Question 2.1

Solution (a) Newton solver is coded in the python script the convergence tolerance was kept 10^{-5}.

The value of **E = 2.154785293118822** after 6 iteration.

The maximum precision in the 64-bits to represent numbers in 16 decimal digits.

Solution (b) Fixed point Jacobi iteration are coded in the python script.

The value of **E = 2.1547852930807987** after 25 iterations.

Solution (c) For comparing the number of iterations and the rate of convergence we will keep the initial guess $E_0 = 10$.

Method	Iteration	Root
$Newton$	6	2.1095534574169883
$Fixed\ point$	25	0.9999996761547266

Solutions (d) To evaluate the robustness, we start at different initial guess and count the number of iterations and check if iterative solver converges to the same optimum.

For Newton:

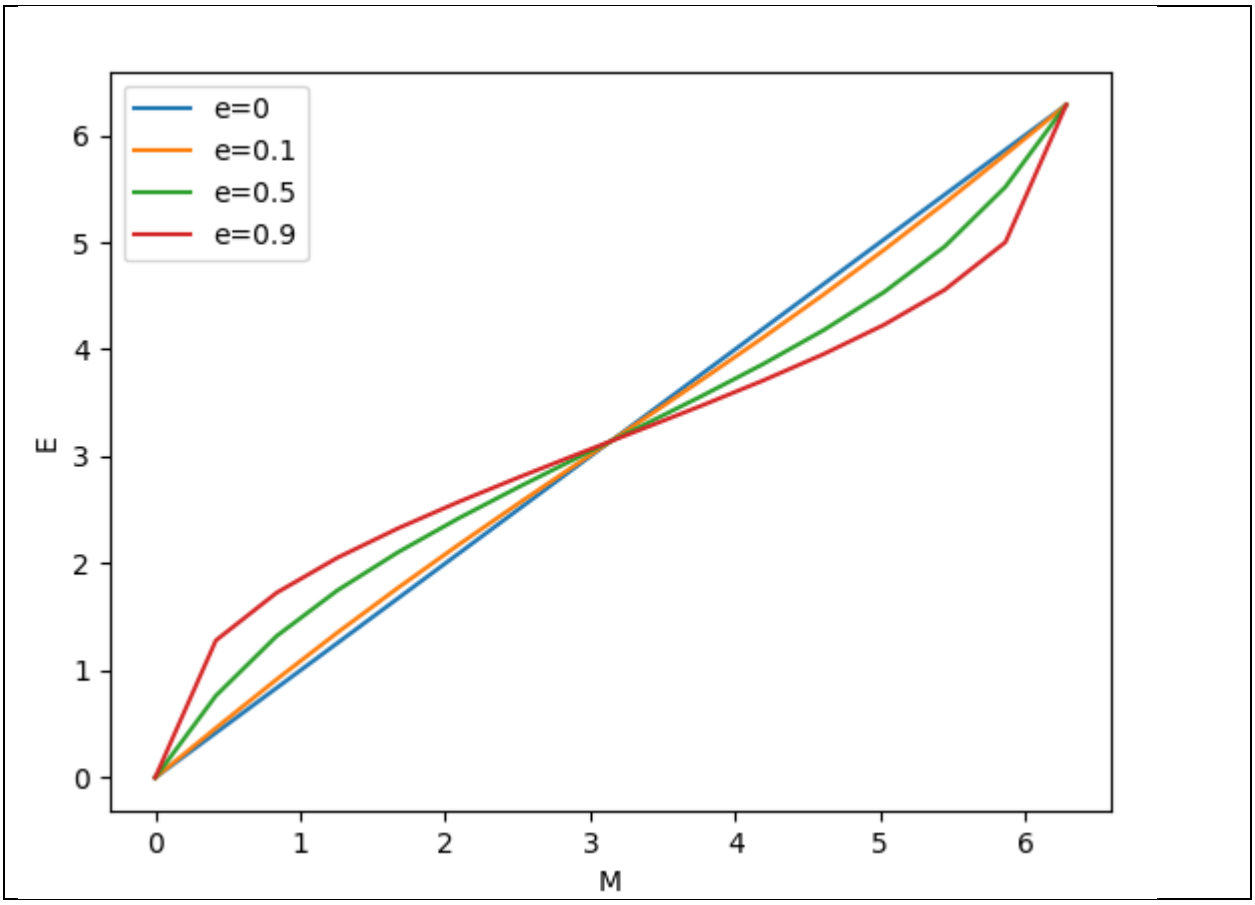
Start	Iteration	Root
$E_0 = 5$	8	2.154785293101842
$E_0 = 7$	9	2.154785293101842
$E_0 = 9$	7	2.1547852931018423
$E_0 = 10$	6	2.154785293118822

For Fixed point:

Start	Iteration	Root
$E_0 = 5$	25	2.154785293101842
$E_0 = 7$	25	2.1547852931018423
$E_0 = 9$	26	2.154785293088159
$E_0 = 10$	25	2.1547852930807987

Solution(e)

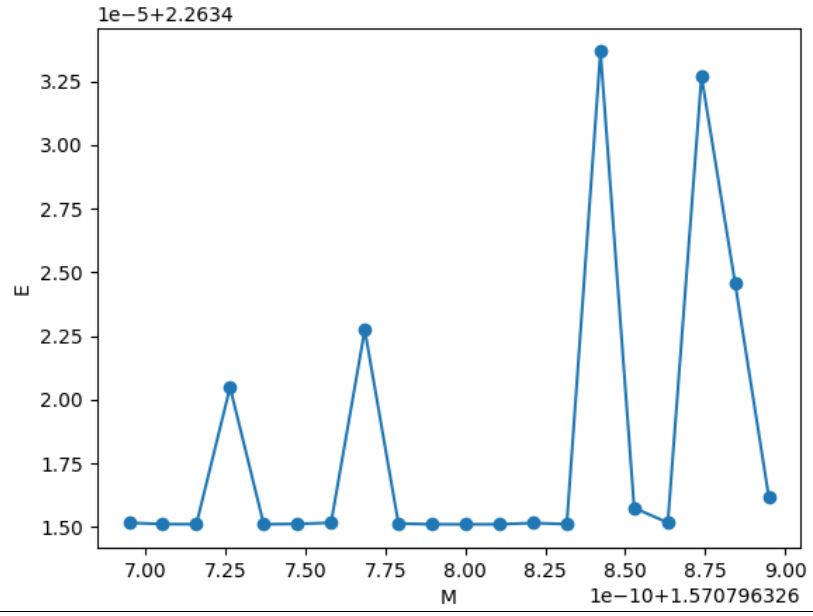
The plot of E vs M is shown below:



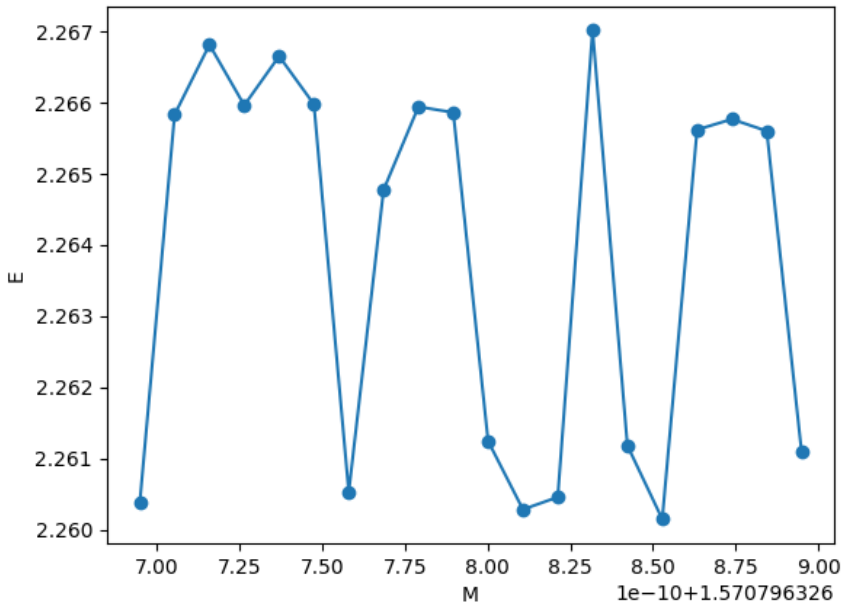
Solution (f):

The plot for numerical noise is shown below, the initial starting point was randomized using NumPy's random.randn(). The mean was taken 1 and the standard deviation was taken 2. e=0.7 and convergence tolerance is 10^{-2}.

The values of M are perturbed about $\pi/2$ by 10^{-10}.



Numerical Noise using newton solver



Numerical noise using Gauss Jacobi

Solution 2_2

We will find the gradient of the give function and set it to zero vector as that's first optimality condition. This will give use the critical points.

$f(x_1, x_2) = x_1^4 + 3x_1^3 + 3x_2^2 - 6x_1x_2 - 2x_2$
$\nabla \boldsymbol{f} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 4x_1^3 + 9x_1^2 - 6x_2 \\ 6x_2 - 6x_1 - 2 \end{pmatrix} = 0$
$4x_1^3 + 9x_1^2 - 6x_2 \dots\dots\dots\dots (1)$ $6x_2 - 6x_1 - 2 \dots\dots\dots\dots (2)$

Using (1) and (2), we get

$4x_1^3 + 9x_1^2 - 6x_1 - 2 = 0$
$x_1 = -\frac{1}{4}, -1 - \sqrt{3}, -1 + \sqrt{3}$

Substituting in (2)

$x_2 = \frac{1}{12}, -\frac{2}{3} - \sqrt{3}, -\frac{2}{3} + \sqrt{3}$
$\rightarrow \boldsymbol{Xa} = \left(-\frac{1}{4}, \frac{1}{12}\right), \boldsymbol{Xb} = \left(-1 - \sqrt{3}, -\frac{2}{3} - \sqrt{3}\right), \boldsymbol{Xc} = \left(-1 + \sqrt{3}, -\frac{2}{3} + \sqrt{3}\right),$

Now calculating the Hessian,

$H(x_1, x_2) = \begin{bmatrix} 12x_1^2 + 18x_1 & -6 \\ -6 & 6 \end{bmatrix}$
$H(\boldsymbol{Xa}) = \begin{bmatrix} -\frac{15}{4} & -6 \\ -6 & 6 \end{bmatrix}; H(\boldsymbol{Xb}) = \begin{bmatrix} 30 + 6\sqrt{3} & -6 \\ -6 & 6 \end{bmatrix}; H(\boldsymbol{Xc}) = \begin{bmatrix} 30 - 6\sqrt{3} & -6 \\ -6 & 6 \end{bmatrix}$

We find the eigen value of the hessians above:

$eigen\big(H(\boldsymbol{Xa})\big) \sim (8.8, -6.6),$	$eigen\big(H(\boldsymbol{Xb})\big) \sim (41.4, 4.9),$	$eigen\big(H(\boldsymbol{Xb})\big) \sim (21.8, 3.7)$

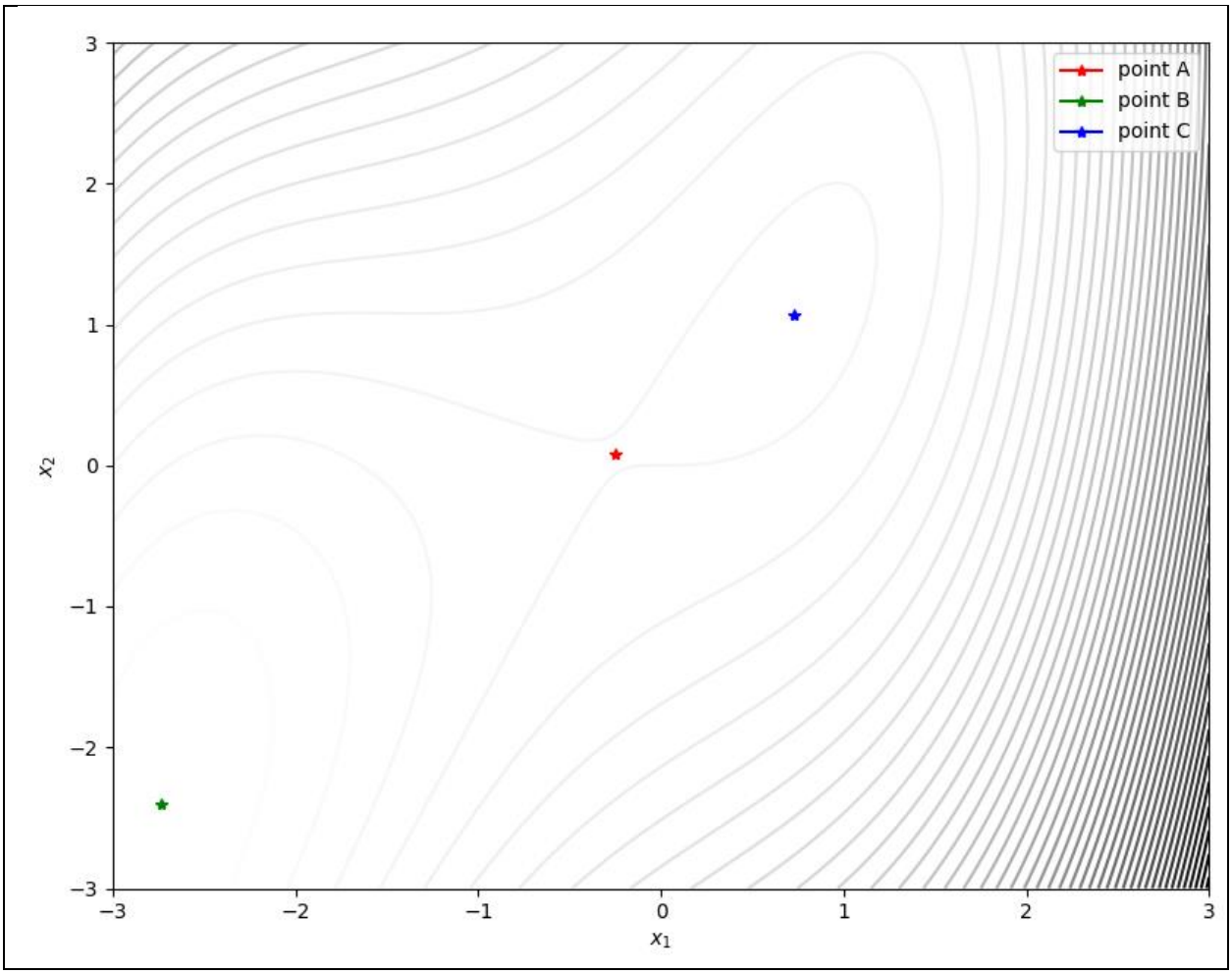
Since, one eigen value is positive and the other is negative for $\boldsymbol{Xa}$**, hence is saddle point.**

Both eigen values are positive for $\boldsymbol{Xb}$**, hence it is a local minima.**

Both eigen values are positive for $\boldsymbol{Xc}$**, hence it is a local minima.**

Since f($\boldsymbol{Xb}$) = -22.725638178746593 < f($\boldsymbol{Xc}$) $= -1.941028487920069$, hence $\boldsymbol{X_b}$ is a global minima.

The plot consolidates this:


point A
point B
point C
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

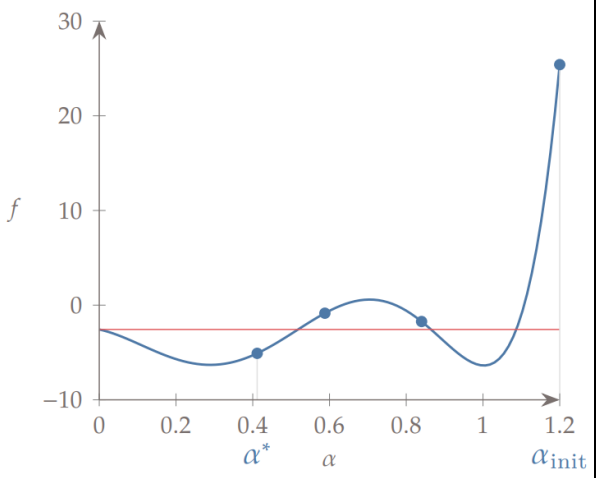
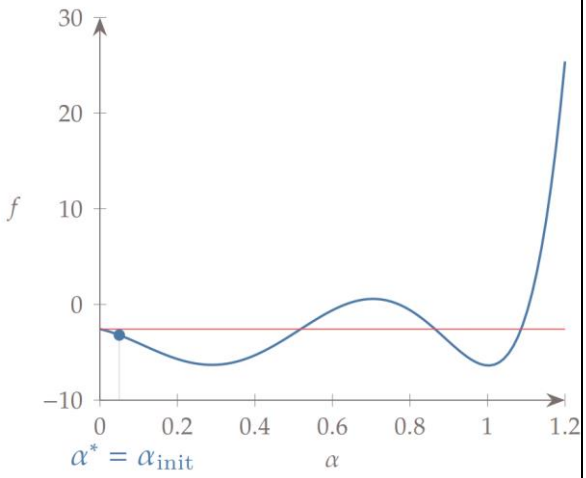
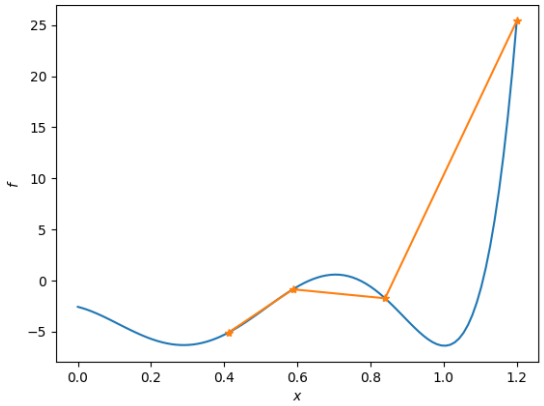
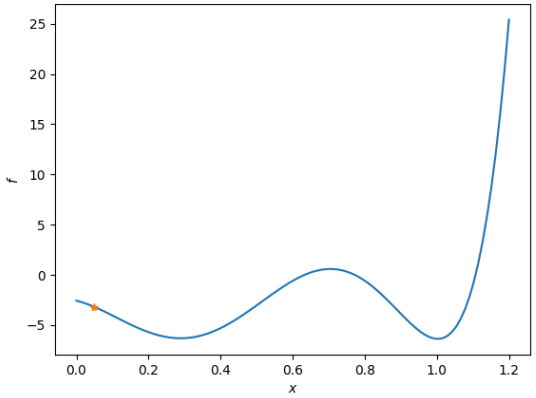
Question 2.3

Solution(a): The two line-search algorithms implemented were:

1. Backtracking – This involved using sufficient decrease condition to perform the line search and uses steepest descent for finding the direction of descent.
2. Bracketing and pinpointing with Quadratic interpolations – This involves satisfying Strong Wolfe condition for line search via pinpointing method. Prior to pinpointing we use the bracketing algorithm to find an interval that contains a point satisfying Strong Wolfe. Steepest descent was used to find the descent directions.

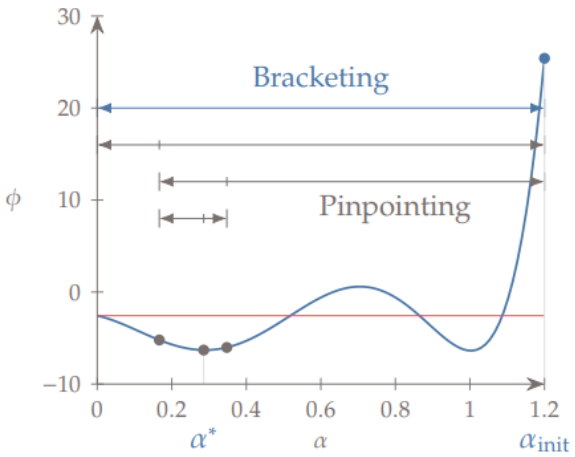
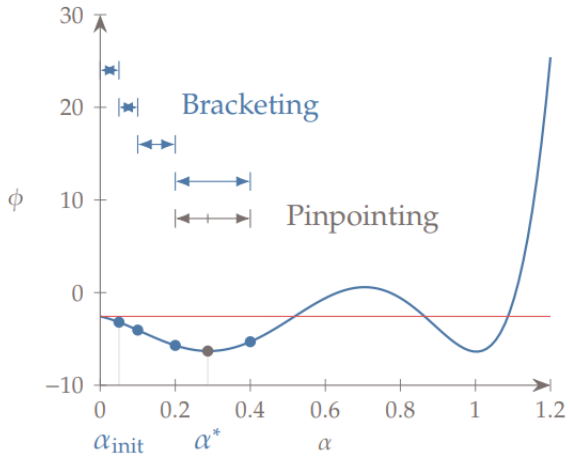
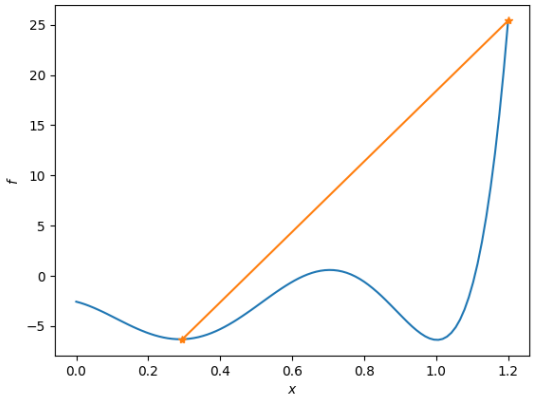
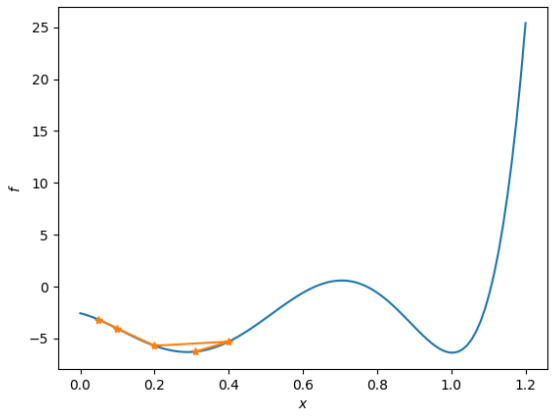
We begin by implementing backtracking algorithm which is tested validated using Examples 4.8 of Engineering Design Optimization book. Here the direction $\hat{p} = [4,0.75]$, $\rho = 0.7$, $\mu_1 = 10^{-4}$ and the $x_0 = [-1.25,1.25]$. We verifying the different α values used by the line search using the available example where the $\alpha_{init} = 0.05\ and\ 1.2$

Backtracking:

Initial $\alpha = 1.2$ (Example from the book)	Initial $\alpha = 1.2$ (Example from the book)
	
Initial $\alpha = 1.2$ (Validation)	Initial $\alpha = 0.05$ (Validation)
	

Bracketing and Pinpointing:

For this case $\mu_2 = 0.9$, $\sigma = 2$, and $\alpha_{init} = 0.05$ *and* 1.2

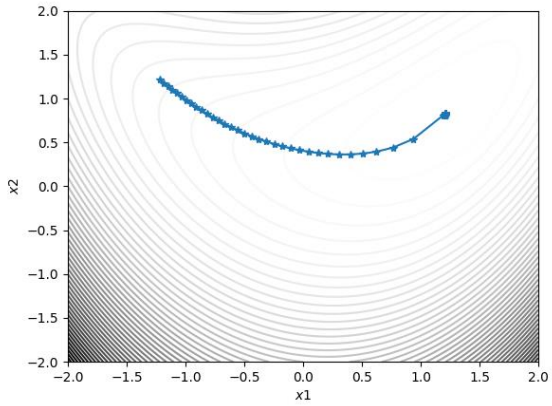
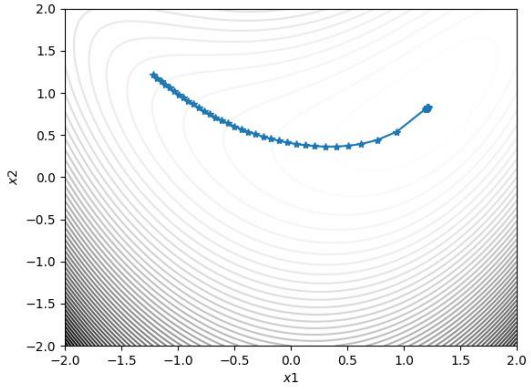
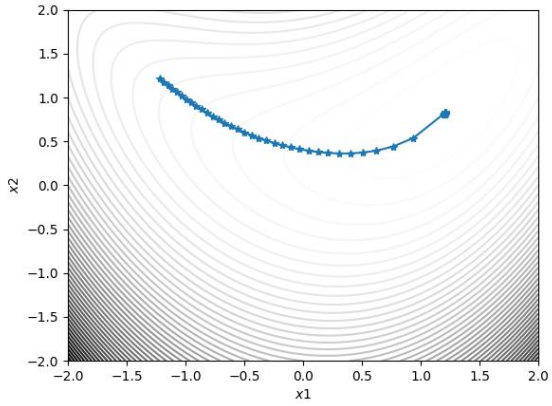
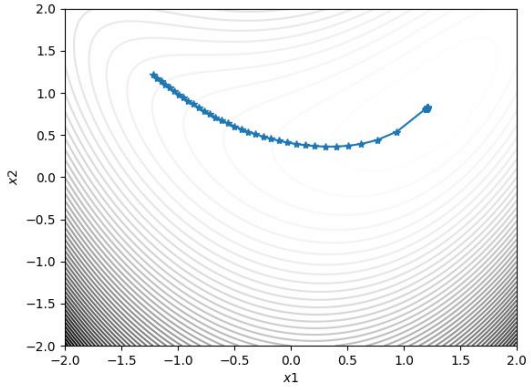
Initial $\alpha = 1.2$ (Example from Book)	Initial $\alpha = 0.05$ (Example from Book)
	
Initial $\alpha = 1.2$ (Validation)	Initial $\alpha = 0.05$ (Validation)
	

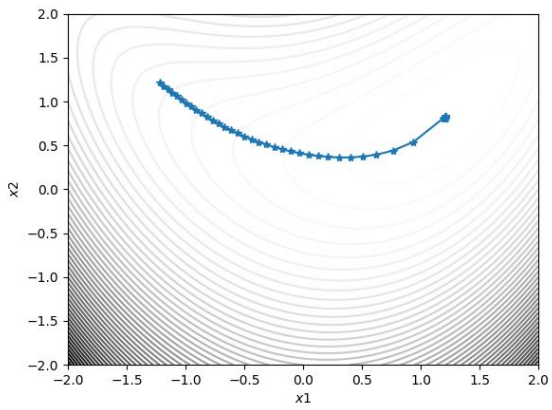
We see that the directional derivative is very close to zero but not exactly zero. The amount of accuracy depends on μ_2 as well. Ideally we want it to be zero.

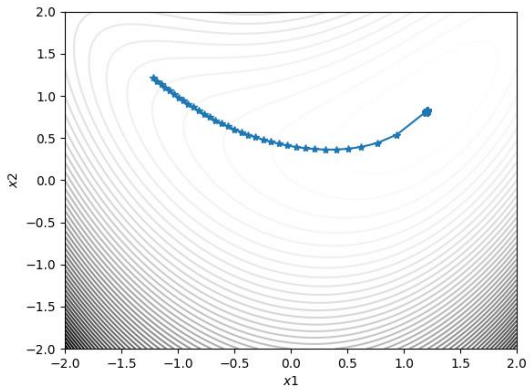
Solution 2(b): Now that we have validated the optimizers developed above, we can run the optimization to minimize the functions using various values of μ_2 and ρ starting with initial step of $\alpha_{init} = 0.05$ from the starting position $x_0 = [-1.25, 1.25]$.

The tolerance τ=10^{-5}, such the exit condition is **while**($\left|\left|\nabla f\right|\right|_\infty > \tau$).

Firstly, we look at steepest descent algorithm with backtracking for line search. For the bean function we plot the contours, and then plot the path traced by the optimizer. The path of the optimizer is shown by blue line with the corresponding position denoted with a '*'.

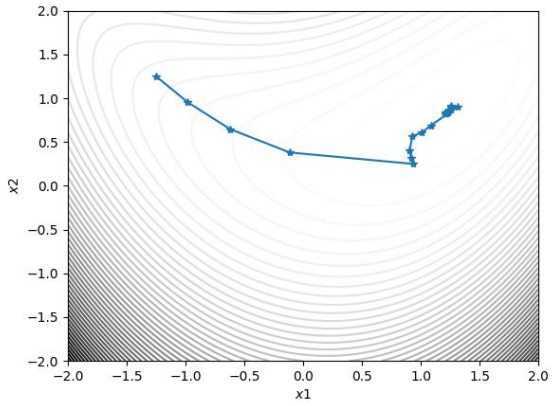
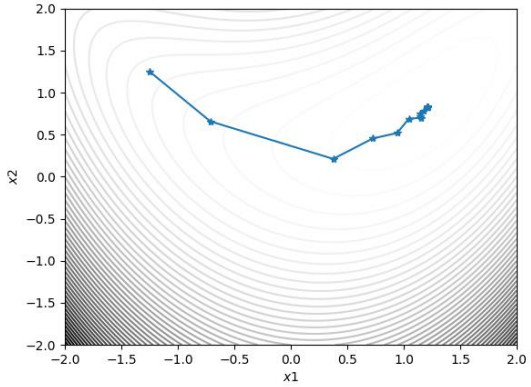
Backtracking, $\rho = 0.9$, iterations = 70	Backtracking, $\rho = 0.7$, iterations =131
	
Backtracking, $\rho = 0.5$, iterations =68	Backtracking, $\rho = 0.3$, iterations =50
	
Backtracking, $\rho = 0.1$, iterations =56	Backtracking, $\rho = 0.001$, iterations =688

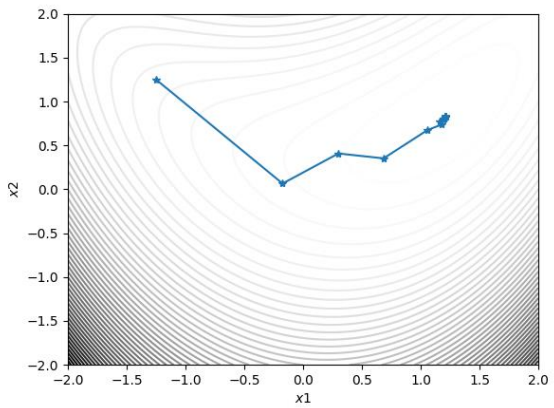




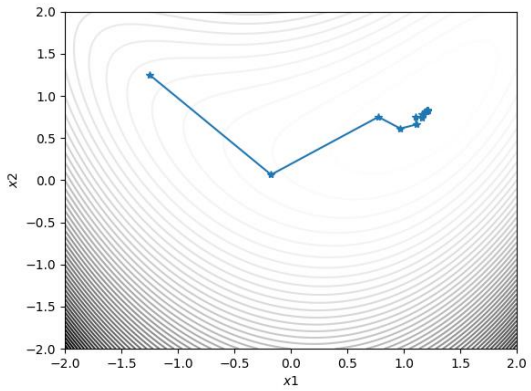
From the above plots it is observed that there is no significant change in the number of iterations needed to converge with ρ, but having a very small value of $\rho = 0.001$ takes an of order of magnitude more iterations to converge to the minimum (~ 700 iterations).

Now we would vary μ_2 and observe the variations:

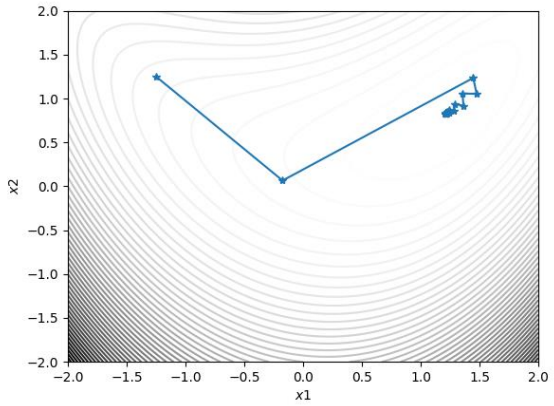
Backtracking, $\mu_2 = 0.9$, iterations = 37	Backtracking, $\mu_2 = 0.7$, iterations =29
	
Backtracking, $\mu_2 = 0.5$, iterations =24	Backtracking, $\mu_2 = 0.3$, iterations =23

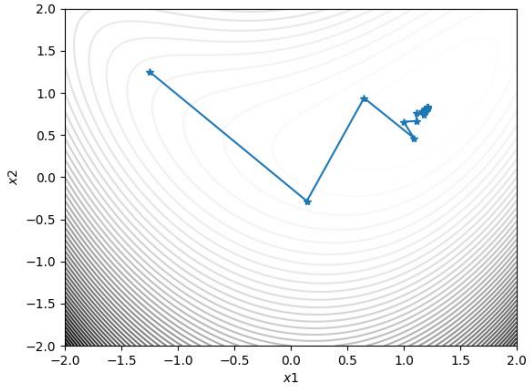


Backtracking, $\mu_2 = 0.1$, iterations =33



Backtracking, $\mu_2 = 0.001$, iterations=38



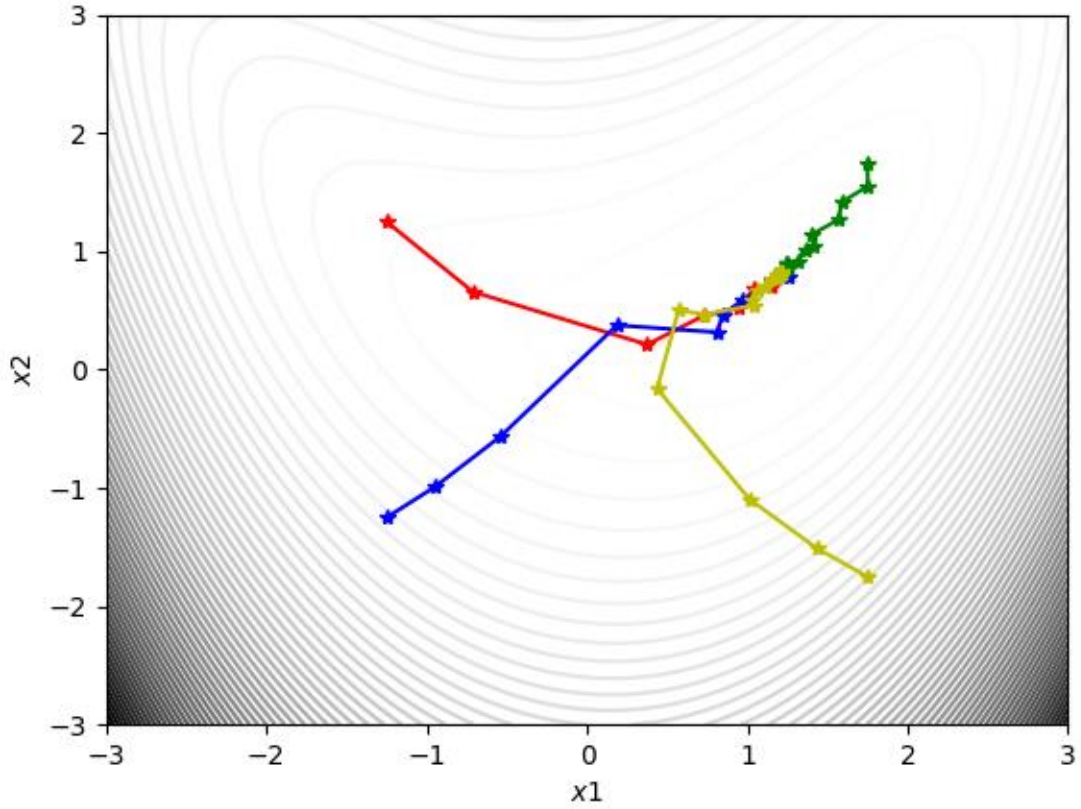


We know as $\mu_2 \to 0$, the sufficient curvature condition tends to $|\phi'(\alpha)| \to 0$. This means that we recover the condition for exact optima. However, upon reducing the value to μ_2 =0.001 we do see a slight increase in the number of iterations to 38 from 23 iteration for $\mu_2 = 0.3$.

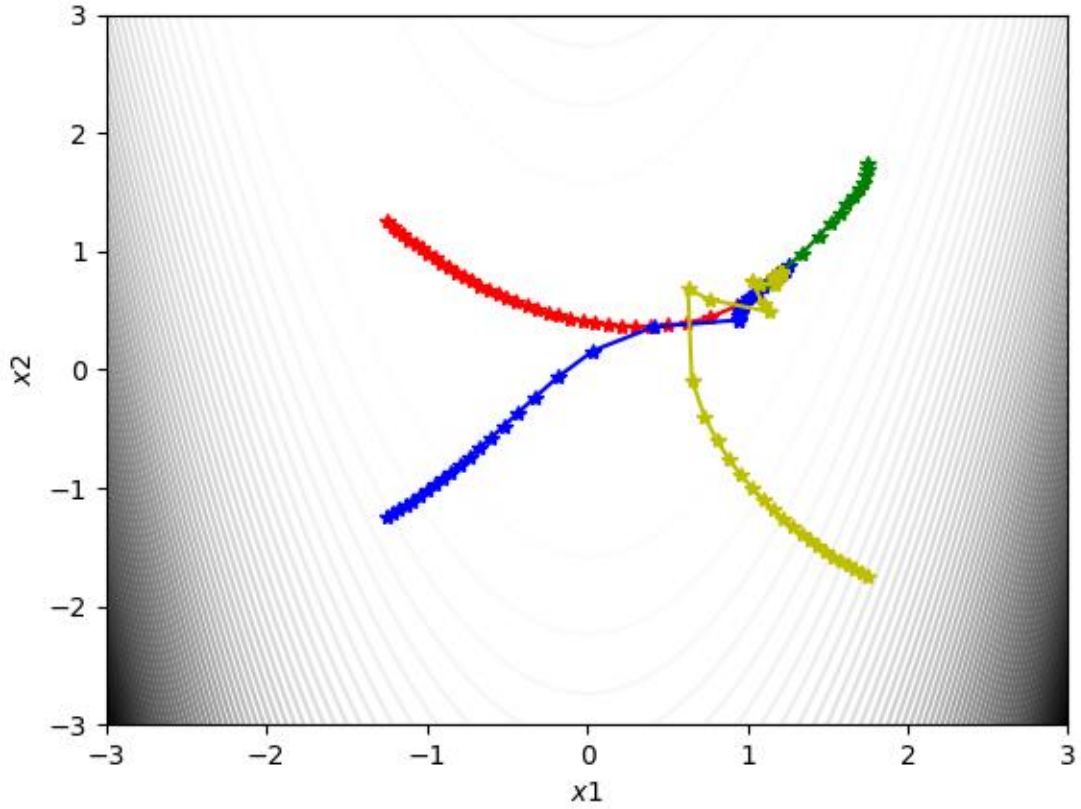
Starting different initial guess, the steepest descent optimizer with bracketing algorithm, converges to the optimum which are pretty close.

Initial Starting point	Optimum Value	Color on the graph
-1.25, 1.25	0.09194381641936278	Red
-1.25, -1.25	0.09194381641966551	Blue
1.75, 1.75	0.0919438164199806	Green
1.75, -1.75	0.0919438164174801	Yellow

The plot tracing the path taken by the optimizer to converge to the optimum value is shown in the figure below:



Bracketing line search with steepest descent



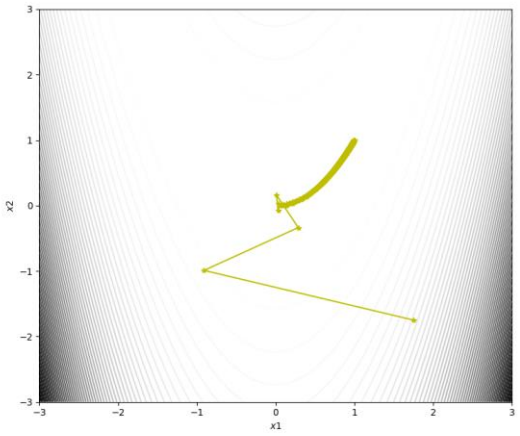
Backtrack with Steepest descent

Solution 2.3(c)

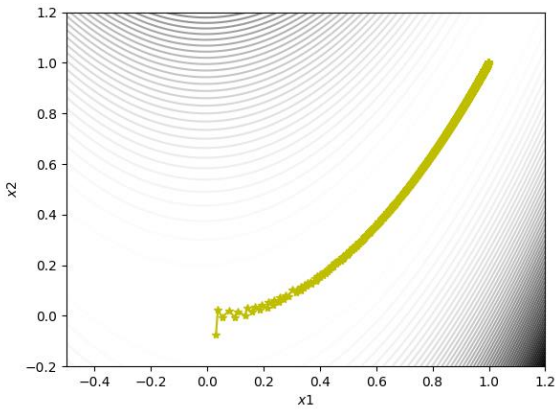
We take the following Rosen Brock function:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

We minimize this using steepest descent using bracketing and pinpoint line search algorithm. The $\alpha_{init} = 0.05$ and $\mu_2 = 0.001$. The Results are shown the figure below:



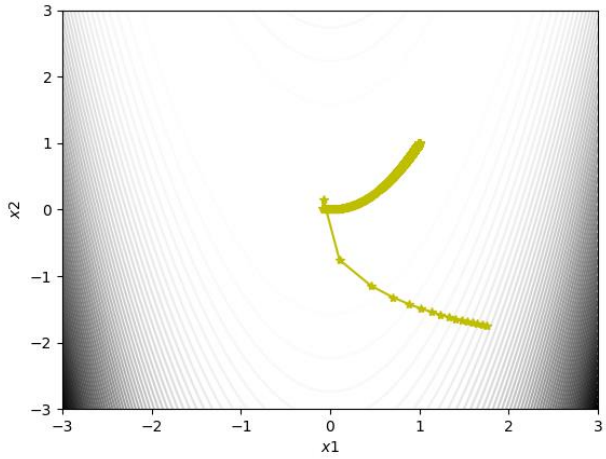
The path of the optimizer



Zoomed in view of the path of the optimizer

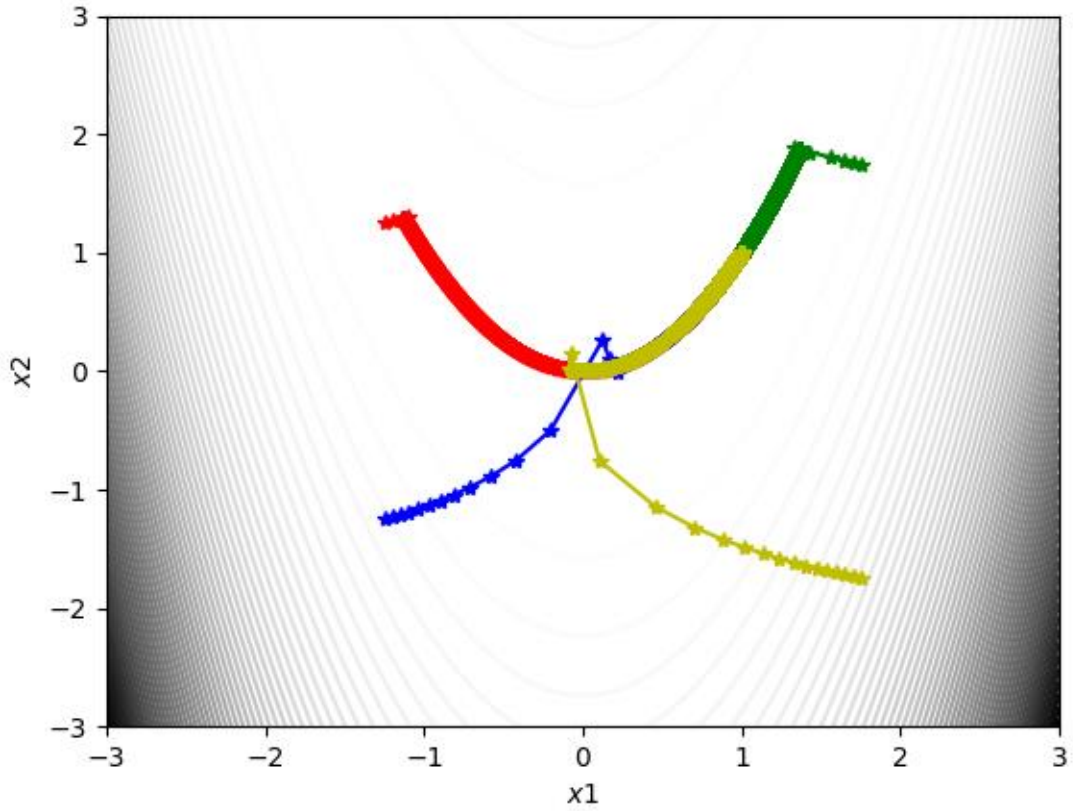
The above optimization took 11141 iterations.

When the Rosenbrock function is minimized using steepest descent with backtracking algorithm with $\alpha_{init} = 0.05$ and $\rho = 0.001$, the following path is taken by the optimizer and the optimization took 12343 iterations.



Optimizer's path while finding minimum of rosenbrock function using backtracking

Using different starting points and directions for finding the minimum using **backtracing line search** using steepest descent gives the following figure:



The $\alpha_{init} = 0.05$ and $\rho = 0.7$

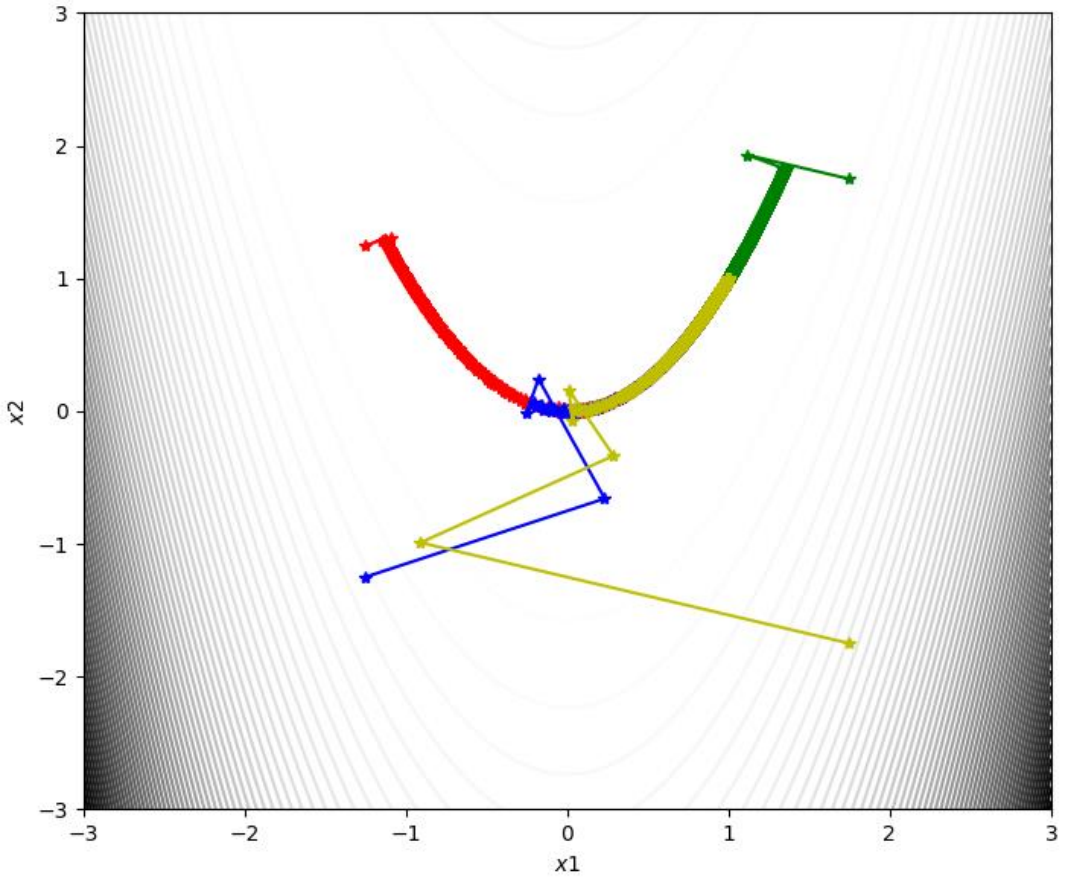
Initial Starting point	Optimum Value	Color on the graph	Iterations
-1.25, 1.25	1.2510795181731303e-10	Red	14113
-1.25, -1.25	1.2507641395243425e-10	Blue	12274
1.75, 1.75	1.248814949471067e-10	Green	17453
1.75, -1.75	1.2516779161198528e-10	Yellow	12343

On tuning ρ for a particular case ($\alpha_{init} = 0.05 \; and \; x_0 = [-1.25, 1.25]$):

ρ	Iterations
0.001	Didn't converge even with 100,000 iterations
0.5	13628
0.7	14113
1.3	18092

For $\rho = 1.4$ and more we observe nans during optimization.

Using different starting points and directions for finding the minimum using **Bracketing and pinpointing line search** using steepest descent gives the following figure:



The $\alpha_{init} = 0.05 \; and \; \mu_2 = 0.001$

Initial Starting point	Optimum Value	Color on the graph	Iterations
-1.25, 1.25	8.238321845376736e-11	Red	11431
-1.25, -1.25	8.82194036512735e-11	Blue	11161
1.75, 1.75	9.072595594275349e-11	Green	13577
1.75, -1.75	8.795482394951342e-11	Yellow	11141

On tuning μ_2 for a particular case ($\alpha_{init} = 0.05 \; and \; x_0 = [-1.25, 1.25]$):

μ_2	Iterations
0.001	11431

0.5	Didn't converge
0.7	760
0.9	2375

Now we try RosenBrock in **n-Dimension**, here we are keeping n=6.

The function can be written as follows:

$$f(\boldsymbol{x}) = f(x_1, x_2, x_3, \dots, x_n) = \sum_{k=0}^{n} \left(100\left(\mathrm{x}_{k+1} - \mathrm{x}_{\mathrm{k}}^2\right)^2 + (1 - x_k)^2\right)$$

Where $\boldsymbol{x} = (x_1, x_2, \dots., x_N) \in \mathbb{R}^{\boldsymbol{N}}$

There are two minimums that exist for N∈ [4,7], they are (1,1,…1,1) and (-1,1,1,…1).

For this function gradient can be written as follows:

If i∈**[1, n-1]**

$$\frac{\partial f}{\partial x_i} = -400x_i\left(x_{i+1} - x_i^2\right) - 2(1 - x_i)$$

Else if i= n

$$\frac{\partial f}{\partial x_i} = 200(x_i - x_{i-1}^2)$$

Results for RosenBrock using **bracketing and pinpoinitng** for n=6 is for $\alpha_{init} = 0.05, \mu_2 = 0.9$ shown in figure below:

```
the optimal is [0.99999933 0.99999867 0.99999733 0.99999464 0.99998926 0.99997847] and minimum is 1.5373280460513578e-10
```

Other values such as $\mu_2 = 0.7$ with $\mu_1 = 10^{-4}$ were tried but the line search didn't converge even after many iterations

Results for Rosen Brock using **Backtracking** with parameters $\rho = 0.7,\ \alpha_{init} = 1.5, \mu_1 = 10^{-4}$ are shown below:

```
the optimal is [1.00000067 1.00000135 1.0000027  1.0000054  1.00001083 1.0000217 ] and minimum is 1.5617536143567668e-10
```

Other values such as $\rho = 0.5, 0.9$ with 0.05 don't converge and $\rho = 0.9$ with $\alpha_{init} = 1.5$ takes many iterations but doesn't converge.