Constraints Optimization Problems using bracket-operator penalty method

Naren M (M.Tech),
Department of Mechanical Engineering,
Indian Institute of Technology Guwahati,
Assam, India
naren_m@iitg.ac.in

Ashwinee Narware (M.Tech),
Department of Mechanical Engineering,
Indian Institute of Technology Guwahati,
Assam, India
nashwinee@iitg.ac.in

Dr. Deepak Sharma, (Associate Professor)
Department of Mechanical Engineering,
Indian Institute of Technology Guwahati,
Assam, India
dsharma@iitg.ac.in

***Abstract*—In this paper constraints optimization problems are solved using bracket-operator penalty function method. Penalty function method transform constrained problem into a sequence of unconstrained problems. These unconstrained problems are then solved using Cauchy's Steepest-Descent multi-variable optimization method. The Cauchy's Steepest-Descent method uses negative of gradient at any particular point as search direction. Along this search direction optimum point is found out using single-variable optimization techniques and taken as next point. We have used Bounding Phase Bracketing Method to bracket the optimum point and then Bisection Gradient-based Method to improve the accuracy of the solution.**

Keywords—constrained optimization, penalty function method, steepest-descent method, bounding phase method, bisection method, bracket-operator penalty

I. Introduction

Optimization algorithms are becoming increasing popular in multi-engineering design activities. In engineering design activities, objective function that are to be optimized are almost always subjected to satisfying some constraints. These constraints may arise due to space, strength, or stability consideration. There are many constraint optimization algorithms. In this paper we are studying the bracket-operator penalty function method for optimization of constrained problems. As it is transformation method of optimization technique the constrained optimization problem is transformed into unconstrained optimization problem. This unconstrained optimization problem can be solved various direct search and gradient search multi-variable optimization algorithm. The unconstrained optimization problem is solved using Cauchy's Steepest Descent method. Bounding phase bracketing method and bisection gradient-based methods are used to find the optimum in the various search direction created by Steepest Descent method. This algorithm is used to evaluate a set of constrained optimization problem. The same problems are also solved using various standard optimization solver to validate the result.

II. Optimization Algorithms

A. Bracket-operator penalty function method

Penalty function constrained optimization method is a transformation constrained problem method. In transformation method the constrained problem is transformed into a sequence of unconstrained problems by adding penalty terms for each constraint violation.

Bracket operator penalty is exterior penalty method. The bracket operator assigns a positive value to the infeasible points. This penalty function can handle both equality and inequality constraints. This function can be defined as

$$\Omega = R\langle g(x)\rangle^2 \tag{1}$$

where $\langle\alpha\rangle = \alpha$, when α is negative; zero, otherwise.

Algorithm 1 Bracket-operator penalty method [1]

Step 1 Choose two termination parameters ϵ_1, ϵ_2, an initial solution $x^{(0)}$, a penalty term Ω, and an initial penalty parameter $R^{(0)}$. Choose a parameter c to update R such that $0 < c < 1$ is used for interior penalty terms and $c > 1$ is used for exterior penalty terms. Set $t = 0$.

Step 2 Form $P(x^{(t)}, R^{(t)}) = f(x^{(t)}) + \Omega\left(R^{(t)}, g(x^{(t)}), h(x^{(t)})\right)$.

Step 3 Starting with a solution $x^{(t)}$, find $x^{(t+1)}$ such that $P(x^{(t+1)}, R^{(t)})$ is minimum for a fixed value of $R^{(t)}$. Use ϵ_1 to terminate the unconstrained search.

Step 4 If $\left|P(x^{(t+1)}, R^{(t)}) - P(x^{(t)}, R^{(t-1)})\right| \le \epsilon_2$ set $x^T = x^{(t+1)}$ and Terminate; Else go to Step 5.

Step 5 Choose $R^{(t+1)} = cR^{(t)}$. Set $t = t + 1$ and go to Step 2.

B. Cauchy's (steepest descent) Method

Cauchy's method uses the negative of gradient at any particular point as search direction. The unidirectional search is performed in this search direction to find the minimum point along that direction. The minimum point becomes the current point and the search is continued from this point.

Algorithm 2 Cauchy's steepest descent method [1]

Step 1 Choose a maximum number of iterations M to be performed, an initial point $x^{(0)}$, two termination parameters ϵ_1, ϵ_2 and set $k = 0$.

Step 2 Calculate $\nabla f(x^{(k)})$, the first derivative at the point $x^{(k)}$.

Step 3 If $\left\|\nabla f(x^{(k)})\right\| \le \epsilon_1$, Terminate; Else if $k \ge M$; Terminate; Else go to Step 4.

Step 4 Perform a unidirectional search to find $\alpha^{(k)}$ using ϵ_2 such that $f(x^{k+1}) = f\left(x^{(k)} - \alpha^{(k)}\nabla f(x^{(k)})\right)$ is minimum. One criterion for termination is when $\left|\nabla f(x^{(k+1)}) . \nabla f(x^{(k)})\right| \le \epsilon_2$.

Step 5 If $\frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\left\|x^{(k)}\right\|} \le \epsilon_1$, Terminate; Else set $k = k + 1$ and go to Step 2.

C. Bounding Phase Method

Bounding Phase method is used to bracket the minimum of a function. The algorithm begins with an initial guess and thereby finds a search direction based on two more function evaluations in the vicinity of the initial guess.

Algorithm 3 Bounding Phase method [1]

Step 1 Choose an initial guess value $x^{(0)}$ and an increment Δ, Set $k = 0$;

Step 2 If $f(x^{(0)} - |\Delta|) \geq f(x^{(0)}) \geq f(x^{(0)} + |\Delta|)$, then Δ is positive; Else if $f(x^{(0)} - |\Delta|) \leq f(x^{(0)}) \leq f(x^{(0)} + |\Delta|)$, then Δ is negative; Else go to step 1;

Step 3 Set $x^{(k+1)} = x^k + 2^k\Delta$;

Step 4 If $f(x^{(k+1)}) < f(x^{(k)})$, set $k = k + 1$ and go to Step 3; Else the minimum point lies in the interval $(x^{(k-1)}, x^{(k+1)})$ and Terminate;

D. Bisection Method

Bisection method is gradient-based optimization method. In this method both the function value and the sign of the first derivative is used to eliminate a certain portion of the search space. Number of required function evaluation is less than the Newton-Raphson method.

Algorithm 4 Bisection method [1]

Step 1 Choose two points *a* and *b*, such that $f'(a) < 0$ and $f'(b) > 0$. Also, choose a small number ϵ. Set $x_1 = a$ and $x_2 = b$.
Step 2 Calculate $z = (x_2 + x_1)/2$ and evaluate $f'(z)$.
Step 3 If $|f'(z)| \leq \epsilon$, Terminate; Else if $f'(z) < 0$ set $x_1 = z$ and go to step 2; Else if $f'(z) > 0$ set $x_2 = z$ and go to step 3

III. Result and Discussion

The above algorithm was implemented in python language. Three uniquely defined problem was solved using this algorithm. The mathematical equation of these constrained problems is given in the appendix. Since, the program was made to calculate the gradients numerically the accuracy of the result obtained using low level language like C will be less if proper care is not taken about the significant digits while representing the different variables. The high-level language like python or MATLAB take care by assigning the proper data type during the run-time. The package NumPy [2] in python was used to store vectors and process the same. The NumPy package uses vectorization of data to process multiple data with single instruction (SIMD) which make computationally expensive algorithm used here run faster. The hardware used to run the test problem is given in the table I. For verification the same problem was solved in MATLAB using $fmincon()$ function.

A. Test Problem 1

The contour of the first test problem is given in Fig. 1. The infeasible region is masked out in this color contour plot. It can be observed that the feasible region is narrow when compared to the bounds of the variables. It also becomes narrower near the minimum point. We expect that there will be large variation in optimum point obtained through this algorithm, as we are using the numerical methods to calculate gradient. The large number of function evaluation can also increase the variation of the optimum point due to the error that can creep out. So, to get better optimal point the proper selection of penalty parameters, rate of increase of penalty parameters with iteration is important.

TABLE I. HARDWARE INFORMATION

Hardware Used		
Component	***Make***	***Model or Build***
CPU	Intel Pentium	N3710
	Clock speed	1.6GHz (up to 2.56GHz)
RAM	Samsung	4GB DDR3 @1600MHz
Operating System	Windows 10 Home Single Language	64-bit 19042.630
Software relase	Python	3.8.3
	NumPy	1.19.2
	MATLAB	2020b Update 2

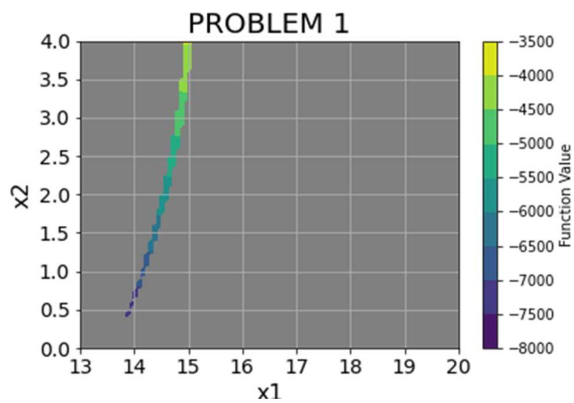

Fig. 1. Contour plot of 1st test problem (feasible region only)

a) Selection of penalty parameters

From the color bar of the contour map we can observe that function value in the region vary in the order of 10^3. Bracket operator penalty method is exterior penalty function method, so the initial value of penalty parameter is less and the increase should be greater than 1. At termination of penalty methods if the penalty parameter is less, we cannot expect that the constrained equations are taken care properly.

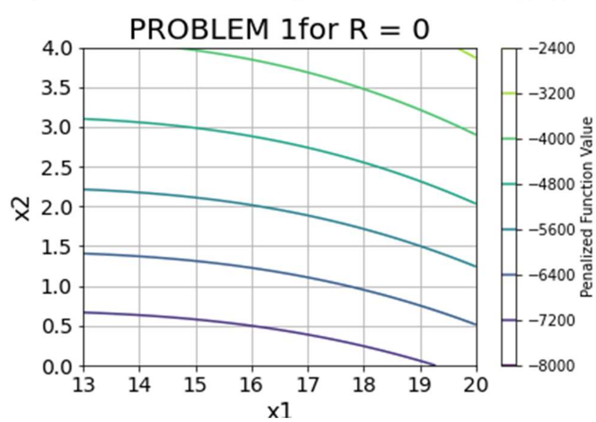

Fig. 2. Contour of the objective function

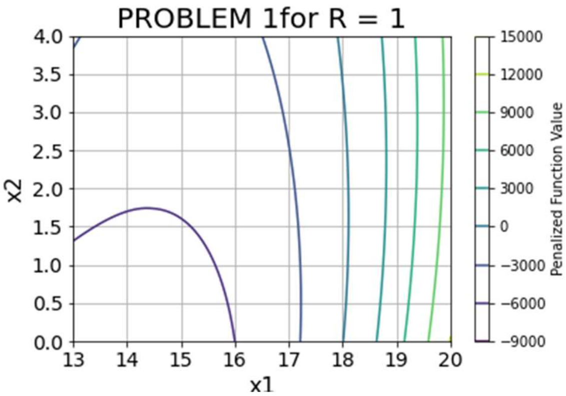

Fig. 3. Distorted contour plot for $R = 1$

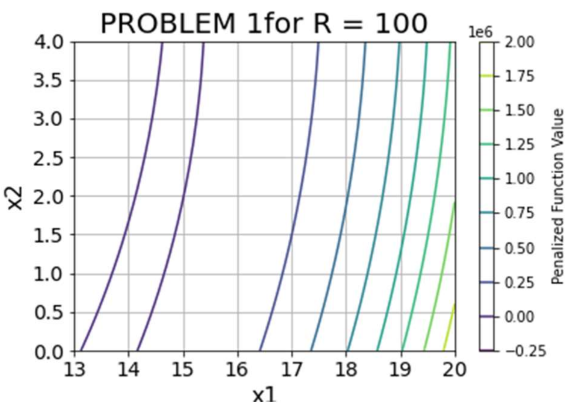

Fig. 4. Distorted contour plot for $R = 100$

The use of bracket operator penalty function distorts the given function which can have many artificial minima. This can be observed in the contour plot (Fig. 2, Fig. 3. And Fig. 4.) for the modified equation to accommodate bracket operator penalty function. If the starting point of the search space is far from the optimum point then large number of iterations are to be carried out with less penalty constant to minimize the probability of getting stuck at this artificial optimum points.

b) Selection of starting point of search space

The initial point is selected randomly within the search space. The initial penalty parameter is selected every low to account for the initial point that are far from the minimum point. The function value and time taken by the program to solve vary with selection of initial.

c) Result obtained

The algorithm was run for 10 times and the result of the same is tabulated in Table II. The convergence plot for the best-obtained solution is plotted in Fig. 5. The statistics of the data obtained is given in Table III. It can be observed that the position of the initial starting point with respect to optimum point play major role in the accuracy of the result. The difference between the two points does not matter much. This can be observed in iteration no. 1,3 and 4 as opposed to 7 in the Table II. This can be explained by the exterior penalty nature of the bracket operator function. For this problem if the iteration is started from far right of the bounds, we get better result because the optimum point lies in the same direction in the feasible region. The optimum value of the function was obtained using the penalty parameter, $R_0 = 10^{-3}$ And rate of increase of penalty parameter per iteration was taken as $c = 5$.

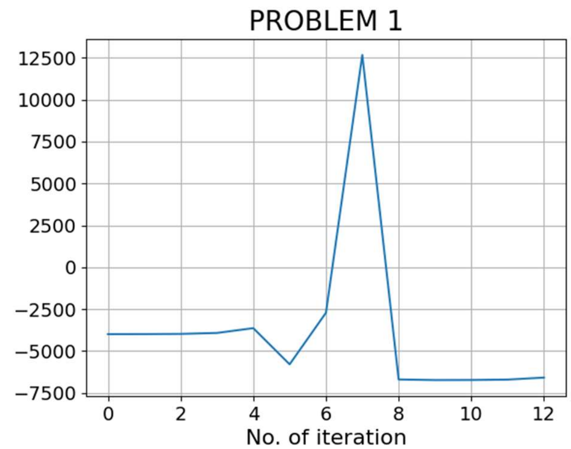

Fig. 5. Convergence plot for problem #1

TABLE II. RESULT OF 1ST PROBLEM

	Result			
	Initial starting point	***Final starting point***	***Time taken (s)***	***Function value***
1	[19.4309, 3.3292]	[14.0179, 0.6984]	0.199	-7136.0975
2	[15.9300, 0.1678]	[13.635, 0.0016]	0.159	-7950.0152
3	[19.9825, 3.9467]	[14.1740, 0.9978]	0.198	-6788.6467
4	[19.8330, 2.1091]	[14.2458, 1.1901]	0.285	-6578.683
5	[15.0571, 0.6297]	[13.7661, 0.2203]	0.14	-7685.072
6	[19.58597, 1.65958]	[14.30513, 1.33742]	0.206	-6420.2314
7	[14.98799, 1.76025]	[13.7849, 0.2344]	0.189	-7667.820
8	[14.8422, 2.8802]	[14.84225, 2.8802]	0.00799	-4904.091
9	[19.46399, 3.06539]	[14.1536, 0.9485]	0.187	-6843.299
10	[16.7347, 3.8765]	[13.7572, 0.2052]	0.147	-7703.195

TABLE III. COMPARISON OF THE RESULT

Statistics of Result		
Best	x	[14.1536,0.9485]
	f(x)	-6843.299
MATLAB	x	[14.095,0.8430]
	f(x)	-6961.8
Worst	f(x)	-4904.091
Mean	f(x)	-6967.6459
Median	f(x)	-6989.6945
Standard Deviation	f(x)	900.411

B. Test Problem 2

The contour plot of the second plot is given in Fig. 6. The infeasible region is masked out in this plot. It can be observed that the there are many local optimum points in the feasible region. This an oscillating function. The infeasible region also contains number of local minimum point.

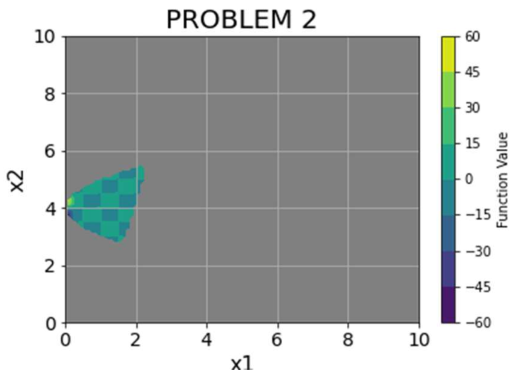

Fig. 6. Contour plot of the 2nd problem

The global maximum is at the boundary of the feasible region. For some initial point that are near global maxima we can get good result using the bracket operator penalty function. This is because of bracket operator penalty function penalized the infeasible region. The search point when using this method will be in infeasible region most of the time. There is also very good chance that the selected algorithm will convergence to the local maxima. The penalty function must be evaluated number of times before getting the required global maximum point.

a) Selection of penalty paramaters

From the color bar of the contour plot in Fig. 6. it can be observed that the function value through out the feasible region is of the order 10. So, the penalty parameter at the time of termination condition need not be high as in the case of first test problem. But, the intial penalty parameter should be low as posible to avoid the posible distortion and getting stuck in the artificial minimum.

b) Selection of starting point of search space

Same as the first problem the initial point for the algorithm is obtained randomly using the inbuilt function available in NumPy package ($numpy.random.rand()$).

TABLE IV. RESULT OF 2ND PROBLEM

	Result			
	Initial starting point	***Final starting point***	***Time taken (s)***	***Function value***
1	[3.10498, 7.4582]	[1.22799, 4.2434]	0.072	0.09582
2	[0.04714, 7.41103]	[2.0053, 5.0064]	0.319	2.6786e-8
3	[8.52705, 3.48986]	[1.2277, 3.2604]	0.142	0.1167
4	[8.1904, 7.8912]	[0.7133, 3.7443]	0.120	0.56997
5	[8.3590, 8.8517]	[1.2504, 3.2598]	0.150	0.113
6	[5.7182, 2.4099]	[1.2273, 3.2604]	0.117	0.1166
7	[4.0468, 2.6546]	[0.7357, 2.6547]	0.119	0.604
8	[6.6998, 1.1935]	[1.5765, 1.1935]	0.132	-0.132
9	[6.1958, 6.5454]	[1.6844, 4.7461]	0.095	0.025
10	[8.5024, 6.5215]	[0.7151, 3.7456]	0.103	0.5699

c) Result obtained

The algorithm for this problem was iterated for 10 times and the result of the same is tabulated in Table IV. The convergence plot for the best obtained solution is plotted in Fig. 7. The statistics of the result obtained is given in Table V. It can be observed that from statistics of the result, there is a lot of variation in the result. Even though the best matching result was obtained this is due to the fact that there are lot of local maxima in that region. The optimum result for this was obtained for lot of values of penalty parameters and increase rate. But the variation for most of selected parameter remains the same.

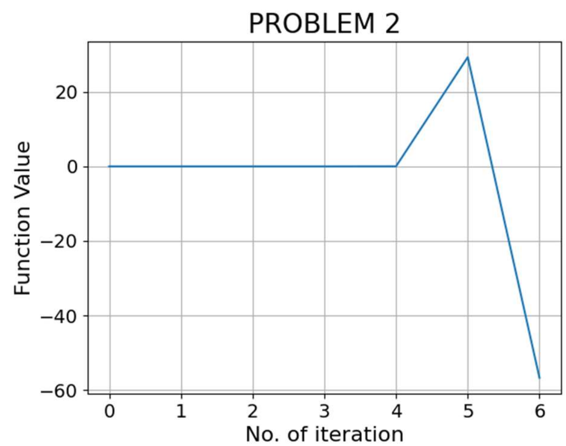

Fig. 7. Convergence plot of Problem 2

TABLE V. COMPARISON OF THE RESULT

Statistics of Result		
Best	x	[1.22799,4.2434]
	f(x)	0.09582
MATLAB	x	[1.2280, 4.2454]
	f(x)	0.0958
Worst	f(x)	0.5699
Mean	f(x)	0.20789
Median	f(x)	0.1148
Standard Deviation	f(x)	0.2686

C. Test Problem 3

This function is more than two dimesnional problem. It is actually 8 dimension problem where the objective function just depend on the 3 variable. The function value is of the order $10^3 - 10^4$. So, even small violation in constrained equation will give result that are far from the optimum point.

a) Selection of penalty parameters

As already discused the function value in the feasible region is large. So, care must be taken to select the optimum point that have high penalty parameter at the termination condition.

TABLE VI. RESULT OF 3RD PROBLEM

	Result			
	Initial starting point	*Final starting point*	*Time taken (s)*	*Function value*
1	[1209.7, 5734.3, 9309.5, 269.7, 337.8, 925.2, 18.6, 238.9]	[343.309, 5443.5, 9018.6, 269.74, 337.8, 51.06, 18.6, 142.5]	0.353	14805.356
2	[7439.2, 3417.4, 7060.4, 339.0, 155.6, 129.76, 102.10,803.52]	[7438.94, 3306.06, 7060.06, 33905, 155.61, 129.75, 102.1, 792.08]	0.473	17805.066
3	[637.823, 4040.15, 7986.4, 886.6, 597.1, 86.58, 273.42, 299.94]	[176.30, 3578.63, 7524.96, 122.115, 333.67, 103.08, 10, 10]	0.457	11279.906
4	[1136.6, 6648.3, 9136.2, 268.86, 28.38, 226.8, 782.39, 185.54]	[1136.655, 6588.92, 9136.21, 268.86, 28.38, 226.87, 10, 185.55]	0.458	16861.797
5	[4448.3, 9117.56, 5598.03, 494.94, 875.18, 513.67, 767.76, 633.84]	[4416.35, 9098.52, 5597.99, 469.16, 875.18, 172.30, 10.0, 255.69]	1.284	19112.87
6	[1320.26, 3033.45, 3886.5, 87.75, 448.13, 610.114, 936.50, 628.49]	[1079.40, 2787.66, 3871.13, 87.75, 365.55, 10, 10, 10]	0.408	7738.91
7	[6557.27,3693.16,6843.24,981.25,32.8,766.38,656.27,332.98]	[6457.29,3624.37,6780.311,310.14,10,282.19,725.69,190.67]	0.182	16861.97
8	[1267.81,6418.47,2285.82,142.1037,145.104,614.99, 267.32,829.66]	[1147.79,6298.45,2168.26,10,147.1,757.55,267.32,829.45]	0.180	9614.52
9	[9732.18,4248.79,4891.70,520.57,54.96,79.49,981.29,779.40]	[9510.30, 4028.08, 4670.99, 321.49, 10.00, 629.84, 936.33, 591.38]	0.164	18209.38
10	[2861.58, 6697.10 5237.575 , 761.36 863.54, 962.159 , 429.385 ,122.319]	[2765.97, 6573.42, 5065.34, 22.49, 10.00, 684.68, 579.16 331.13]	0.1017	14404.74

b) Selection of intial point

Because of the multi-dimensionalty of this problem the intial point close to the minimum point was given for some value. Depending on the starting point of the search algorithm the problem can go to infinte loop. This is due to fact that the objective function is not dependent on many vaiables. The gradient of few vaiable can be very high when compared to the other variable. The search direction will give minimum point at the point outside the bounds or there will not be any change in the obtained solution. To avoid this gradients were normalized above certain value and very small value were neglected but still this problem will araise for some value of initial search point.

c) Result obtained

The algorithm was run for 10 times and the result of the same is tabulated in Table II. The convergence plot for the best-obtained solution is plotted in Fig. 8. In Table II the first six iteration was run using the normalized constraints function. The other four iteration was solved without normalizing. It was observed that the constraints violation was more in case of normalized function when compared to that solved without normalizing function. But there is not much variation in the minimum value of the function. Even when the

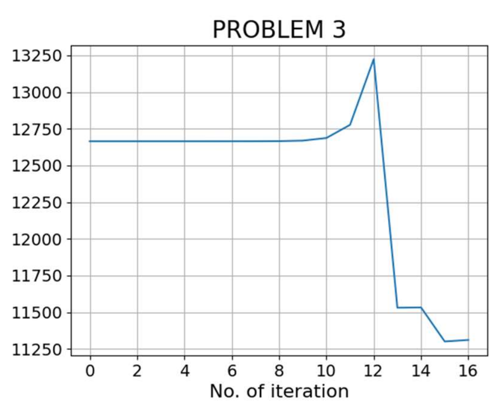

Fig. 8. CONVERGENCE PLOT FOR PROBLEM #3

TABLE VII. COMPARISON OF THE RESULT

Statistics of Result		
Best	x	[1147.79,6298.45,2168.26,10,147.1,757.55,267.32,829.45]
	f(x)	9614.52
MATLAB	x	*[100, 1000, 2552.1, 10, 10, 390, 314.2, 490]*
	f(x)	3652.1
Worst	f(x)	18209.38
Mean	f(x)	14669.45
Median	f(x)	15833.577
Standard Deviation	f(x)	3899.900

IV. CONCLUSION

The bracket operator penalty method can be used to solve any constrained problem. It can also be observed that the parameter such penalty parameter and rate of increase in penalty parameter with each iteration play an important role in obtaining the optimum point. These parameters vary from problem to problem. Proper selection of these parameters is important part of solving a problem using penalty. Further study can be done by changing the termination parameter of steepest descent based on penalty parameter, or exploring option for termination condition for penalty function. This may lead an algorithm that is more generic.

REFERENCES

[1] Kalyanmoy Deb, Optimization for Engineering Design, PHI Learning Private Limited, II edition, 2012, pp. 49, 65, 112, 156

[2] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke & Travis E. Oliphant. Array programming with NumPy, Nature, 585, 357–362 (2020), DOI:10.1038/s41586-020-2649-2

APPENDIX

A. Constraint optimixation Proble no.1

The mathematical equation of the first test problem is

$$\min f(x) = (x_1 - 10)^3 + (x_2 - 20)^3,$$
$$subject\ to$$
$$g_1(x) = (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0,$$
$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0,$$
$$13 \leq x_1 \leq 20, \qquad 0 \leq x_2 \leq 4$$

The bracket operator penalty function for the above problem is

$$\begin{aligned}\min F(x) = {} & (x_1 - 10)^3 + (x_2 - 20)^3 \\ & + R(\langle (x_1 - 5)^2 + (x_2 - 5)^2 - 100\rangle^2 \\ & + \langle -(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81\rangle^2) \\ & + \langle x_1 - 13\rangle^2 + \langle -x_1 + 20\rangle^2 + \langle x_2\rangle^2 + \langle -x_2 + 4\rangle^2)\end{aligned}$$

B. Constraint Optimization Problem no. 2

The mathematical equation of the second test problem is

$$\max f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)},$$
$$subject\ to$$
$$g_1(x) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$
$$0 \leq x_1 \leq 10, \qquad 0 \leq x_2 \leq 10$$

The bracket operator penalty function for the above problem is

$$\begin{aligned}\min F(x) = {} & -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \\ & + R(\langle -x_1^2 + x_2 - 1\rangle^2 + \langle -1 + x_1 - (x_2 - 4)^2\rangle^2 \\ & + \langle x_1\rangle^2 + \langle -x_1 + 10\rangle^2 + \langle x_2\rangle^2 + \langle -x_2 + 10\rangle^2)\end{aligned}$$

C. Constrint optimization problem no. 3

The mathematical equation of the third test problem is

$$\min f(x) = x_1 + x_2 + x_3$$
$$subject\ to$$
$$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0,$$
$$g_2(x) = -1 + 0.0025(-x_4 + x_5 + x_7) \leq 0,$$
$$g_3(x) = -1 + 0.01(x_6 + x_8) \leq 0,$$
$$g_4(x) = 100x_1 - x_1x_6 + 83333.333 \leq 0,$$
$$g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0,$$
$$g_6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0,$$
$$100 \leq x_1 \leq 10000, \qquad 1000 \leq x_i \leq 10000, i = 2,3$$
$$10 \leq x_i \leq 1000, i = 4,5, \dots ,8$$

The Bracket operator penalty function for the above problem is

$$\begin{aligned}\min F(x) = {} & x_1 + x_2 + x_3 \\ & + R(\langle 1 - 0.0025(x_4 + x_6)\rangle^2 \\ & + \langle 1 - 0.0025(-x_4 + x_5 + x_7)\rangle^2 \\ & + \langle 1 - 0.01(-x_6 + x_8)\rangle^2 \\ & + \langle -0.0012x_1 + 1.2 \times 10^{-5}x_1x_6 - 0.009999x_4 \\ & + 1\rangle^2 + \langle -0.0008x_2x_4 + 0.0008x_2x_7 + x_4 - x_5\rangle^2 \\ & + \langle -0.0000008x_3x_5 + 0.0000008x_3x_8 + 0.002x_5 \\ & + 1\rangle^2 + \langle x_1 - 100\rangle^2 + \langle 10000 - x_1\rangle^2 \\ & + \langle 10000 - x_2\rangle^2 + \langle x_2 - 1000\rangle^2 + \langle 10000 - x_3\rangle^2 \\ & + \langle x_3 - 1000\rangle^2 + \langle 1000 - x_4\rangle^2 \\ & + \langle x_4 - 10\rangle^2 + \langle 1000 - x_5\rangle^2 \\ & + \langle x_5 - 10\rangle^2 + \langle 1000 - x_6\rangle^2 \\ & + \langle x_6 - 10\rangle^2 + \langle 1000 - x_7\rangle^2 \\ & + \langle x_7 - 10\rangle^2 + \langle 1000 - x_8\rangle^2 + \langle x_8 - 10\rangle^2)\end{aligned}$$