# Metaheuristic Optimization

<u>Exercise 1</u>

Revenue Maximisation Problem using the Simplex Method

A.

Formulation as an optimization problem:

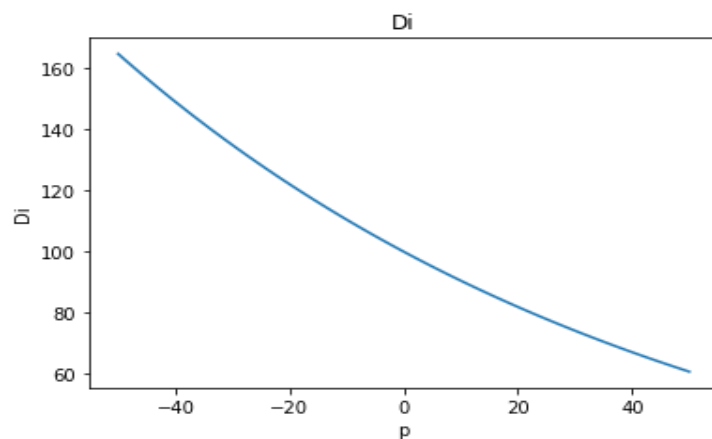Min [ - ((a1×exp(-p1/a1)) + (a2×exp(-p2/a2)) +(a3×exp(-p3/a3))]

Under the following constraints:

D1= a1×exp(-p1/a1)

D2= a2×exp(-p2/a2)

D3= a3×exp(-p3/a3)



B.

See python file

C.

See python file

<u>Exercise 2</u>

- The Banana (Rosenbrock) Function

In minimizing Rosenbrock function, the gradient descent approach (implemented in python) was used within bounds [-4, 4]. In all cases, the global minimum was found within 6 significant digits.

This a feasible solution as we can observed on the python file, for each run the algorithm is converging toward the global optimum

| Run | Starting point | Optimal point | Optimal value | Number of iterations | Feasible | Time (sec) |
|---|---|---|---|---|---|---|
| 1 | [ 1.66250252 -0.90 712071] | [1.000000 21 1.0000 0042] | 0 | 34 | Yes | 0.0260787010192871 1 |
| 2 | [ 1.34471794 -2.61 502665] | [0.99999996 0.999 9992] | 0 | 30 | Yes | 0.0258681774139404 3 |
| 3 | [ 1.49774056 -3.51 780251] | [0.99999995 0.999 9999 ] | 0 | 26 | Yes | 0.0208878517150878 9 |
| 4 | [-3.15569399 -0.4 1769905] | [0.99999998 0.999 99996] | 0 | 57 | Yes | 0.0292642116546630 86 |
| 5 | [-3.52320854 -0.9 3248649] | [1. 1.] | 0 | 74 | Yes | 0.0476920604705810 55 |
| 6 | [1.22956255 2.465 09778] | [0.99999997 0.999 99993] | 0 | 20 | Yes | 0.0260801315307617 2 |
| 7 | [ 2.36148569 -2.64 056721] | [1.00000002 1.000 00003] | 0 | 41 | Yes | 0.0258457660675048 83 |
| 8 | [-1.18806422  1.04 054283] | [1. 1.] | 0 | 35 | Yes | 0.0234503746032714 84 |
| 9 | [-1.66251644 -2.8 2015772] | [0.99999999 0.999 99998] | 0 | 55 | Yes | 0.0456044673919677 7 |
| 10 | [-3.72101968 -2.9 4140858] | [1. 1.] | 0 | 72 | Yes | 0.0517435073852539 06 |

Rosenbrock Optimization with BFGS

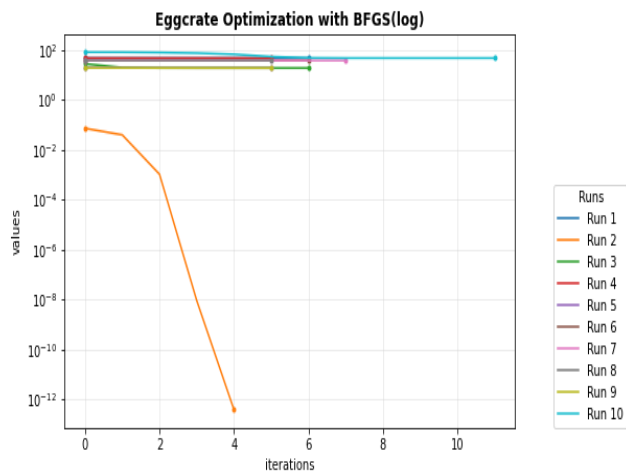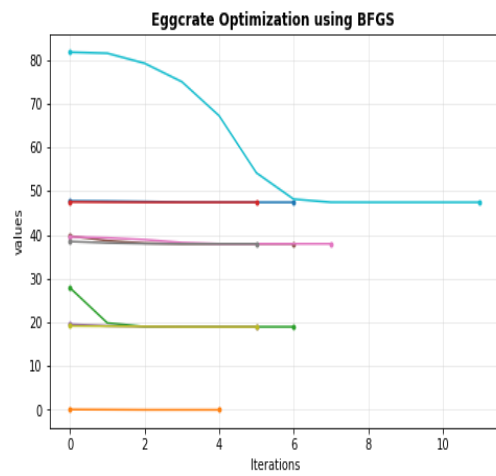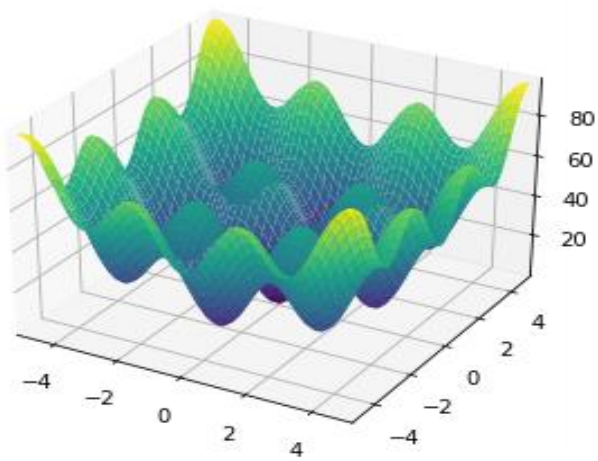Rosenbrook Optimization with BFGS(log)
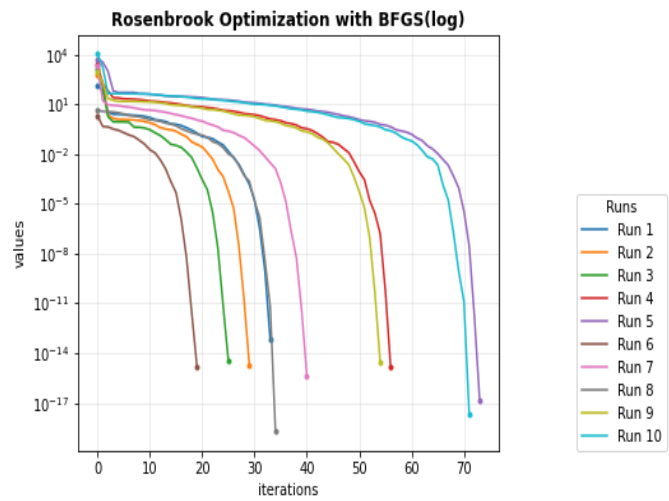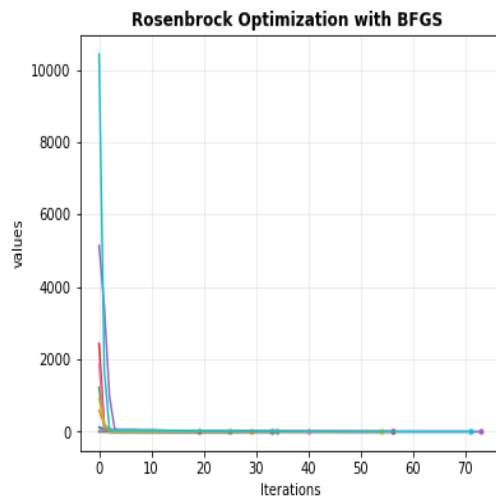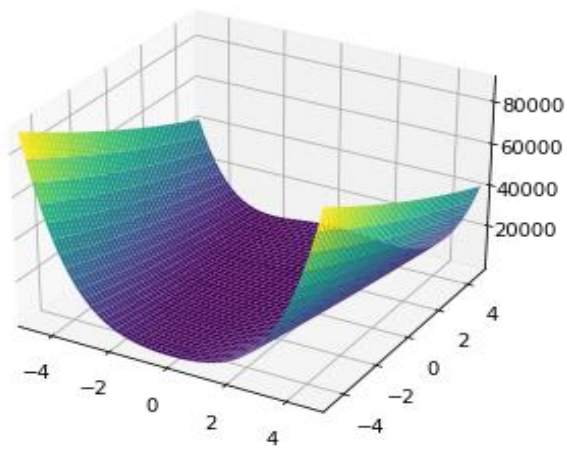
See Python file

- The eggcrate function

For eggcrate function, the same algorithm was employed and it was noticed that gradient- based optimizer were stuck in the local optima, depending on the starting point that were chosen randomly within bounds [-2π, 2π].

Out of 10 runs, 9 was stuck in a local minimum while 1 found the global optima.

In this case we still have feasible solutions. Although the optimum at each run is feasible, it is still unlikely to catch the global optimum.

| Run | Starting point | Optimal point | Optimal value | Number of iterations | Feasible | Time (sec) |
|-----|----------------|---------------|---------------|---------------------|----------|------------|
| 1 | [-4.13925234 -5.9 932668 ] | [-3.01960188 -6. 03142402] | 47.417669 | 7 | Yes | 0.00765991210 9375 |
| 2 | [ 0.64868558 -0.72 541048] | [-3.92466624e-0 8 -1.14949864e- 07] | 0 | 5 | Yes | 0.00628852844 23828125 |
| 3 | [4.39006606 2.425 64939] | [3.01960187 3.0 1960187] | 18.976395 | 7 | Yes | 0.00611758232 1166992 |
| 4 | [ 5.58882228 -2.95 615944] | [ 6.03142402 -3. 01960194] | 47.417669 | 6 | Yes | 0.01380848884 5825195 |
| 5 | [-4.05061976 -2.5 695851 ] | [-3.0196019 -3. 0196087] | 18.976395 | 6 | Yes | 0.00565719604 4921875 |
| 6 | [0.6891563 5.750 93381] | [8.02955849e-0 9 6.03142401e+ 00] | 37.929472 | 7 | Yes | 0.01337194442 7490234 |
| 7 | [-5.56695809 -0.6 3792794] | [-6.03142401e+ 00 3.85023245e -10] | 37.929472 | 8 | Yes | 0.01063036918 6401367 |
| 8 | [ 4.53745943 -5.02 133461] | [-3.57050750e-1 0 -6.03142401 + 00] | 37.929472 | 6 | Yes | 0.00766730308 53271484 |
| 9 | [2.68352678 3.890 65855] | [3.01960195 3.0 1960187] | 18.976395 | 6 | Yes | 0.00612545013 4277344 |
| 10 | [ 4.86697494 -5.98 830213] | [ 3.01960189 -6. 03142401] | 47.417669 | 12 | Yes | 0.01013779640 1977539 |

Eggcrate Optimization using BFGS

Eggcrate Optimization with BFGS(log)

Runs
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Run 7
Run 8
Run 9
Run 10

See python file

- Golinski's Speed Reducer

SLSQP – Sequential Least Squares Programming approach has been used for this optimization problem

Finally the Golinski's speed reducer problem was solved using sequential quadratic programming approach (implemented in Python) that handles gradient-based constrained optimization problems. This problem has 11 constraints, in addition to bound constraints and objective is to minimize the weight of the speed reducer. The 10 starting points were picked at random as before and the quickly optimizer converged to the optimum in all cases.

| Run | Starting point | Optimal point | Optimal value | Number of iterations | Feasible | Time (sec) |
|---|---|---|---|---|---|---|
| 1 | [2.7551, 0.7436, 21.1665, 7.3794, 8.0887, 2.9018, 5.2142] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 11 | Yes | 0.207408189773559 57 |
| 2 | [3.3695, 0.8088, 25.0352, 7.9858, 8.0607, 3.1946, 5.4470] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 8 | Yes | 0.094806671142578 12 |
| 3 | [3.4303, 0.7682, 19.6928, 7.3937, 7.3078, 3.3007, 5.5022] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 5 | Yes | 0.066616296768188 48 |
| 4 | [2.8209, 0.7467, 22.2140, 7.9013, 7.7479, 3.8998, 5.5598] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 10 | Yes | 0.205561399459838 87 |
| 5 | [3.0868, 0.7663, 19.6152, 7.7766, 8.0903, 3.8417, 5.8043] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 9 | Yes | 0.088186979293823 24 |
| 6 | [2.8003, 0.7415, 20.2288, 7.8916, 8.1696, 3.7621, 5.5744] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 8 | Yes | 0.059932708740234 375 |
| 7 | [3.5311, 0.7799, 22.3920, 7.9197, 7.4234, 3.8438, 5.2505] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 5 | Yes | 0.072154521942138 67 |
| 8 | [3.0801, 0.7039, 17.4099, 7.9046, 8.1320, 3.7719, 5.1876] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 10 | Yes | 0.125942468643188 48 |

| 9 | [3.4462, 0.7669, 24.0182, 8.1382, 7.8775, 2.9449, 5.4831] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 12 | Yes | 0.12774682044982921 |
|---|---|---|---|---|---|---|
| 10 | [2.6080, 0.7266, 25.8264, 7.9898, 7.8160, 3.7639, 5.4972] | [3.5000, 0.7000, 17.0000, 7.3000, 7.7153, 3.3502, 5.2867] | 2994.3516 | 11 | Yes | 0.13335514068603516 |
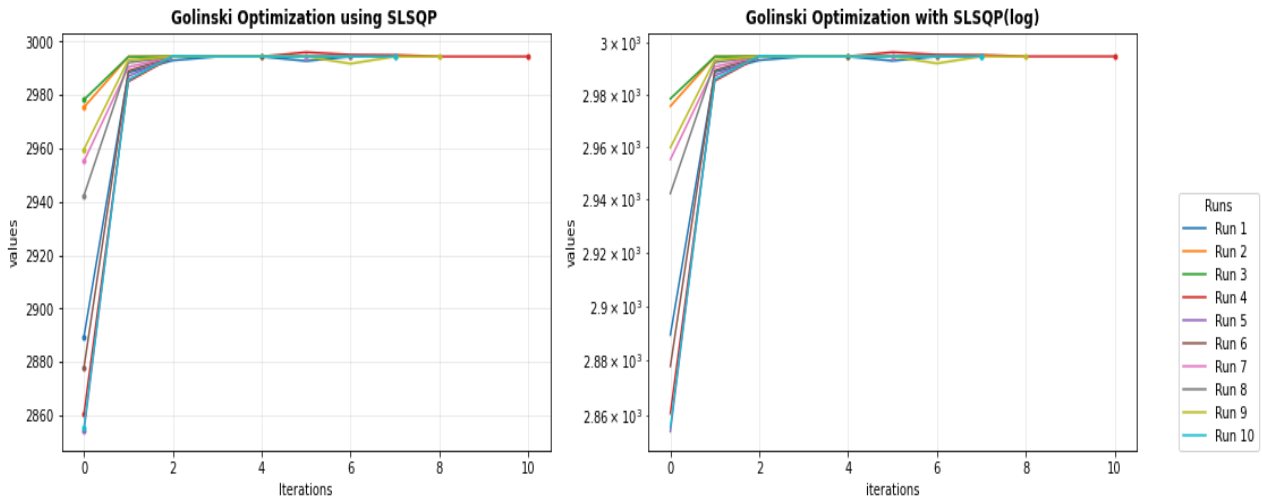


See python file

## Exercise 3

As heuristic technique, we have used the PSO - Particle Swarm Optimization approach (See Python file).

i.      Dependence of answers on initial design vector (start point, initial population)

| Problem Name | Gradient-Based Optimizer | Particle Swarm Otpimizer |
|---|---|---|
| Rosenbrock Function | low | Low |
| Eggcrate Function | High | Low |
| Golinski Speed Reducer | Low | low |

ii.     Computational effort (CPU time [sec] or FLOPS)

| Problem Name | Gradient-Based Optimizer | Particle Swarm Otpimizer |
|---|---|---|
| Rosenbrock Function | 0.031 | 0.347 |
| Eggcrate Function | 0.008 | 0.151 |
| Golinski Speed Reducer | 0.117 | 17.43 |

iii.    Convergence history

| Problem Name | Gradient-Based Optimizer | Particle Swarm Otpimizer |
|---|---|---|
| Rosenbrock Function | Always converged to global minimum. | Converged, but efficiency depends on the tuning parameters selected |
| Eggcrate Function | Always converged, but either a local or a global minimum. | Converged, but efficiency depends on the tuning parameters selected |
| Golinski Speed Reducer | Always converged to global minimum. | Converged, but efficiency depends on the tuning parameters selected |

iv.     Frequency at which the technique gets trapped in a local optimum

| Problem Name | Gradient-Based Optimizer | Particle Swarm Otpimizer |
|---|---|---|
| Rosenbrock Function | 0 | 0 |
| Eggcrate Function | 0.9 | 0 |
| Golinski Speed Reducer | 0 | 0 |

Conclusion

Gradient-based optimizers can get stuck in local optima and are sensitive to the starting point, especially if there are multiple optima in the design space.

Genetic Algorithms are computationally expensive and requires considerable "tuning" effort, especially for complex problems.