Constrained Optimization Methods

Anupam Khandelwal
Dept. of Electronics & Electrical Engineering
Indian Institute of Technology, Guwahati
Guwahati, India
anupam1997@iitg.ac.in

Konark Jain
Dept of Electronics & Electrical Engineering
Indian Institute of Technology, Guwahati
Guwahati, India
konark.145@iitg.ac.in

***Abstract*—Optimization presents itself as a challenging task in many real world applications and so does comes the need of numerical techniques to solve such problems.**
This report presents a comparative study of two such algorithms namely, Penalty Function Method and Method of Multipliers that are widely used for solving constrained optimization problems. Both the algorithms converts a constrained optimization problem into an unconstrained one which can be solved using any of the multivariable optimization techniques available such as Cauchy's Steepest Descent Method, etc.

***Index Terms*—optimization, Steepest Descent Method**

I. INTRODUCTION

Optimization techniques are developed for tackling a multitude of problems. Ranging from material sciene, physics, mechanical, aerospace, aeronautical engineering to the most recent advancements in artificial intelligence and machine learning, optimization techniques have been applied to problems in all domains of scientific research.

Optimization is defined as the process of finding the optimum value (usually on a global scale) i.e. the maxima or the minima of a mathematical function which might be explicitly stated or may come from an implicit simulation. Optimization is an extremely active area of research and advances in the field have been increasing day by day.

Optimization can be characterised into three subcategories: single-variable optimization, multivariable optimization and finally constrained optimisation. We employ the Penalty Function Method and the Method of Multipliers to tackle the problem of constrained optimization for the final project of the ME609 course taught in IIT Guwahati.

The problem of constrained optimization is formulated as follows:

$$
\begin{aligned}
&\text{Minimize:} \quad f(x) \\
&\text{Subject to:} \\
&\qquad g_j(x) \geq 0 \quad j = 1 \dots J \\
&\qquad h_k(x) = 0 \quad k = 1 \dots K \\
&\qquad x^{(L)} \leq x \leq x^{(U)} \quad \text{where } x \text{ is a vector}
\end{aligned}
$$

$g_j(x)$ are the inequality constraints and $h_k(x)$ are the equality constraints for $j = 1 \dots J$ and $k = 1 \dots K$ while $x^{(L)}$ and $x^{(U)}$ are the lower bounds and upper bounds for the vector x.

II. BRIEF OVERVIEW OF ALGORITHMS

We now describe the algorithms and their mathematical formulation to solve the problem of constrained optimization as formulated above.

A. Penalty Function Method

Penalty function method involves converting a constrained optimization problem into an unconstrained one by introducing a penalty term depending upon the severity of each constraint violation and instead of minimizing the objective function directly, a penalty function of combined objective function and penalty terms is minimized.

The penalty function *P*, is defined as:

$$P(x, R) = f(x) + \Omega(R, g(x), h(x)) \tag{1}$$

where *f(x)* is the objective function, *g(x)* and *h(x)* are the inequality and equality constraints respectively, *R* is a set of penalty parameters, and Ω being the penalty term.
Depending upon the the type of constraints and feasibility requirements, various penalty operators are used, but this report will only be considering bracket operator penalty.

$$\Omega = R < g(x) >^2 \tag{2}$$

where $< \alpha >= \alpha$ when $\alpha < 0$; zero, otherwise.

Algorithm 1 Penalty Function Method.

```
Initialize : ε1, ε2, x(0), R(0), c, t = 0
while |x(t+1) − x(t)| ≥ ε1 do
  P(x(t), R(t)) = f(x(t)) + Ω(R(t), g(x(t)), h(x(t)))
  x(t+1) ← steepest_descent(x(t), P)
  if |P(x(t+1), R(t)) − P(x(t+1), R(t))| ≤ ε2 then
    xT ← x(t+1)
    TERMINATE
  else
    R(t+1) ← cR(t)
    t ← t + 1
  end if
end while
return xT
```

B. Method of Multipliers

One of the fundamental challenges with Penalty Function Method is that the contours of penalty function with respect to the original objective function are highly distorted which in turn results in formation of artificial minimas.
One such way to overcome this problem of distortion is to use a fixed penalty parameter *R* with a multiplier corresponding to each constraint. The constraint violation is increased by the multiplier value before calculating the penalty term. Thereafter, an equivalent term is subtracted from the penalty term.

The penalty function is thus modified as follows:

$$P(x,\sigma^{(\mathrm{t})},\tau^{(\mathrm{t})}) = f(x) + R\sum_{j=1}^{J}[(< g_j(x)+{\sigma_j}^{(\mathrm{t})} >)^2 - ({\sigma_j}^{(\mathrm{t})})^2] + R\sum_{k=1}^{K}[(h_k(x)+{\tau_k}^{(\mathrm{t})})^2 - ({\tau_k}^{(\mathrm{t})})^2] \quad (3)$$

where σ_j and τ_j are updated iteratively as:

$$\begin{aligned} {\sigma_j}^{(\mathrm{t+1})} &=< g_j(x^{(\mathrm{t})}) + {\sigma_j}^{(\mathrm{t})} > \\ {\tau_k}^{(\mathrm{t+1})} &= h_k(x^{(\mathrm{t})}) + {\tau_k}^{(\mathrm{t})} \end{aligned}$$

The above formulation significantly reduces the distortion in the penalty function thereby minimizing the chances of creating artificial minimas.

Algorithm 2 Method of Multipliers.

Initialize : $\epsilon_1, \epsilon_2, x^{(0)}, R^{(0)}, c, t=0, {\sigma_j}^{0}=0, {\tau_k}^{(0)}=0$
while $|x^{(\mathrm{t+1})} - x^{(\mathrm{t})}| \geq \epsilon_1$ **do**

$$\begin{aligned} P(x,\sigma^{(\mathrm{t})},\tau^{(\mathrm{t})}) = R\textstyle\sum_{j=1}^{J}&[(< g_j(x)+{\sigma_j}^{(\mathrm{t})} >)^2 - ({\sigma_j}^{(\mathrm{t})})^2] \\ + R\textstyle\sum_{k=1}^{K}&[(h_k(x)+{\tau_k}^{(\mathrm{t})})^2 - ({\tau_k}^{(\mathrm{t})})^2] \\ + f(x^{(\mathrm{t})})& \end{aligned}$$

$x^{(\mathrm{t+1})} \leftarrow steepest\_descent(x^{(\mathrm{t})}, P)$

if $|P(x^{(\mathrm{t+1}),\ \mathrm{R}^{(\mathrm{t})}}) - P(x^{(\mathrm{t+1}),\ \mathrm{R}^{(\mathrm{t})}})| \leq \epsilon_2$ **then**
$x^{\mathrm{T}} \leftarrow x^{(\mathrm{t+1})}$
TERMINATE
else
${\sigma_j}^{(\mathrm{t+1})} \leftarrow < g_j(x^{\mathrm{t}}) + {\sigma_j}^{(\mathrm{t})} > \quad \forall \quad j = 1,2,3..,J$
${\tau_k}^{(\mathrm{t+1})} \leftarrow h_k(x^{\mathrm{t}}) + {\tau_k}^{(\mathrm{t})} \quad \forall \quad k = 1,2,3..,K$
$t \leftarrow t+1$
end if
end while
return x^{T}

Both the methods require an unconstrained optimization method namely, *steepest_descent()* to compute subsequent search points, which is discussed next.

C. Cauchy's Steepest Descent Method

Cauchy's descent method uses negative of the gradient value at a particular point as a search direction to find the subsequent search points.

$$s^{(\mathrm{k})} = -\nabla f(x^{(\mathrm{k})}) \quad (4)$$

$$x^{(\mathrm{k+1})} = x^{(\mathrm{k})} + \alpha^{(\mathrm{k})} s^{(\mathrm{k})} \quad (5)$$

At every iteration, the derivative is computed at the current point and a unidirectional search is performed in the negative to this derivative direction to find the minimum point along that direction. The minimum point becomes the current point and the search is continued from this point. The algorithm continues until a point having a small enough gradient vector is found.

III. Results and Discussions

A. Implementation details

We use the MATLAB programming language's 2019b edition to develop codes for the problems as described in the Appendix. The code was run on a computer with the following specifications: Intel Core i5-6200U CPU @2.30 GHz and with 8GB RAM with a 64-bit operating system (Windows 10).

The implementation is organised in the following way:

1) **exhaustive_search.m :** This executes the exhaustive search method as a bracketing method which gets called in the steepest descent function.
2) **partial_derivative.m :** Calculates the partial derivative of the function passed into it as a function handler.
3) **steepest_descent.m :** Steepest descent algorithm of multi variable optimization is defined here. It has in itself the code for Golden Section Search Method for single variable optimization which calls the exhaustive search function for bracketing of α.
4) **bracket_operator.m :** employs the penalty function method with penalty function as the bracketing operator. It calls the steepest descent function for getting solution of each penalty function formed by it.
5) **method_multiplier.m :** employs the method of multiplier method and calls the following two functions for it:
   a) **evaluator.m :** evaluates the penalty function using the σ as well as the constraints defined for each function.
   b) **bracketer.m :** an implementation of the bracket operator for calling in the evaluator.
6) **phase3_main.m :** Main function takes in the parameters from an excel sheet and runs the two algorithms for the three problems one by one printing out the optima found along with the function value calculated at the optima.

The number of iterations for both methods is set to be 20. Initial R is set to be 0.1 and for Penalty function method, we increment R by a factor of 10 in every iteration. In Cauchy's steepest descent method we use maximum iterations as 5000 and $\epsilon = 10^{-6}$. We employ exhaustive search followed by golden section search method for univariate optimization within the steepest descent method. Partial derivatives are calculated by taking $\delta = 10^{-8}$.

B. Drawbacks of Penalty Function Method

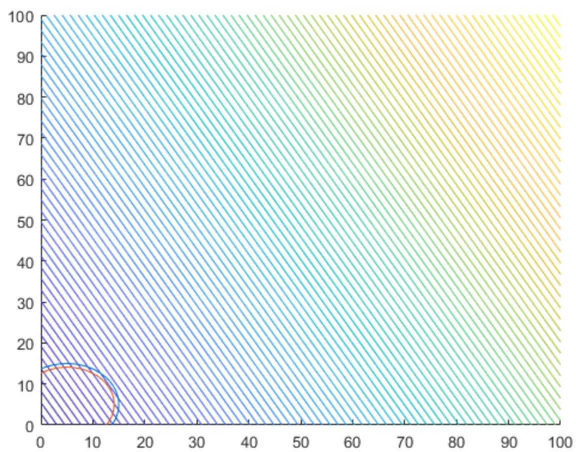

Fig. 1. P1: Objective function contour.

For problem P1, contour of the original objective function is as shown in Fig. 1. Whereas the contour plot for penalty function obtained using Penalty Function Method is as shown in Fig. 2.

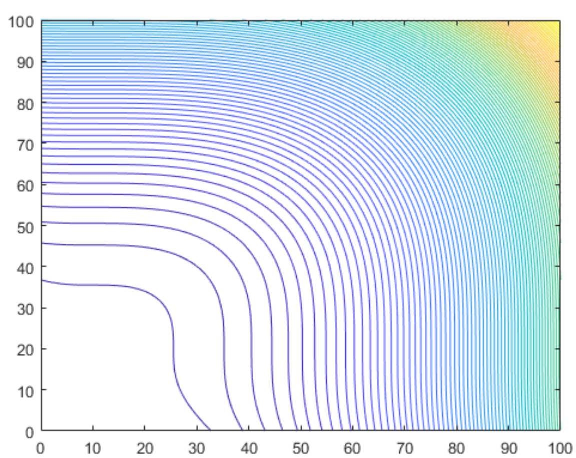

Fig. 2. P1: Penalty function contour.

It is clearly visible that by combining penalty terms with the original objective function distorts the contour plot of minimizing function which in turn can creates artificial minima points and thereby limiting the solving capability of the algorithm.

C. Varying the values of epsilon

Solution: $x^*=(1.2279, 4.2453)^T$ & $f(x^*)=0.09582$ for P2.

We use the second problem for performing an analysis over the various epsilon value and its effect on the convergence of the algorithms. As shown in Table II, the reducing epsilon value leads to slight improvement in the function value and the solution is also closer to the optima.

D. Comparing the two methods with ten different starting points

As can be seen in Table I, we start each algorithm for each problem with ten different initial points: five in feasible region and five in the infeasible region.

We observe that firstly, penalty function method while gives us better mean function values, the standard deviation is extremely high when compared to Method of Multipliers. Therefore we can conclude that the latter method is more stable and produces less premature convergence. This is primarily due to the fact that penalty function distorts the objective function much more than the method of multiplier does. Note that the method of multiplier has remarkable stability for any starting point in the problem 2 with zero standard deviation.

Secondly, we observe that the best values for all penalty function problems is remarkably close to the actual optima as described in the Appendix. This implies that penalty function, when started with the right initial point, can indeed perform better than method of multipliers. On the other hand, we see that the worst value is much worse than those of method of multipliers as well.

Thirdly, problem 2 is the problem which performs the best in each of the two algorithms with maxima achieved remarkably close to the global optima. Problem 3 poses a high dimensional problem with extremely complex feasible search space due to the presence of 6 inequality constraints as well as 8 variable bounds. Therefore, not surprisingly, we have subpar performance in the 3rd problem by both of these algorithms.

IV. Conclusion

After employing both of the algorithms, following conclusions were made. Firstly, penalty function method performs much better than the method of multipliers if the initial guess is close to the true optima. Secondly, due to distortion of penalty function in penalty function method, artificial minimas are more likely to occur which can result in convergence to false minimas, so random initialization of search point may not always lead to accurate convergence. And finally, it can be concluded that in general, method of multipliers is a computationally less expensive algorithm compared to the penalty function method.

References

[1] K. Deb, " Optimization for Engineering Design Algorithms and Example" 1st ed., vol. 9., Prentice-Hall of India Limited, 1995.

TABLE I
ALGORITHMS RUNNING FOR 10 DIFFERENT START POINTS

Algorithm	Problem No.	**Final Function values**				
		Mean	**Best**	**Worst**	**Median**	**Std**
Penalty	1	-1.20E+04	-1.41E+04	27.0095	-1.41E+04	4577.544437
Penalty	2	-3.46E-02	-9.58E-02	-2.32E-02	-8.43E-02	0.03953
Penalty	3	2.12E+03	1.68E+03	3.00E+03	1.93E+03	448.72
Multipliers	1	-2.20E+04	-4.46E+04	0.0999	-2.41E+04	500.45
Multipliers	2	-3.20E-02	-9.58E-02	-2.91E-02	-9.13E-02	0.04358
Multipliers	3	2.67E+03	2.40E+03	3.14E+03	2.63E+03	216.40

TABLE II
PENALTY FUNCTION METHOD FOR DIFFERENT EPSILON VALUES FOR P2.

ϵ	**Optima value**	**Function evaluation at optima**
10^{-1}	$(1.2260, 4.2733)^T$	0.0943
10^{-2}	$(1.2293, 4.2424)^T$	0.0958
10^{-3}	$(1.2280, 4.2456)^T$	0.0958
10^{-4}	$(1.2280, 4.2454)^T$	0.0958

V. APPENDIX

The problems given to solve using the Penalty Function Method and the Method of Multipliers are:

A. Problem 1:

It is formulated as follows:

$$\min\ f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to:

$$g_1(x) = (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0.$$
$$g_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 - 82.81 \geq 0.$$
$$13 \leq x_1 \leq 100; \quad 0 \leq x_2 \leq 100.$$

The number of variables are 2 and the global minima as specified by the question paper is $x^* = (14.095, 0.84296), f(x^*) = -6961.81388$. The plot of the function without constraints for the given range is as shown in Fig. 3. The feasible space is shown in Fig. 1 in the contour plot of our objective function. The blue line corresponds to $g_1(x)$ and red corresponds to $g_2(x)$. The feasible space is the region expanding from the circles outwards.

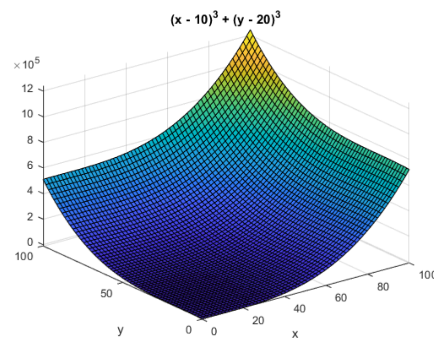

Fig. 3. Problem 1

B. Problem 2:

It is formulated as follows:

$$\min\ f(x) = \frac{sin^3(2\pi x_1) sin(2\pi x_2)}{x_1^3 (x_1 + x_2)}$$

subject to:

$$g_1(x) = x_1^2 - x_2 + 1 \leq 0.$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0.$$
$$0 \leq x_1 \leq 10; \quad 0 \leq x_2 \leq 10.$$

The number of variables are 2 and the global minima as specified by the question paper is $x^* = (1.2279713, 4.2453733), f(x^*) = 0.095825$. The plot of the function without constraints for the given range is as shown in Fig. 4. The feasible space is shown in Fig. 5 in the contour plot of our objective function. The green line corresponds to $g_1(x)$ and red corresponds to $g_2(x)$. The feasible space is the region enclosed by them.

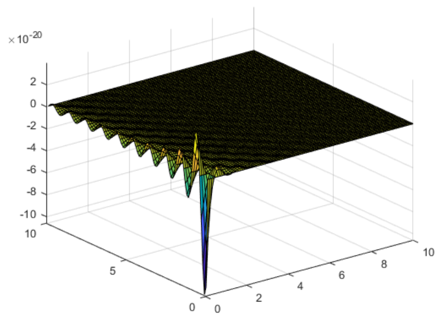

Fig. 4. Problem 2

C. *Problem 3:*

It is formulated as follows:

$$\min\ f(x) = x_1 + x_2 + x_3$$

subject to:

$$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0.$$
$$g_2(x) = -1 + 0.0025(-x_4x_5 + x_7) \leq 0.$$
$$g_3(x) = -1 + 0.01(-x_6 + x_8) \leq 0.$$
$$g_4(x) = 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0.$$
$$g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0.$$
$$g_6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0.$$
$$100 \leq x_1 \leq 10000$$
$$1000 \leq x_i \leq 10000, i = 2, 3$$
$$10 \leq x_i \leq 1000, i = 4, 5, \ldots, 8.$$

The number of variables are 8 and the global minima as specified by the question paper is $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979), f(x^*) = 7049.3307$. The plot of the function could not be built for the dimension of the problem is not less than three.

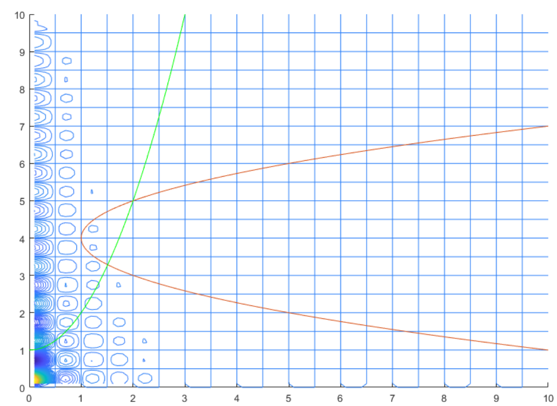

Fig. 5. Problem 2: Contour