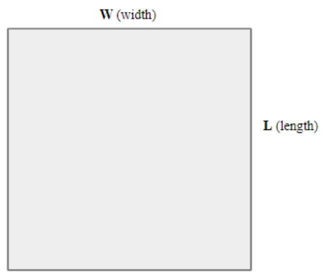
AE588 HW4

Q1:



Problem Formulation:

$$\text{Minimize} \quad -WL$$
$$\text{By varying} \quad W, L$$
$$\text{Subject to} \quad 2(W+H) - P = 0$$

**Negative sign is introduced in objective function as the function needs to be maximized*
**P denotes the perimeter of the rectangle*

The Lagrangian equation = $\mathcal{L}(W, L) = -WL + \lambda(2(W+H) - P)$

Differentiating the Lagrangian equation w.r.t. design variable and Lagrangian multiplier to get the first-order optimality conditions:

$$\frac{\partial \mathcal{L}}{\partial W} = -L + 2\lambda = 0$$
$$\frac{\partial \mathcal{L}}{\partial L} = -W + 2\lambda = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 2W + 2L - P = 0$$

Solving the system of equations, we get:

$$\lambda = P/8, \quad W = P/4, \quad L = P/4$$

Therefore, width (W) is equal to length (L), which is square, is the optimum point for a rectangle of a given perimeter, to have the largest area.

Thus, validating the Euclid's statement that among rectangles of a given perimeter, the square has the largest area.

Unit: Lagraginan paramter tells us the rate of change in area with respect change in dimension of length or width. Thus the units of the lagragian parameter is $m^2/m = m$

Minimizing the perimeter with an area constrained to the optimum value, found before, i.e, $A = (P/4)^2 = P^2/16$.

Problem Formulation:

$$\text{Minimize} \qquad 2(W+L)$$

$$\text{By varying} \qquad W, L$$

$$\text{Subject to} \qquad WL - A = 0$$

The Lagrangian equation = $\mathcal{L}(W,L) = 2(W+L) + \lambda(WL - A)$

Differentiating the Lagrangian equation w.r.t. design variable and Lagrangian multiplier to get the first-order optimality conditions:

$$\frac{\partial \mathcal{L}}{\partial W} = 2 + \lambda L = 0$$
$$\frac{\partial \mathcal{L}}{\partial L} = 2 + \lambda W = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda} = WL - A = 0$$

Solving the system of equations, we get:

$$\lambda = -\sqrt{\frac{4}{A}}, \quad W = \sqrt{A} = \frac{P}{4}, \quad L = \sqrt{A} = \frac{P}{4}$$

**Ignoring negative length and width (Lagrangian multiplier could be negative for equality constraint thus it is a minimum.*

Thus, obtaining the same solution as previous problem formulation, where width (W) is equal to length (L), which is square.

Q2.

Problem Formulation:

$$\text{Minimize} \qquad 2bt_b + ht_w$$

$$\text{By varying} \qquad t_b, t_w$$

$$\text{Subject to} \qquad g_1 \quad \frac{Plh}{2I} - \sigma_{yield} \leq 0$$

$$g_2 \quad \frac{1.5P}{ht_w} - \tau_{yield} \leq 0$$

Were,

$$I = \frac{h^3}{12}t_w + \frac{b}{6}{t_b}^3 + \frac{h^2 b}{2}t_b$$

The Lagrangian equation is:

$$\mathcal{L}(t_b,\ t_w, \sigma_1, \sigma_2, s_1, s_2) = 2bt_b + ht_w + \sigma_1\left(\frac{Plh}{2I} - \sigma_{yield} + {s_1}^2\right) + \sigma_2\left(\frac{1.5P}{ht_w} - \tau_{yield} + {s_2}^2\right)$$

Differentiating w.r.t to design variables, equality & inequality Lagrangian multipliers and complementary slackness variables to get the first order optimality conditions:

$$\frac{\partial \mathcal{L}}{\partial t_b} = Phl\sigma_1 \frac{-bh^2 - b{t_b}^2}{bh^2 t_b + \frac{b{t_b}^3}{3} + \frac{h^3 t_w}{6}} + 2b = 0$$

$$\frac{\partial \mathcal{L}}{\partial t_w} = \frac{-Ph^4 l\sigma_1}{6\left(bh^2 t_b + \frac{b{t_b}^3}{3} + \frac{h^3 t_w}{6}\right)^2} + \frac{1.5P\sigma_2}{h{t_w}^2} + h = 0$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_1} = \frac{Plh}{2\left(\frac{h^3}{12} t_w + \frac{b}{6} {t_b}^3 + \frac{h^2 b}{2} t_b\right)} - \sigma_{yield} + {s_1}^2 = 0$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_1} = \frac{1.5P}{ht_w} - \tau_{yield} + {s_2}^2 = 0$$

$$\frac{\partial \mathcal{L}}{\partial s_1} = 2\sigma_1 s_1 = 0$$

$$\frac{\partial \mathcal{L}}{\partial s_2} = 2\sigma_2 s_2 = 0$$

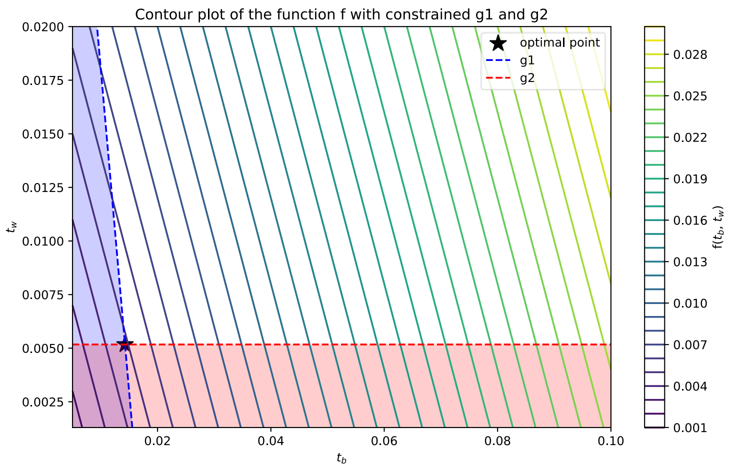
The two complementary slackness conditions yield four potential combinations as shown in the table below:

Assumption	Meaning	t_b	t_w	σ_1	σ_2	s_1	s_2	Point
$s_1 = 0$ $s_2 = 0$	g₁ active g₂ active	0.0142604	0.00517241	1.9e-11	7.4e-12	**0**	**0**	x^*
$\sigma_1 = 0$ $\sigma_2 = 0$	g₁ inactive g₂ inactive	-	-	**0**	**0**	-	-	-
$s_1 = 0$ $\sigma_2 = 0$	g₁ active g₂ inactive	-	-	-	**0**	**0**	-	-
$s_2 = 0$ $\sigma_1 = 0$	g₁ inactive g₂ active	-	-	**0**	-	-	**0**	-

Thus, by solving the system of equation for two active constraints, the optimal point is, when both constraints were active:

$$\boldsymbol{t_b = 0.0142604}\textbf{m}$$
$$\boldsymbol{t_w = 0.0051724}\textbf{m}$$

**Only solution with real value were considered and imaginary values were disregarded.*



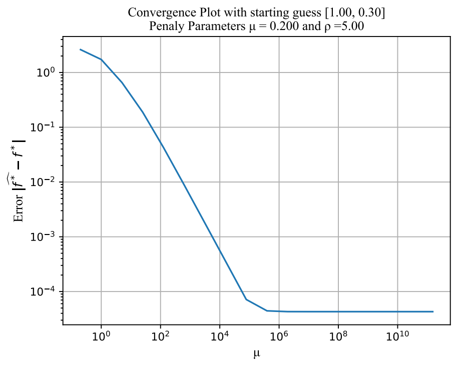
4.3

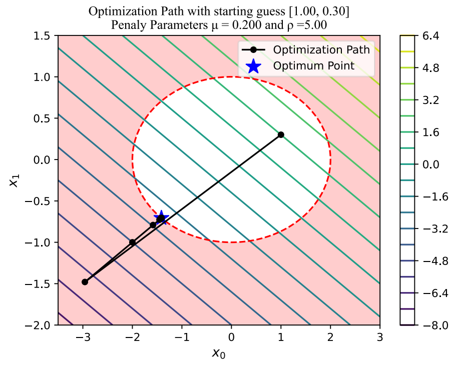
Implemented Quadratic Penalty Method for Exterior Penalty method and Logarithmic Penalty Method for Interior Penalty method in conjunction with BFGS method previously implemented in HW3.

<u>For Exterior Penalty Method:</u>

The Optimal Point obtained:

$$\boldsymbol{x_1 = -1.41425196, x_2 = -0.70708758}$$

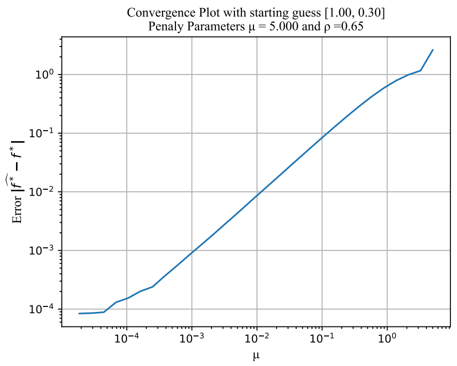


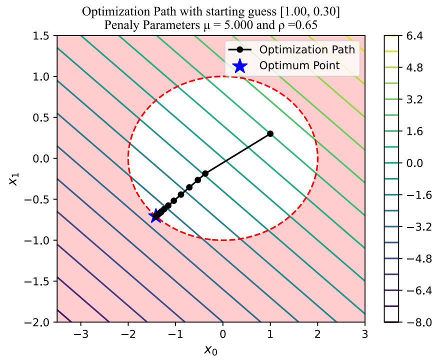


<u>For Interior Penalty Method:</u>

The Optimal Point obtained:

$$\boldsymbol{x_1 = -1.41413035, x_2 = -0.70712617}$$



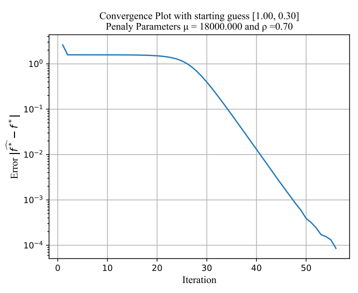


<u>Exterior Penalty Method – Penalty Parameters:</u>

The penalty parameter varied from $0.001 \leq \mu \leq 9000$ while keeping the $\rho = 5$, after which the penalty method failed to converge to the optimum. It is to be noted as we increase the penalty parameter the number of iterations it took to converge to the solution decreased all the way down to 3 iterations.

<u>Interior Penalty Method – Penalty Parameters:</u>

The penalty parameter varied from $0.01 \leq \mu \leq 18000$ while keeping the $\rho = 0.7$, after which the penalty method failed to converge to the optimum. However, unlike exterior method, interior method took a greater number of iterations to converge to the solution as we increase the penalty.



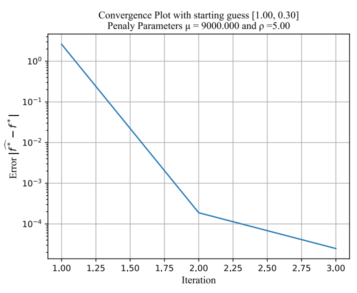


Figure 1: a) Interior Penalty Method b) Exterior Penalty Method

Invariant of penalty parameter or starting point, both methods achieved a max convergence of 1e-4. It is to be noted most often exterior penalty method was able to achieve an optimum closer to actual minimum point more often than interior method. Thus, it could be concluded that **both Exterior and Interior Penalty Method could achieve an accuracy up to 4th decimal point or 1e-4 error.**

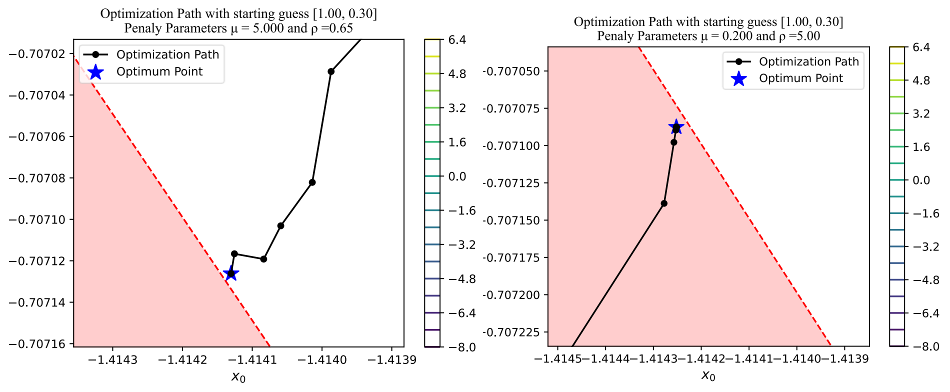
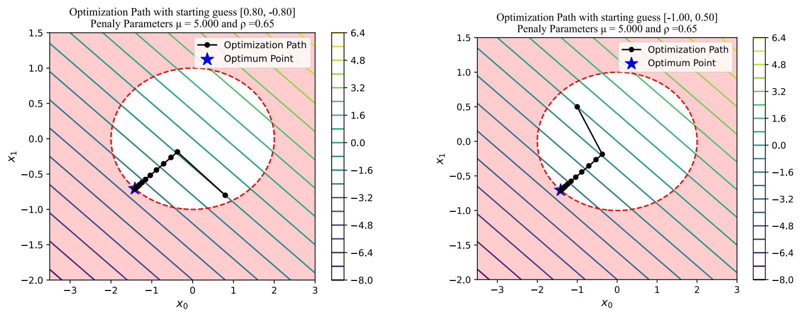


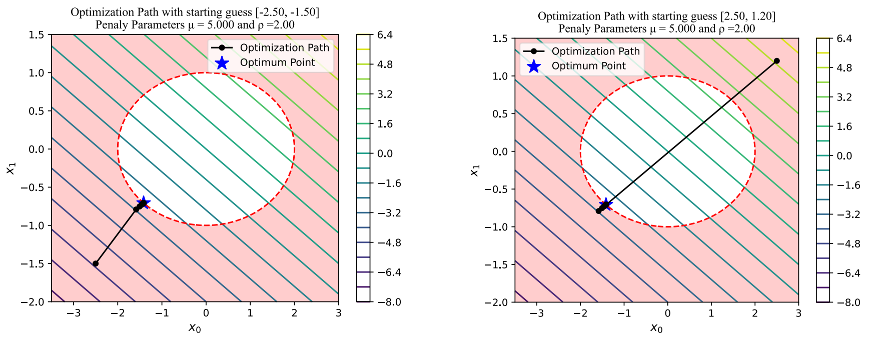
Figure 2: a) Interior Penalty Method b) Exterior Penalty Method

It is evident from the image that exterior penalty method converges to a point that lies in the infeasible area, however, the interior penalty method is within the feasible area. But interior point method still hasn't converged to the absolute boundary of the constraint where the true optimum lies, thus explaining why it was only able to achieve a converge error of 1e-4. Thus, it can be concluded **that interior method optimum is strictly feasible with respect to constraint, however, external penalty method optimum is not strictly feasible.**

<u>Different Starting points for interior penalty method:</u>



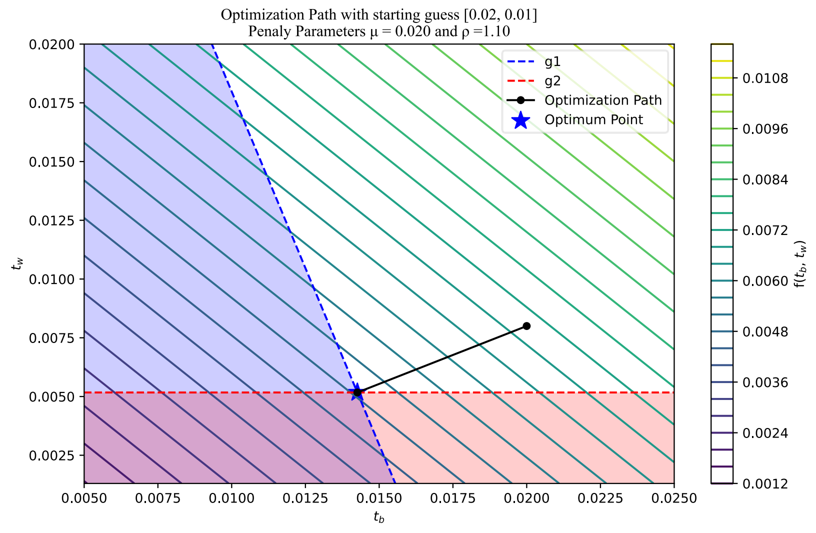
<u>Different Starting points for exterior penalty method:</u>



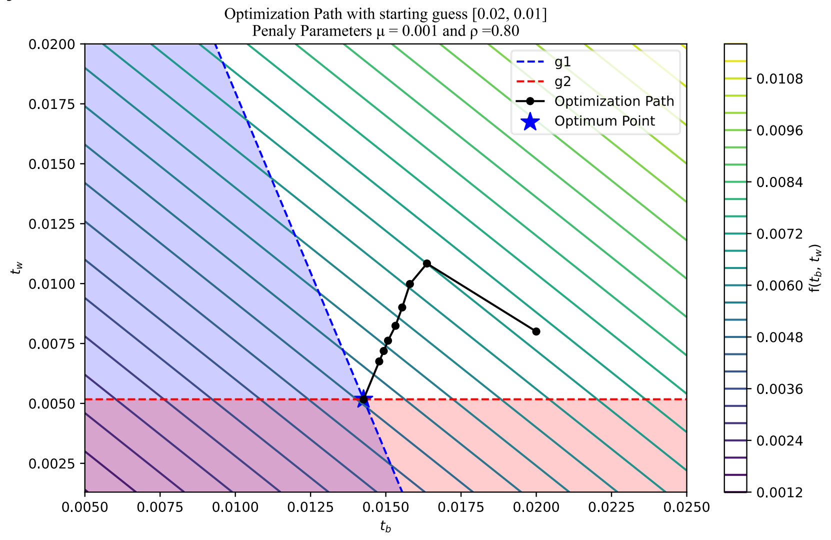
Both methods converge to the same optimum with 4 decimal places irrespective of starting point. Starting point could be in the unfeasible region for exterior penalty method, however, for interior penalty method the starting must be within the feasible region for it to converge.

Cantilever beam optimization:

<u>For Exterior Penalty Method:</u>



<u>For Interior Penalty Method:</u>



For exterior penalty method, the first iteration itself it achieved an error of 1e-12, thus achieving the exact analytical solution. This is maybe due to a good line search method implemented. This was independent of penalty parameters, it always took only 1 iteration, however, the computation time spend in BFGS increases significantly for higher penalty parameter.

Similarly, for interior method it took more iteration however, it also converges to1e-12 error. However, it was extremely sensitive to ρ and μ (penalty parameters), different values lead to a convergence of 1e-4, or 1e-6.

Method	$\boldsymbol{t_b}$	$\boldsymbol{t_w}$
Exterior Method	0.0142604	0.00517241
Interior Method	0.0142604	0.00517241

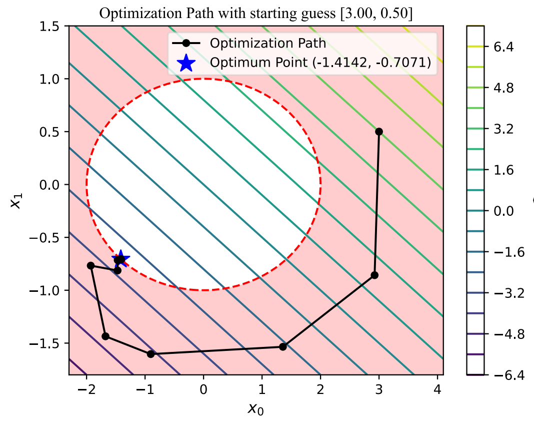
4.4

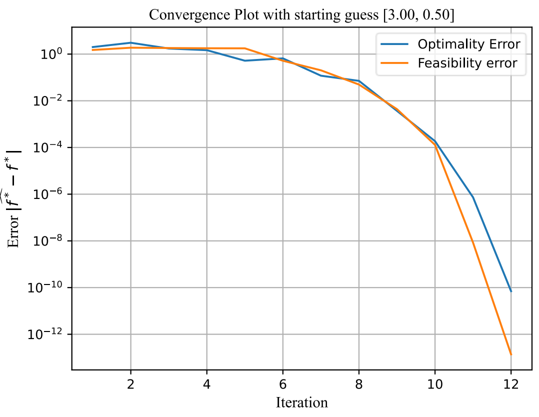
Implementing Quasi-SQP with convergence criteria $\tau_{optimality} \leq 10^{-9}$ *and* $\tau_{feasibility} \leq 10^{-9}$:

Example Function:

$$\text{Minimize} \qquad x_1 + 2x_2$$

$$\text{Subject to } \frac{1}{4}{x_1}^2 + {x_2}^2 - 1 = 0$$

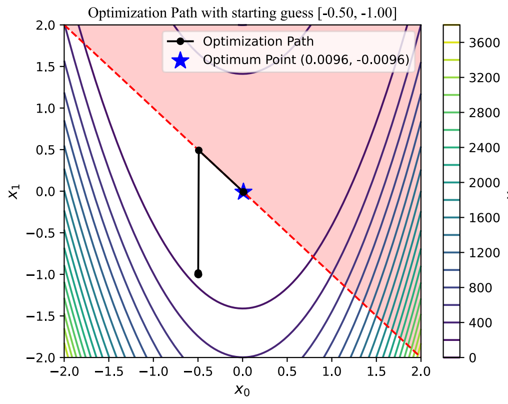


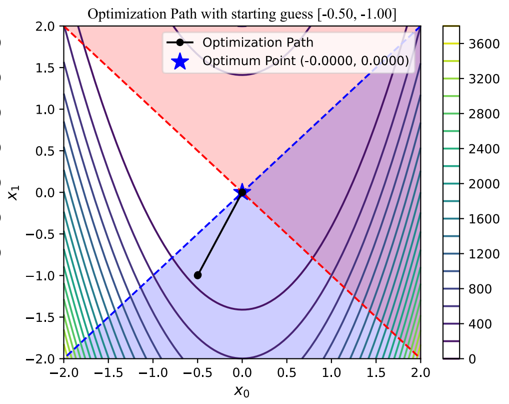


2-D Rosenbrock Function (*image below on left*):

$$\text{Minimize} \qquad \sum_{i=1}^{d-1} 100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2, where\ d\ =\ 2$$

$$\text{Subject to } x_1 + x_2 = 0$$





2-D Rosenbrock Function multiple constraint (*image above on right*):

$$\text{Minimize} \qquad \sum_{i=1}^{d-1} 100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2, where\ d\ =\ 2$$

$$\text{Subject to:} \quad x_1 + x_2 = 0$$
$$x_1 - x_2 = 0$$

4-D Rosenbrock Function:

$$\text{Minimize} \qquad \sum_{i=1}^{d-1} 100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2, where\ d\ =\ 4$$

$$\text{Subject to } x_1 + x_2 + x_3 = 0$$

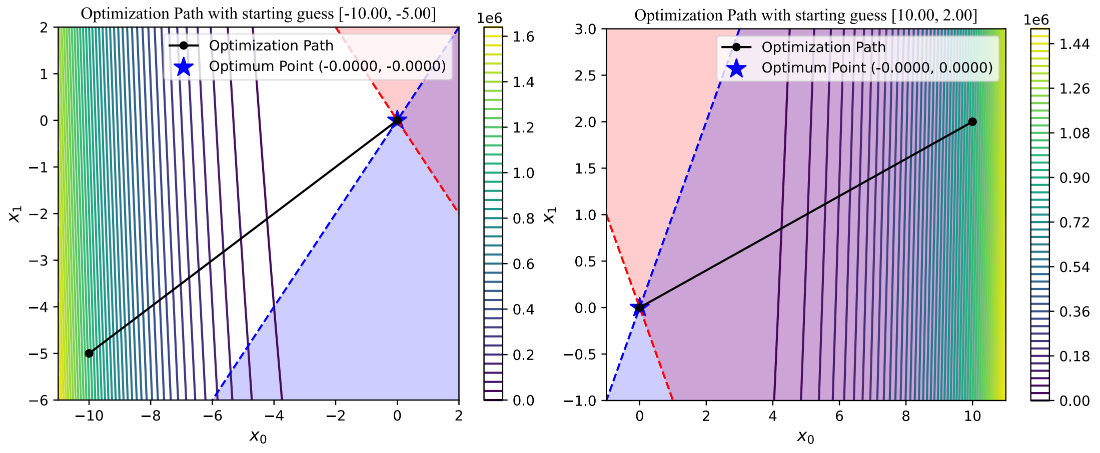
Initial Guess $x_0 = [1, 1, 1, 1]$

Minimum $\boldsymbol{x}^* = [7.3 \times 10^{-14}, -5.4 \times 10^{-14}, -1.8 \times 10^{-14}, 1.3 \times 10^{-15}] \cong [0, 0, 0, 0]$

$\tau_{optimality} = 5.8371 \times 10^{-12}$ *and* $\tau_{feasibility} = 1.357 \times 10^{-12}$

Iterations = 20

For 2-D Rosenbrock Function multiple constraint, starting with different point:



It is evident from above, that the quasi SQP method performs well with multiple constraint and increase dimensionality of the functions. However, if bracketing and pinpointing is used, the convergence depends significantly on starting point, and the computational time increases to hours, but it always solves in one iteration. Thus, backtracking was chosen for rest of problem formulation.

Comparing Quasi SQP methods to SciPy's minimize:

```
minimize(scipy_func, x0, constraints=con, method='SLSQP', jac=scipy_grad, options=
   {'disp': True, 'tol': 1e-9})
```

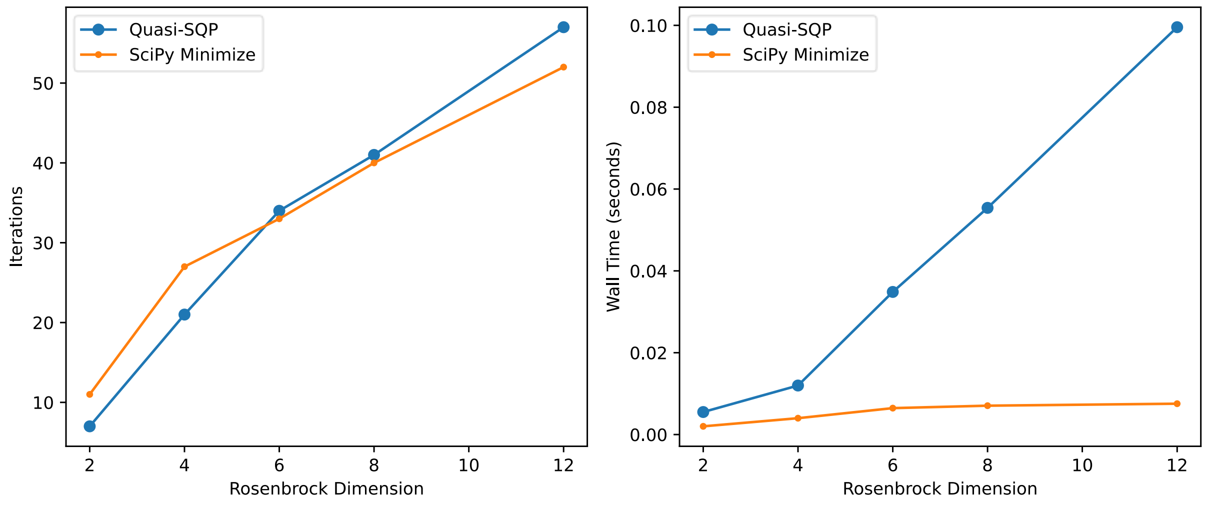
1) Varying the dimensions:

$$\text{Minimize} \qquad \sum_{i=1}^{d-1} 100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2, \textit{where d is dimension}$$

$$\text{Subject to} \qquad \sum_{i=1}^{d-1} x_i = 0$$

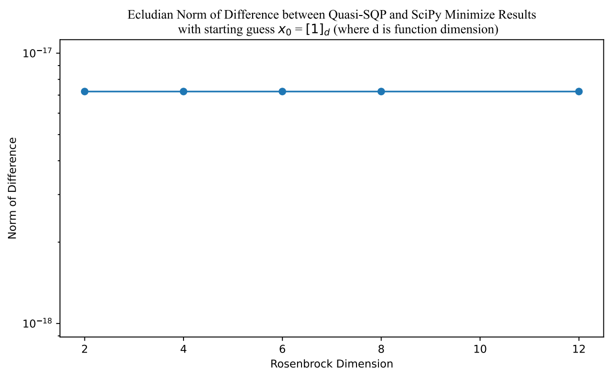
<u>Computational Cost:</u>

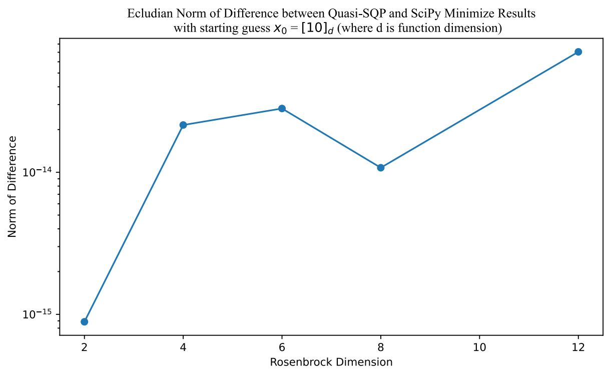
Rosenbrock Function with a starting guess $x_0 = [1]_d$ (where d is function dimension) Subjected to: h(x) = $\sum_{i=1}^{n_x - 1} x_i = 0$



It is evident that the number of iterations taken were almost the same for both methods, which implies that our method was correctly implemented and resetting the hessian parameter was tuned correctly. The SciPy took fewer iteration for higher dimension function. However, looking at wall time, even though it took same iteration Quasi-SQP method spend more time per iteration, and the wall time was significantly higher for higher dimensional Rosenbrock function. It could be attributed to higher function call in the Quasi-SQP method.

Accuracy:





The Euclidean norm between each method was about 1e-16 irrespective of dimension, when starting close to the optimal point. However, when starting point was farther away SciPy performed a bit better for higher dimensional Rosenbrock function, however, difference was negligible .

Robustness:
Both methods converge to same optimal point when starting from different guess, thus we can conclude that both methods are robust for higher dimensional cases. But we still need to investigate if they are robust for higher dimensional function and higher number of constraints.

2) Varying number of constraints:
Objective function was 8-D Rosenbrock Function:

Minimize $\sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, where\ d\ = 8$

1 constraint:

Subject to $\sum_{i=1}^{7} x_i = 0$

2 constraints:

Subject to
$$x_1 + x_2 = 0$$
$$x_1 + x_2 + x_3 = 0$$

3 constraints:

Subject to
$$x_1 + x_2 = 0$$
$$x_2 + x_3 = 0$$
$$x_4 + 2x_5 = 0$$

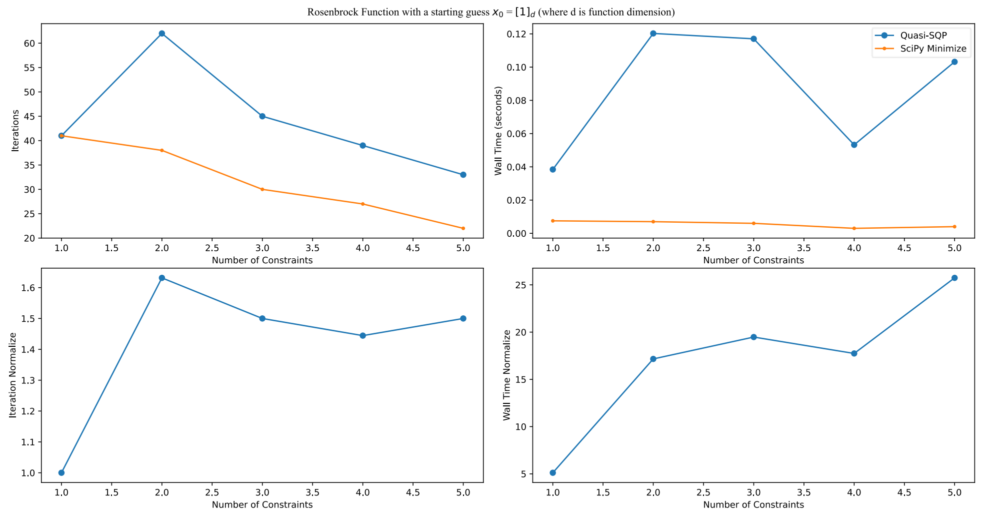
4 constraints:

Subject to
$$x_1 + x_2 = 0$$
$$x_2 + x_3 = 0$$
$$x_4 + x_5 = 0$$
$$x_7 + x_8 = 0$$

5 constraints:

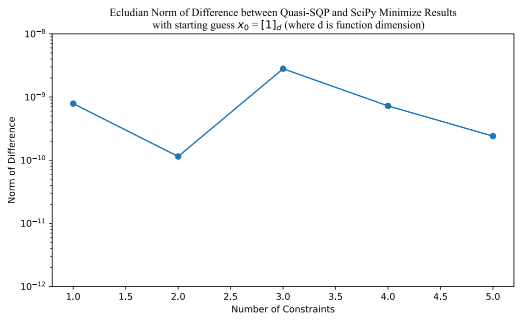
Subject to
$$x_1 + x_2 = 0$$
$$x_2 - x_3 = 0$$
$$x_4 + 2x_5 = 0$$
$$x_7 + x_8 = 0$$
$$x_1 + 5x_5 + x_6 = 0$$

<u>Computational Cost:</u>



Surprisingly number of iterations went down as number of constraints increases (ignoring 2 constraints run). My method took on average 1.5 times total iteration to converge when compared to SciPy. Moreover, from the plot, it is evident that my function took longer to run. On average it was 20 times slower when compared to SciPy wall time. However, these results were highly sensitive to constraint formulation talked more in detail in later section.

<u>Accuracy:</u>



The accuracy of my function was comparable to SciPy optimal value. On average the difference or norm was within 1e-9, irrespective of number of constraints. Similar results were observed as stated before when running from farther away starting point, with norm decreasing to 1e-8.

<u>Robustness:</u>

Both methods converge to same optimal point when starting from different guess. However, the constraint formulation had a huge impact on performance. For some constraint, my function was unable to solve the quadratic problem because the matrix became singular, because the step it took was to the same point making the results zeros and producing Nan results. However, if we examine the point from last iteration to SciPy converged results, the point was less than 1e-7 different (Euclidean norm). Thus, by adding some preventive measures in my function for Quasi-SQP, it became robust enough to converge to the minimum point, but the accuracy decreased to 1e-7 or 1e-6. Thus, my function for multiple constraint is not robust enough without safeguards.