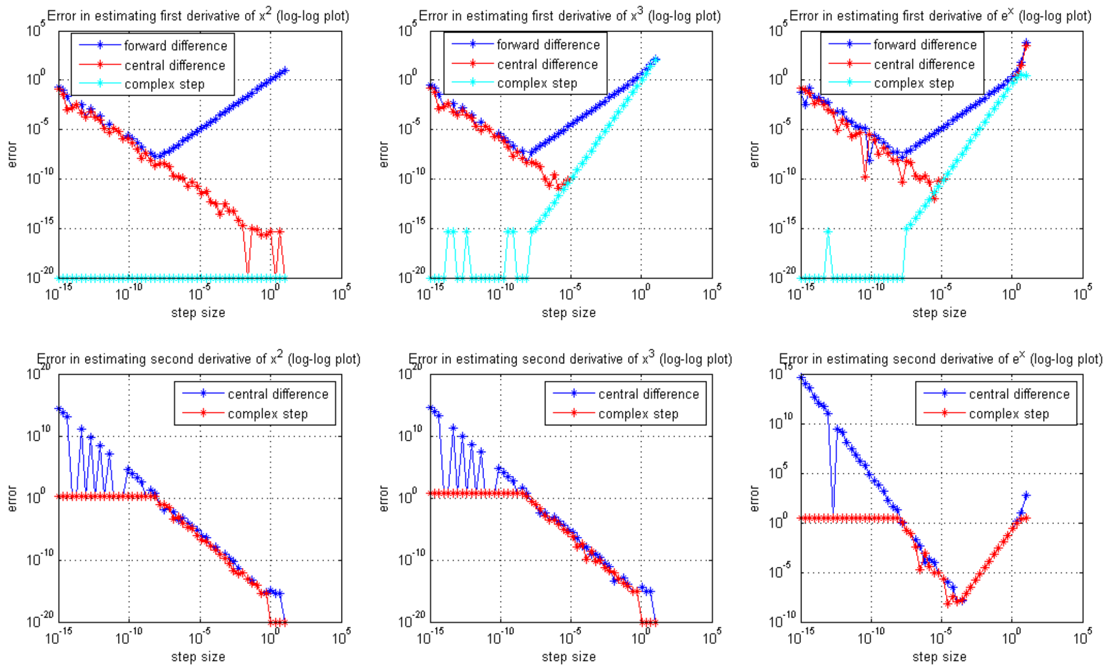
16.888/ESD 77 Multidisciplinary System Design Optimization: Assignment 3 Part a) Solution

Part A1

The absolute error between first and second derivative estimates vis-à-vis their analytical value at x =1 by suggested methods is shown in the figure below using log-log plots.



For estimation of first derivatives, the forward difference is a first order method while central and complex step methods are second order. Therefore we expect reduced estimation error from the last two methods in general. Looking at the plots, the forward difference shows optimal stepsize for all three problems. Central difference method shows monotonic behavior for x^2 and the estimation error is essentially zero for complex step method for this function. This is the case because the function itself is of second order or quadratic. The error in central-difference method at very small stepsizes for this function is due to loss of precision in the subtraction step. Since there is no subtraction operation in complex step, the error remains essentially zero.
For x^3, both central difference and complex step methods shows optimal stepsizes, which one usually observes in more general cases. At very small stepsize, the error is primarily due to loss of precision and at much larger stepsize, the error is primarily due to approximation error.

For the second derivative approximation, the absolute error for central-difference method grows at a much higher rate for very small stepsize while it levels off for complex step estimate. The optimal stepsize for second derivative approximation is much larger as compared to the same for first derivative approximation.

For these three problems, a step size of 10^{-6} seems adequate if for forward difference scheme is used for estimating first derivative. However a larger stepsize of 10^{-4} or 10^{-3} might be adequate for central difference or complex step schemes. For the second derivative, a step size of 10^{-3} would be advisable for these problems. The choice of stepsize would also depend on the computational resources/time and the accuracy required.

Part A2

The revenue from this flight is given by:

$$J(p_1, p_2, p_3) = p_1 D_1 + p_2 D_2 + p_3 D_3 \qquad (1)$$

This function has to be maximized under the equality constraint (i.e., assuming all seats are occupied) or you can solve this as an inequality constraint as well, but one can observe that this would be active since this is the bounding constraint on the objective (e.g., to limit the revenue one needs to set a limit of the available capacity):

$$h = \sum_{i=1}^{3} D_i - 150 = \sum_{i=1}^{3} a_i e^{-\frac{p_i}{a_i}} - 150 = 0 \quad (2)$$

Hence the formal problem statement can be written as:

$$\begin{aligned} &\min_p \ -J \\ &s.t\ h = 0 \\ &\quad p_i \geq 0 \end{aligned}$$

We can write the Lagrangian as follows by replacing the objective function by –J and converting into a minimization problem of (-J):

$$L = -J + \lambda h \quad (3)$$

Applying KKT condition on the Lagrangian results in $\nabla L(p_i, \lambda) = 0$, which implies;

$$p_i = a_i + \lambda, \ \forall i \qquad (4)$$

$$\sum_{i=1}^{3} a_i e^{-\frac{p_i}{a_i}} - 150 = 0 \qquad (5)$$

Using (4) in (5) and then using fsolve in MATLAB®, we get:

$$\begin{aligned} \lambda &= 56.7484 \\ p_1 &= 156.7484 \\ p_2 &= 206.7484 \\ p_3 &= 356.7484 \end{aligned} \quad (6)$$

The optimal revenue, $J^* = \$43671$. The demand or number of seats in each category (after rounding) is: [21, 38, 91].
To check if it a local minimum, investigate the Hessian of the objective function at the optimal point:

$$H_{(-J)} = \begin{bmatrix} e^{-\frac{p_1}{a_1}}(2 - p_1/a_1) & 0 & 0 \\ 0 & e^{-\frac{p_2}{a_2}}(2 - p_2/a_2) & 0 \\ 0 & 0 & e^{-\frac{p_3}{a_3}}(2 - p_3/a_3) \end{bmatrix} = \begin{bmatrix} 0.0902 & 0 & 0 \\ 0 & 0.1567 & 0 \\ 0 & 0 & 0.2469 \end{bmatrix}$$

which is positive definite since its eigenvalues are all positive (they are in fact the diagonal entries of the diagonal Hessian matrix above).

It was derived in class that the largange multiplier λ is the negative of sensitivity of objective function to the constraint at the optimum.
Hence we can write:

$$\frac{d(-J)}{dN} = -\lambda$$
$$d(-J) = -56.7484 * 3 = -170.24$$

Therefore the revenue would increase by \$170.24 on increasing the number of seats by 3 (assuming all seats are occupied).

Now, the number of seats is increased from 150 to 153. We can solve this problem approximately by using sensitivity analysis (rather than solving the optimization problem again).

We can write the constraint equation as:

$$h = \sum_{i=1}^{3} D_i - N = \sum_{i=1}^{3} a_i e^{-\frac{p_i}{a_i}} - N = 0 \quad (7)$$

where ‘N’ changed from 150 to 153. The sensitivity equations are summarized as below (refer to class notes);

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{Bmatrix} \delta p \\ \delta\lambda \end{Bmatrix} + \begin{Bmatrix} c \\ d \end{Bmatrix} = 0$$

$$A_{ik} = \frac{\partial^2 J}{\partial p_i \partial p_k} + \sum_{j\in M} \lambda_j \frac{\partial^2 h}{\partial p_i \partial p_k}$$

$$B_{ij} = \frac{\partial h}{\partial p_i}$$

$$c_i = \frac{\partial^2 J}{\partial p_i \partial N} + \sum_{j\in M} \lambda_j \frac{\partial^2 h}{\partial p_i \partial N}$$

$$d_j = \frac{\partial h}{\partial N}$$

$$\delta p = \begin{Bmatrix} \frac{\partial p_1}{\partial N} & \frac{\partial p_2}{\partial N} & \frac{\partial p_3}{\partial N} \end{Bmatrix}^T$$

$$\delta\lambda = \frac{\partial \lambda}{\partial N}$$

Using the values at the optimal solution, this system can be written as:

$$\begin{bmatrix} 0.2086 & 0 & 0 & -0.2086 \\ 0 & 0.252 & 0 & -0.252 \\ 0 & 0 & 0.3045 & -0.3045 \\ -0.2086 & -0.252 & -0.3045 & 0 \end{bmatrix} \begin{Bmatrix} \delta p \\ \delta\lambda \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \qquad (8)$$

Solving (8) we get, $\left[\frac{\partial p_1}{\partial N} = -1.3071;\ \frac{\partial p_2}{\partial N} = -1.3071;\ \frac{\partial p_3}{\partial N} = -1.3071;\ \frac{\partial \lambda}{\partial N} = -1.3071 \right]$.

Now using the linear approximation, we have:

$$\Delta p = \Delta N \delta p = [-\$3.92\ \ -\$3.92\ \ -\$3.92]$$

Hence the airline should reduce the ticket price by \$3.92 in all segments with the addition of 3 seats.

The new prices in the three categories are:

$$p_1 = \$152.827$$
$$p_2 = \$202.827$$
$$p_3 = \$352.827$$

The optimal revenue, $J^* = \$43837$. The demand or number of seats in each category (after rounding) is: [22, 39, 92]. The revenue has increased by \$166. This is quite close to that predicted earlier. Since we have solved it using gradient-based methods although the numbers of seats are discrete variables, there are some errors due to rounding operations. The results can be checked by re-running the optimization which returns revenue of \$43835, an increase of \$164. This is extremely close to what we got using sensitivity analysis.

Alternative solution method: This problem can be solved in different ways, all using the linear approximations.

Observe the KKT equations (4). We can write, $\frac{\partial p_i}{\partial \lambda} = 1, \forall i$. Now from (5), after replacing p_i's with λ, and then on differentiation: $\frac{\partial \lambda}{\partial N} = -1.3071$.

Now using linear approximation, $\Delta\lambda = \frac{\partial \lambda}{\partial N}\Delta N = -3.92$.

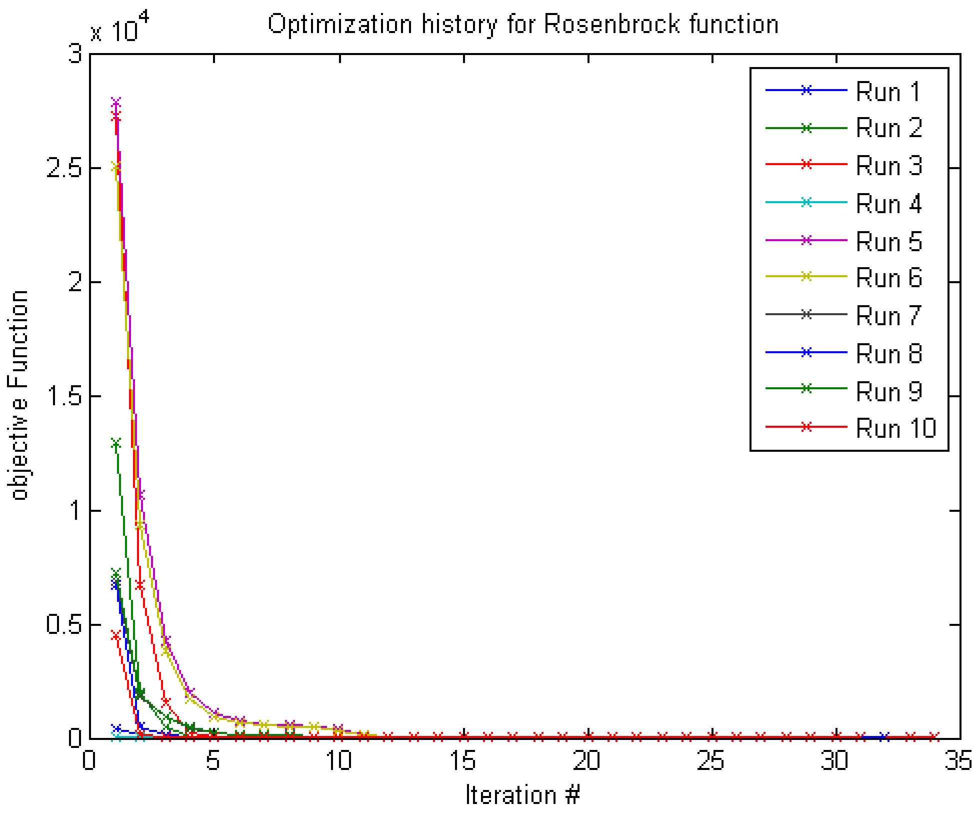
Now we have, $\Delta p = \Delta\lambda \frac{\partial p}{\partial \lambda} = -3.92 * [111] = [-3.92 \ -3.92 \ -3.92]$.

So we arrive at the same result as before and we can easily compute other quantities of importance.

Part A3

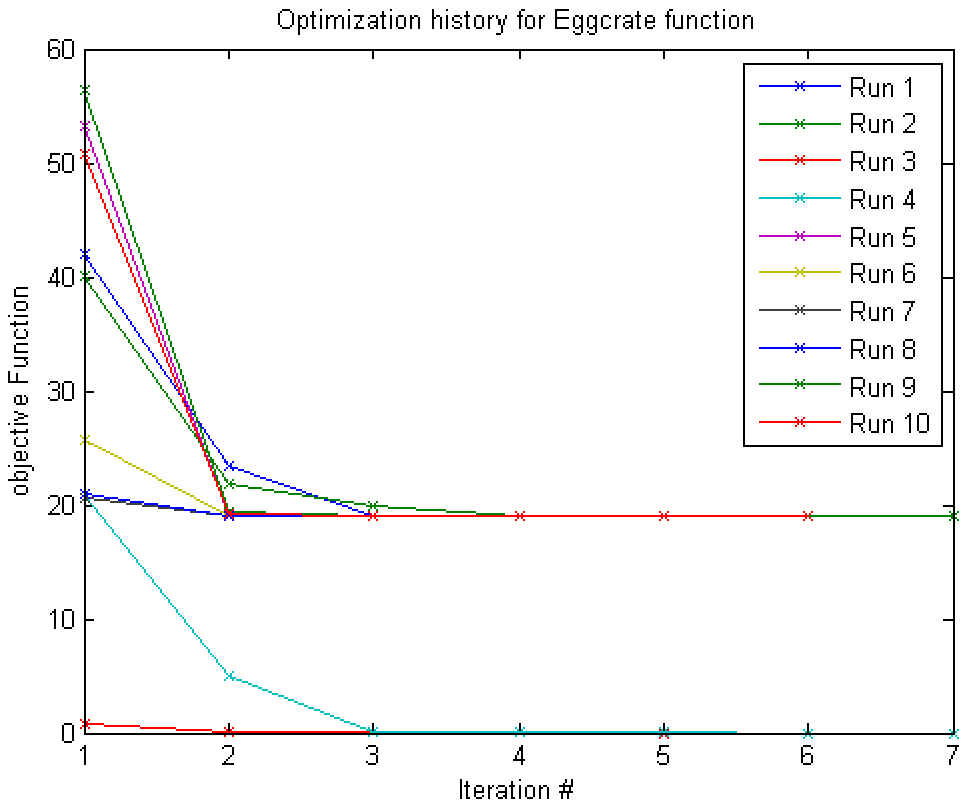
In minimizing Rosenbrock function, quasi-newton method (implemented in MATLAB's fminunc formulation) was used with random starting points chosen within bounds [-5, 5]. Same random number was used for generating the components of the starting vector. This makes both components of the starting vector the same in this problem. In all cases, the global minimum was found within 6 significant digits.

Run #	Starting point (x0)	Optimal point (x*)	Optimal objective function	Feasible?	Time (Sec.)
1	[1.991, 1.991]	[1,1]	0.00000	Yes	0.0274
2	[3.91, 3.91]	[1,1]	0.00000	Yes	0.0351
3	[4.593, 4.593]	[1,1]	0.00000	Yes	0.0386
4	[0.472, 0.472]	[1,1]	0.00000	Yes	0.0215
5	[-3.6137, -3.6137]	[1,1]	0.00000	Yes	0.0258
6	[-3.5070, -3.5070]	[1.1]	0.00000	Yes	0.0252
7	[-2.424, -2.424]	[1,1]	0.00000	Yes	0.0263
8	[3.4071, 3.4071]	[1,1]	0.00000	Yes	0.0358
9	[-2.4571, -2.4571]	[1,1]	0.00000	Yes	0.0232
10	[3.1428, 3.1428]	[1,1]	0.00000	Yes	0.0306



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, 8 were stuck in a local minimum while 2 found the global optima.

Run #	Starting point (x0)	Optimal point (x*)	Optimal objective function	Feasible?	Time (Sec.)
1	[3.7096, 3.7096]	[3.0196, 3.0196]	18.976	Yes	0.02
2	[-3.9348, -3.9348]	[-3.0196, -3.0196]	18.976	Yes	0.016
3	[-0.1286, -0.1286]	[0.0000, 0.0000]	0.00000	Yes	0.011
4	[-0.6837, -0.6837]	[0.0000, 0.0000]	0.00000	Yes	0.011
5	[1.8386, 1.8386]	[3.0196, 3.0196]	18.976	Yes	0.01
6	[2.631, 2.631]	[3.0196, 3.0196]	18.976	Yes	0.011
7	[3.2, 3.2]	[3.0196, 3.0196]	18.976	Yes	0.01
8	[-2.8145, -2.8145]	[-3.0196, -3.0196]	18.976	Yes	0.011
9	[2.2582, 2.2582]	[3.0196, 3.0196]	18.976	Yes	0.01
10	[1.949, 1.949]	[3.0196, 3.0196]	18.976	Yes	0.01

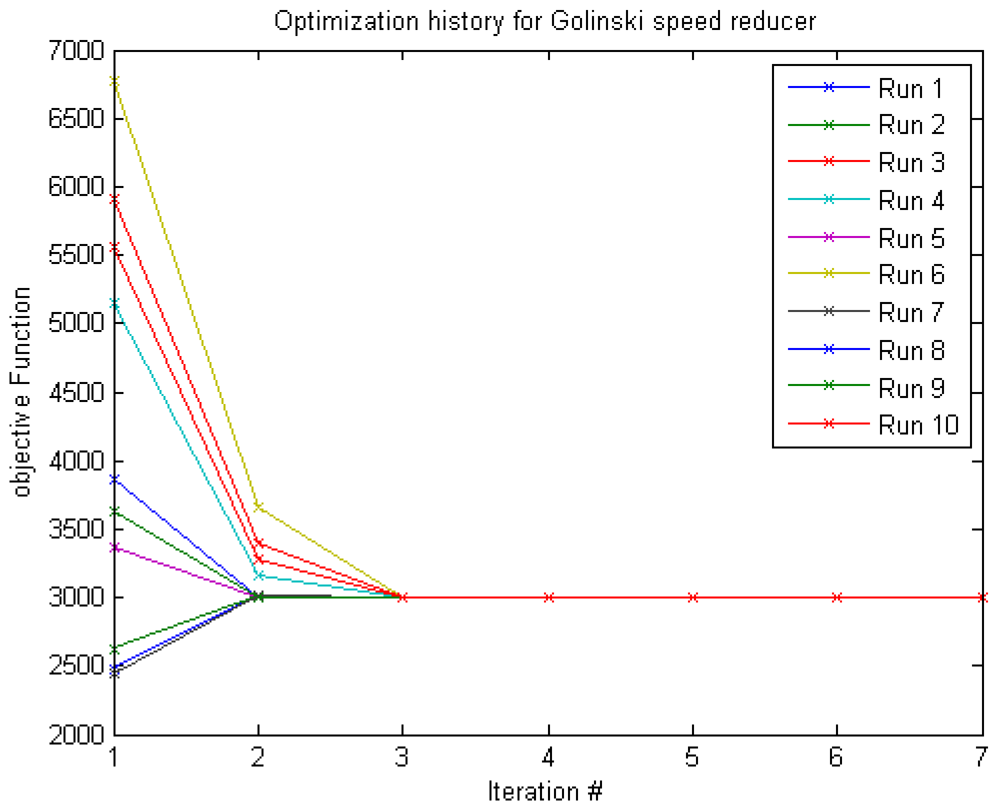


Due to the inherent symmetry in the problem, the local optimum were found at either [3.0196, 3.0196] or [-3.0196,-3.0196], both yield the function value of 18.976. Note that the global optimum of [0.0, 0.0] were found when optimizer started with guess close to that point (i.e., it fell within the basin of attraction of the global optimum).

Finally the Golinski's speed reducer problem was solved using sequential quadratic programming approach (implemented in MATLAB's fmincon formulation) 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 design space for this problem is quite narrow. The 10 starting points were picked at random as before and the quickly optimizer converged to the optimum in all cases.

Run #	Starting point (x0)	Optimal point (x*)	Optimal objective function (kg)	Feasible?	Time (Sec.)
1	[2.646, 0.7046, 17.5078, 7.3462, 7.3462, 2.9462, 5.023]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.030
2	[2.697, 0.7097, 18.0684, 7.3971, 7.3971, 2.9971, 5.0485]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.032
3	[3.4234, 0.7823, 26.058,	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502,	2994.35	Yes	0.031

		5.2867]			
4	[3.2948, 0.7695, 24.643, 7.9948, 7.9948, 3.5948,5.3474]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.045
5	[2.9171, 0.7317, 20.4881, 7.6171, 7.6171, 3.2171, 5.1585]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.029
6	[3.5502, 0.795, 27.452,8.2502,8.2502,3.8502,5.4751]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.034
7	[2.6344, 0.7034, 17.3789, 7.3344,7.3344,2.9344,5.0171]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.033
8	[3.0387, 0.7438, 21.8262, 7.7387,7.7387, 3.3387, 5.2194]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.025
9	[2.9815, 0.7381, 21.197, 7.6815, 7.6815, 3.2815, 5.1908]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.027
10	[3.3655, 0.7765, 25.42,8.0655,8.0655,3.6655,5.3827]	[3.5, 0.7, 17.0, 7.3, 7.7153, 3.3502, 5.2867]	2994.35	Yes	0.043



The eggcrate function is multi-modal while golinski's speed reducer weight and Rosenbrock's problem have single minimum. For the eggcrate function, depending on where one starts, decides which minimum is found. If one starts near the global minimum, it will converge to the global solution.

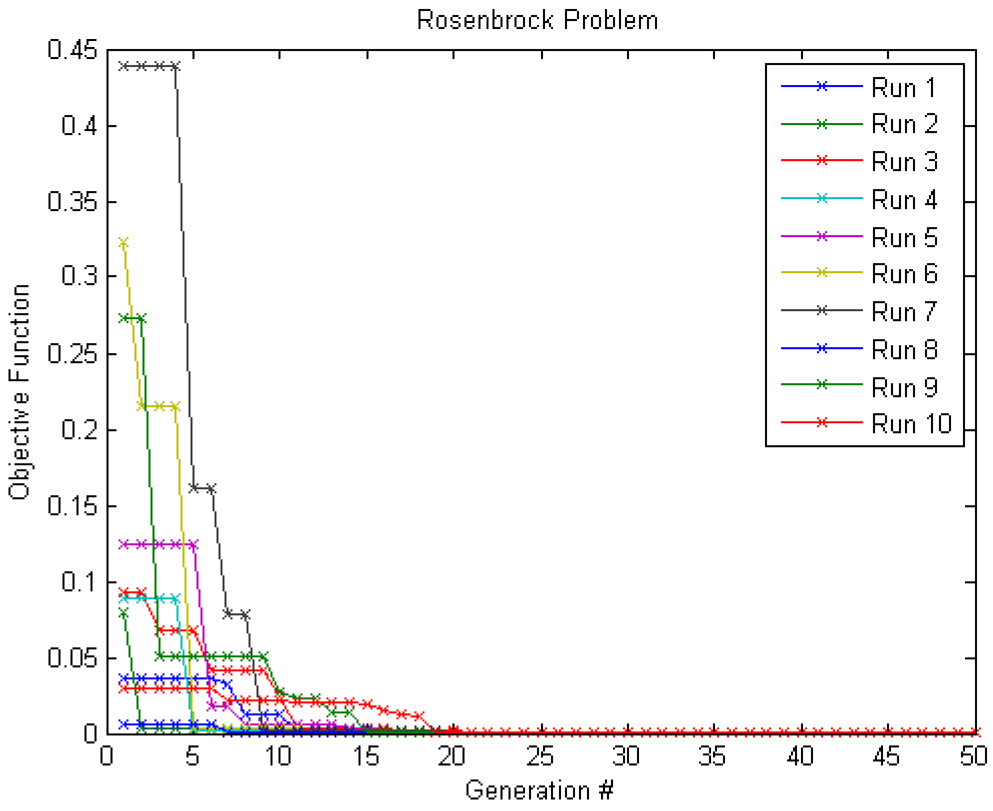
Part A4

The heuristic technique described here pertains to Genetic Algorithms. This problem can be handled by using other heuristic methods like simulated Annealing, Particle Swarm Optimization, etc.

The MATLAB GA toolbox is used as the primary optimization tool. The GA implementation in MATLAB is a little different that traditional GA. Therefore the crossover and mutation parameters used here are a little different. The crossover parameter used in MATLAB GA toolbox refers to a fraction which determines how many individuals would be picked for crossover operation in a generation. The mutation is performed a parent randomly, and the mutation rate is computed based on the diversity of the population. For example, in a generation, #individuals subjected to crossover = CrossoverFraction*(popsize – elite count); #individuals subjected to mutation = (popsize – crossover pop – elite count). The amount of mutation depends on the spread of the generation and decreases with increasing generation count. No parent solution is acted upon by both crossover and mutation at the same time. The mutation rate can be increased by using an increased initial range. In all cases, there are some elite population members who are transferred to the next generation automatically.

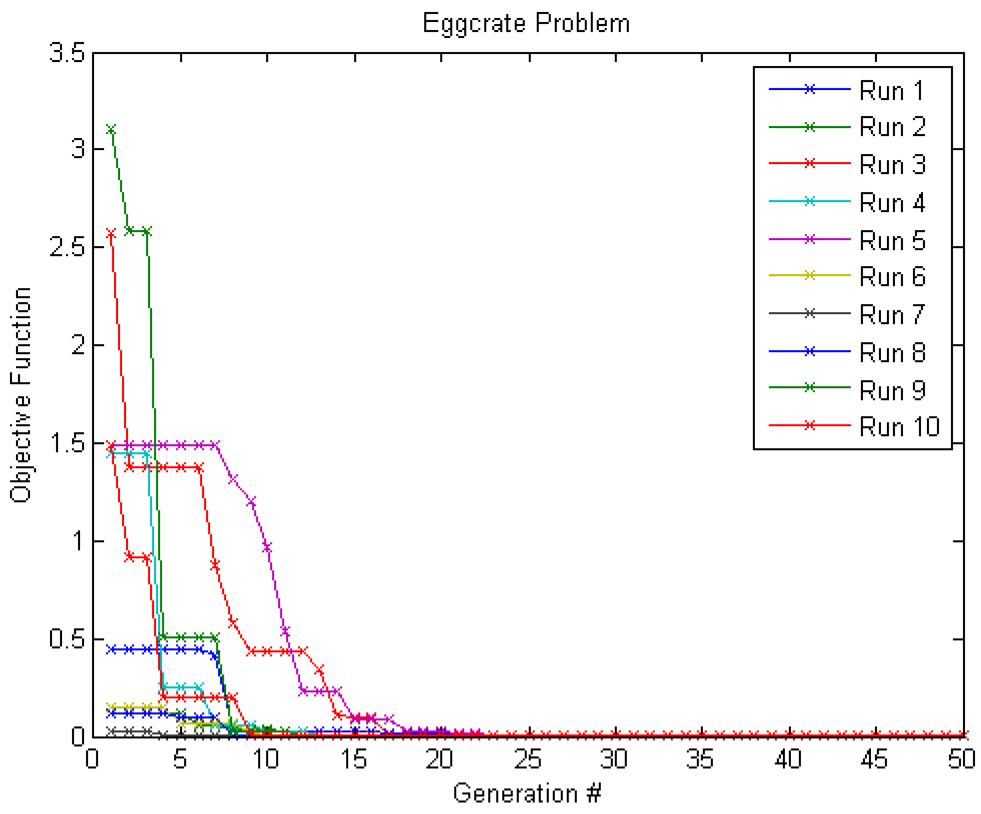
In all cases, the following sequence was used to arrive at the tuned parameter set:

(1) Study the effect of increased generation number and then check the interplay of population size and generation number.
(2) Change the crossover fraction and obtain the best possible setting.
(3) In case of eggcrate problem, use a higher mutation to explore the design space better. The mutation rate should also be a little higher (this means a smaller crossover fraction).
(4) Use mutationadaptfeasible as the mutation function to account for constraints.



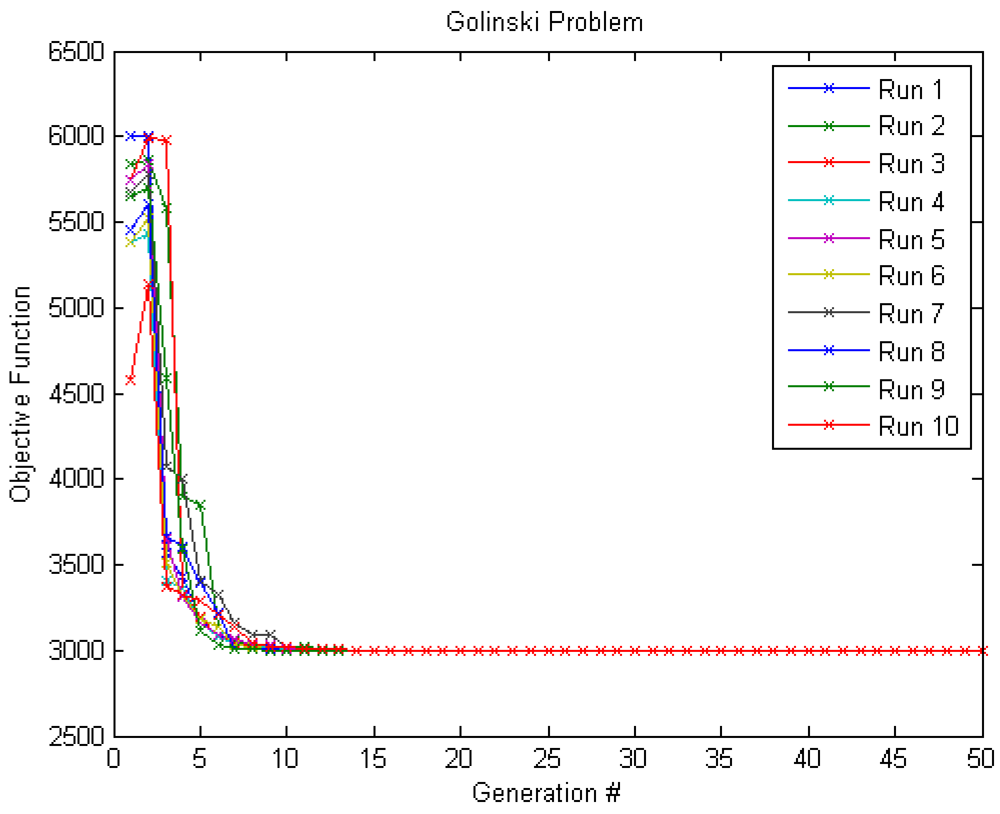
The final tuned results are shown above. The tuned parameters were: [popsize = 20, generation = 50, crossover fraction =0.4, Initial Penalty = 1000, Tolerance Function =1e-16, Elite Count = 3]. In all cases, the solution converged to objective value of 0.00 and the optimal point [1, 1]. Observe that, in all cases, GA has converged very close to the optimal solution by at most 20 generations and subsequent progress is very slow. This is a typical feature of GA's.

Next is the eggcrate function. This function is multi-modal (i.e., has multiple minima with one global minimum). Care should be taken to ensure that GA avoids being stuck in local minima.



Incidentally, MATLAB's default parameters performed quite well on this function. The final tuned parameters are: [popsize = 20, Generations = 50, Elite count = 1, Crossover = 0.8, Initial Penalty = 1000]. In all cases, the solution converged to the global optimum [0, 0] with objective function value of 0.00, and they converged very close to the solution by al most 25 generations. Beyond that it is a very slow progress. At this point, it is better to shift to a gradient-based optimizer (if problem type permits) using the best GA solution as initial guess. You would notice that the gradient-based optimizer converges to the true optimal in very few (generally 2 – 5) iterations.

For weight minimization of Golinski's speed reducer, there are 11 constraints, in addition to bound constraints. MATLAB's GA toolbox uses a constraint handling scheme similar to that of interior point method. There are two parameters, Initial Penalty and Penalty Factor. Initial penalty acts as the penalty factor to start with. If constraints are not satisfied on solving the internal subproblem, then the penalty factor is used to increase the penalty term.



s

The parameters used in this problem, across different runs are: [popsize=20, generations = 50, crossover fraction = 0.9, elite count = 3, initial penalty = 100, penalty factor =10, Tolfun = 1e-08, Tolcon = 1e-16].
In all cases, solution converged to a weight of around 2994.37 kg, which is very close to the solution found by gradient-based method (i.e., 2994.35 kg). In all cases, the solution converged to the global optimum [3.5, 0.7, 17.0, 7.3, 7.7154, 3.3502, 5.2867] with objective function value of 0.00, and they converged very close to the solution by al most 15 generations.

In general, GA performs well on all three problems, but required "tuning" of parameters, different for each problem. The computational effort with GA is about 100 times that of gradient-based method. In case of eggcrate problem, this is more than 250 times. This is due to presence of multiple local minima and GA had to avoid them. Implementation of "niching" strategies in the algorithm might improve efficiency.

(i) Dependence on the initial design vector/population:

Problem Name	Gradient-based Optimizer	GA
Rosenbrock	**Low** (regardless of the starting points, all instances converged to the optimal solution).	**Low** (this may depend on the "tuning" parameters)
Eggcrate	**High**	**Low** (depends on the tuning parameters selected)
Golinski's speed reducer	**Low**	**Low** (converges very close to global optimum quite quickly)

(ii) Computational effort in terms of CPU time:

Problem Name	Gradient-based Optimizer (Sec.)	GA (Sec.)
Rosenbrock	0.028	2.69
Eggcrate	0.01	2.5
Golinski's speed reducer	0.04	48

(iii) Convergence history:

Problem Name	Gradient-based Optimizer	GA
Rosenbrock	Always converged to global minimum.	Converged, but efficiency depends on the tuning parameters selected.
Eggcrate	Always converged, but either a local or a global minimum.	Converged, but efficiency depends on the tuning parameters selected. This took more time (about 250 times more than gradient-based method) due to presence of local minimum. The efficiency can be improved further by embedding "niching" techniques that modifies the mating criteria.
Golinski's speed reducer	Always converged to the global minimum.	Converged, but efficiency depends on the tuning parameters selected.

(iv) Frequency of getting trapped in a local minimum:

Problem Name	Gradient-based Optimizer	GA
Rosenbrock	zero	zero
Eggcrate	80%	zero
Golinski's speed reducer	zero	zero

Comments:

1. 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.
2. Genetic Algorithms are computationally expensive and requires considerable "tuning" effort, especially for complex problems. The GA toolbox in MATLAB is not the most efficient of implementations and does not follow the traditional GA parameters. This might make the tuning process more challenging.

3. An idea of the design space is often very helpful to tune, both gradient-based and Genetic Algorithms and can significantly expedite the convergence.
4. Genetic Algorithms can improve the objective function quickly but tends to slow down as it approaches the global minimum.

In general, the above comments would hold. An understanding of the problem being solved and its associated design space would go a long way in solving optimization problem more appropriately/efficiently. Generally GA is very helpful to explore the design space when the problem at hand is not very well-understood.