorithm and Wolfe-Powell algorithm.

Stepsizes calculated using GOLDSTEIN-ARMIJO ALGORITHM

Function	Step Size	iterations of goldstein algorithm	Total time of goldstein algorithm	iterations of steepest descent	Total time of steepest descent	Initial point	Minimizer	Minimized Value
Rosenbrock	0.001	0	0.00641418	8313	12.1177	[0 0]	[0.98891964 0.97791742]	0.000122974
Rosenbrock	0.000240007	4	0.00766444	9999	14.289	[4.14159265 2.14159265]	[0.4695888 0.21785963]	0.28204

Figure 4: Results for *Rosenbrock's Function* with step sizes computed using Goldstein Armijo Algorithm

- We can observe that Goldstein Armijo gives us a good step size for which the Steepest Descent algorithm converges and gives us a good approximation to the minimizer. We can still notice that for $\tilde{x}$ as the initial point, the algorithm has stopped because it has reached the maximum iteration number. That's why it doesn't give us the global minimizer of the Rosenbrock's function. Maybe if we had increased the maximum number of iterations, it would have converged to approximatively $(1,1)$.

Stepsizes calculated using WOLFE POWELL ALGORITHM

Function	Step Size	iterations of wolf powell algorithm	Total time of wolfe powell algorithm	iterations of steepest descent	Total time of steepest descent	Initial point	Minimizer	Minimized Value
Rosenbrock	0.0867732	1	0.0145817	6	0.00883007	[0 0]	[-3.17576254e+34 3.44502464e+07]	1.01717e+140
Rosenbrock	0.000328628	1	0.00756931	9999	13.5973	[4.14159265 2.14159265]	[1.62015883 2.62666219]	0.384902

Figure 5: Results for *Rosenbrock's Function* with step sizes computed using Wolfe Powell Algorithm

- The Steepest Descent algorithm fails to converge when using the step size determined by the Wolfe-Powell Algorithm for the initial point (0,0). Indeed, here the Wolfe Powell algorithm gave us a stepsize equals to 0.0868 which is too large for the Steepest Descent algorithm to converge.

  However, it exhibits convergence for the initial point $(\pi+1, \pi-1)$ as the $stepsize = 0.00033$, providing a satisfactory approximation to the minimizer. We could still have a better result if we had increased the maximum number of iterations as the algorithm has stopped because it has reached this maximum number of iterations.

4.3 Results with the *Himmelblau's* functions

We are taking our constant step sizes as 0.001, 10 and 1e-6.

For Himmelblau Function

Time taken	No. of iterations	Step Size	Initial point	Minimizer	Minimized Value
0.445648	319	0.001	[0 0]	[2.99985444 2.00035126]	1.8592e-06
0.333325	244	0.001	[4.14159265 2.14159265]	[3.00014674 1.99964557]	1.89176e-06
0.00505996	3	10	[0 0]	[5.60481042e+25 3.12483275e+27]	9.5347e+109
0.00507331	3	10	[4.14159265 2.14159265]	[-5.44115973e+34 1.15516563e+09]	8.76528e+138
14.4771	9999	1e-06	[0 0]	[0.17290899 0.25033464]	160.671
14.4255	9999	1e-06	[4.14159265 2.14159265]	[3.410996 1.97595442]	6.916

Figure 6: Results for *Himmelblau's Function*

- With stepsize 0.001, we can observe that the Steepest Descent algorithm converges to one of the minimizer of the Himmelblau's function (3,2) from both x_0 and $\tilde{x}$. It is also interesting to notice that they converge to this minimizer in the opposite direction (one is below 3 for x_1 and above 2 for x_2 and for the other one, it is the opposite).
- For stepsize 10, for the same reason as before (as the stepsize is too large), the Steepest Descent algorithm doesn't converge from any initial point.
- Despite the convergence of the algorithm with a step size 1e-6, it provides a poor approximation to the function's minimizer for both specified initial points because in the both case, it reaches the maximal iteration number.

Then we computed step size using the Goldstein-Armijo algorithm and Wolfe-Powell algorithm.

Stepsizes calculated using GOLDSTEIN-ARMIJO ALGORITHM

Function	Step Size	iterations of goldstein algorithm	Total time of goldstein algorithm	iterations of steepest descent	Total time of steepest descent	Initial point	Minimizer	Minimized Value
Himmelblau	0.001	0	0.00266957	319	0.420904	[0 0]	[2.99985444 2.00035126]	1.8592e-06
Himmelblau	0.001	0	0.00253916	244	0.335769	[4.14159265 2.14159265]	[3.00014674 1.99964557]	1.89176e-06

Figure 7: Results for *Himmelbalu's Function* with step sizes computed using Goldstein Armijo Algorithm

- It can be noted that the Goldstein-Armijo rule yields an effective step size in this case as well, leading to convergence of the steepest descent algorithm and providing a good approximation to the minimizer. We can observe that we did not reach the maximum iterations and all the stopping criterion are satisfied before reaching before that. We can observe that although Himmelblau's function has four minimizers, in both cases, Goldstein Armijo converges towards the same minimizer i.e. (3,2).

Stepsizes calculated using WOLFE POWELL ALGORITHM

Function	Step Size	iterations of wolf powell algorithm	Total time of wolfe powell algorithm	iterations of steepest descent	Total time of steepest descent	Initial point	Minimizer	Minimized Value
Himmelblau	0.133341	1	0.00770044	6	0.00923967	[0 0]	[4.52977315e+41 -6.70080805e+08]	4.21023e+166
Himmelblau	0.0186449	1	0.00893044	18	0.0241387	[4.14159265 2.14159265]	[2.99999101 2.00000753]	2.60186e-09

Figure 8: Results for *Himmelbalu's Function* with step sizes calculated using Wolfe Powell Algorithm

- Here we can notice that for x_0, the stepsize is equal to 0.1333 which is too large for the Steepest Descent algorithm to converge.
- For $\tilde{x}$, the algorithm gives a really great approximation of the minimizer within only 18 iterations!

5 The Steepest Descent method with variable step size

5.1 The Steepest Descent algorithm

Algorithm 4 Steepest Descent Algorithm

Require: f: Objective function, $gradient$: Gradient function, $initial\_point$: Initial point, $step\_size\_algo$: Algorithm to calculate the stepsize at each step, $max\_iterations$: Maximum number of iterations, ϵ: Convergence tolerance

Ensure: x: Final point

$x \leftarrow initial\_point$
for $iteration$ **in** $range(max\_iterations)$ **do**
  $gradient\_at\_x \leftarrow gradient(x)$
  $step\_size \leftarrow step\_size\_algo(f, x, gradient\_at\_x)$ # stepsize calculated using given algorithm
  $x\_next \leftarrow x - step\_size \times gradient\_at\_x$
  if $\|x\_next - x\| < \epsilon\_$ or $\|gradient\_at\_x\| < \epsilon\_$ **then**
    break
  end if
  $x \leftarrow x\_next$
end for
  return x

Here we are giving step size algorithm as input to calculate stepsize at each step.

5.2 Results with the *Rosenbrock's* functions

Steepest descent calculated using variable stepsizes
For ROSENBROCK FUNCTION

Algorithm	Iterations	total time	Minimizer	Initial point	Minimized Value
Goldstein-Armijo	8313	51.3338	[0.98891964 0.97791742]	[0 0]	0.000122974
Goldstein-Armijo	9999	57.947	[0.98610627 0.97234952]	[4.14159265 2.14159265]	0.00019335
Wolfe Powell	1723	19.8602	[0.99622405 0.99243028]	[0 0]	1.43607e-05
Wolfe Powell	7	0.0808227	[-1.50029614e+43 3.82633962e+09]	[4.14159265 2.14159265]	5.0665e+174

Figure 9: Results for *Rosenbrock's Function* with variable step size

- The Steepest Descent algorithm for Rosenbrock's function, employing a variable step size at each iteration, converges for both specified initial points while utilizing the Goldstein-Armijo Algorithm, providing a reasonable approximation. However, when applying the Wolfe-Powell Algorithm, it converges only with the initial point (0,0).
- We can observe that it doesn't converge for the initial point $(\pi + 1, \pi - 1)$. It might be because we are getting a large step size or the failure of the line search process within the Wolfe Powell algorithm and it breaks out of the loop. Similar to a previous case where we used a step size of 10 resulting in algorithm failure to converge, this might also explain why here we obtained a result in only 7 iterations.

5.3 Results with the *Himmelblau's* functions

Steepest descent calculated using variable stepsizes
For HIMMELBLAU FUNCTION

Algorithm	Iterations	total time	Minimizer	Initial point	Minimized Value
Goldstein-Armijo	319	1.72528	[2.99985444 2.00035126]	[0 0]	1.8592e-06
Goldstein-Armijo	244	1.2902	[3.00014674 1.99964557]	[4.14159265 2.14159265]	1.89176e-06
Wolfe Powell	13	0.157202	[2.99999627 1.9999943]	[0 0]	1.49228e-09
Wolfe Powell	10	0.115299	[2.99999808 2.00000479]	[4.14159265 2.14159265]	3.42939e-10

Figure 10: Results for *Himmelblau's Function* with variable step size

- The Steepest Descent algorithm with variable stepsize applied to Himmelblau's function exhibits convergence for both given initial points when employing both the Goldstein-Armijo Algorithm and the Wolfe-Powell Algorithm. This algorithm yields a very good approximation for the minimizer. Despite the Himmelblau's function having four minimizers, we can observe that regardless of the chosen initial point, both algorithms converge towards the same minimizer, i.e. (3,2).

6 Conclusion

To conclude, our exploration of the Steepest Descent algorithm in the context of optimizing both the Rosenbrock's and Himmelblau's functions has provided valuable insights. Notably, the implementation of a variable step size has proven to be particularly effective, showcasing enhanced performance for Himmelblau's function. It seems to give us better results then the algorithm with a constant step size.

The sensitivity to step size is evident, as an excessively large step size hinders the convergence of the algorithm. The integration of the Goldstein-Armijo algorithm consistently produces improved results, highlighting its importance in optimizing both functions. It has performed better than the Wolfe-Powell algorithm, as expected (since the professor told us during the lecture session that the Goldstein-Armijo algorithm is more efficient).

Furthermore, initiating the algorithm from the point (0,0) significantly expedites the convergence process. Although, Himmelblau's function has four global minimizers, it is worth noting that all the simulations converge to a singular minimizer which is (3,2).

These findings collectively contribute to a comprehensive understanding of the Steepest Descent algorithm's efficacy in tackling complex optimization tasks, offering practical insights for its application in finding the minimum for both the Rosenbrock's and Himmelblau's functions.