Refashioning of Unconstrained Optimization techniques for Constrained Optimization using Variable Metric Method.

Aakash Agrawal
Department of Chemistry
Indian Institute of Technology Guwahati,
Guwahati, India, 781039
Email: akash2016@iitg.ac.in

***Abstract*—Nonlinear constrained optimization problems are an important class of problems with a broad range of engineering, scientific, and operational applications. In this paper we present an algorithm which incorporates Penalty function method, method of multipliers, Davidon-Fletcher-Powell (DFP) method, Bounding phase method and Secant method for nonlinear constrained and unconstrained optimization problems. Finally, we check the robustness of the algorithm by providing a detailed analysis on three problem sets and monitoring the performance of optima by comparing the results with some of the inbuilt functions in a programming platform.**

Keywords—Constrained-Optimization, optima, Nonlinear programming, DFP method.

I. INTRODUCTION

Almost all engineering design and decision making problems have an objective of optimizing a function and simultaneously have a requirement for satisfying some constraints arising due to space, strength, or stability considerations. In Mathematical optimization, constrained optimization is the process of optimizing an objective function with respect to some variables in the presence of constraint of those variables.

A constrained optimization problem with *N* decision/design variables is given by:

$$\begin{aligned} &\text{Minimize} \quad f(x) \\ &\text{Subject to} \quad g_j(x) \geq 0, \ j = 1, ..., J; \\ &\qquad\qquad h_k(x) = 0, \ k = 1, ..., K; \\ &\qquad\qquad x_i^{(L)} \leq x_i < x_i^{(U)}, \ i = 1, ..., N; \end{aligned} \tag{1}$$

where $g_j(x)$ are the J inequality constraints and $h_k(x)$ are the K equality constraints. $f(x)$ is the objective function to be optimized.

In this report, we present results based on implementation of two popular constrained optimization techniques namely, penalty function method and the method of multipliers. DFP method is used for multi-variable optimization and a combination of the bounding phase and the Secant method is used to achieve a unidirectional search.

II. THEORY

A) Penalty function method

Penalty function methods approximate a constrained problem by an unconstrained problem structured such that minimization favors satisfaction of the constraints. We achieve so by introducing an artificial penalty for violating the constraint. The general technique is to add to the objective function a term that produces a high cost for violation of constraints. At any sequence, the following penalty function is minimized:

$$P(x, R) = f(x) + \Omega(R, g(x), h(x)) \tag{2}$$

where, *R* is a penalty parameter, Ω is the penalty term, and *P(x,R)* is the penalty function. There are various penalty operators depending upon the feasibility and type of constraint. This report will be considering only the bracket operator penalty.

$$\Omega = R < g(x) >^2 \tag{3}$$

B) Method of Multipliers

To eliminate the problem of artificial local optima that arises from a penalty function merhod, we use the method of multipliers. The constraint violation is increased by the multiplier value before calculating the penalty term. Thereafter, an equivalent term is subtracted from the penalty term. This method works in successive sequences, each time updating the multipliers in a prescribed manner. The penalty function is modified as:

$$\begin{aligned} P(x, \sigma^{(t)}, \tau^{(t)}) = f(x) &+ R \sum_{j=1}^{J} [(< g_j(x) > + \sigma_j^{(t)})^2 - (\sigma_j^{(t)})^2] \\ &+ R \sum_{k=1}^{K} [(< h_k(x) > + \tau_k^{(t)})^2 - (\tau_k^{(t)})^2] \end{aligned} \tag{4}$$

In method of multipliers the value of R is kept constant throughout and the value of σ_j and τ_k are updated successively as:

$$\begin{aligned} \sigma_j^{(t+1)} &= < g_j(x^{(t)}) > + (\sigma_j^{(t)}) \\ \tau_k^{(t+1)} &= < h_k(x^{(t)}) > + (\tau_k^{(t)}) \end{aligned} \tag{5}$$

The method of multiplier and penalty function method both will convert a constrained optimization problem to a unconstrained problem which further can be solved by DFP method.

C) Variable-metric method (DFP method)

DFP method is a multi-variable optimization algorithm, typically used for a faster convergence to the optima. It eliminates the limitations of several other algorithms by not taking into account the hessian for creating a search direction. Instead, it estimates the inverse of the hessian

matrix iteratively using first order derivatives. The search direction is given by:

$$s^{(k)} = -A^{(k)} \nabla f(x^{(k)}) \quad (6)$$

where the matrix *A* is given by:

$$A^{(k)} = A^{(k-1)} + \frac{\Delta x^{(k-1)} \Delta x^{(k-1)^T}}{\Delta x^{(k-1)^T} \Delta e(x^{(k-1)})} - \frac{A^{(k-1)} \Delta e(x^{(k-1)}) ^T}{\Delta e(x^{(k-1)})^T A^{(k-1)} \Delta e(x^{(k-1)})}$$

$$\Delta x^{(k-1)} = x^{(k)} - x^{(k-1)}$$

$$\Delta e(x^{(k-1)}) = e(x^{(k)}) - e(x^{(k-1)}) \quad (7)$$

$e(x^{(k)})$ represents the gradient of the function at a point *x(k)*. The uni-direction search involves secant method and bounding phase method to find the value of alpha in the search space. The new point obtained after uni-direction search is :

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)} \quad (8)$$

After getting a new point, before entering in the loop to evaluate it again a few termination conditions checks have to be made.

III. CODE DESCRIPTION

All the codes are written in C++ programming platform. There are two separate C++ files, one for the method of multipliers and another for the penalty function method. There is an additional file which will be used by the previously mentioned files for reading the data (initial guess and some parameter values). Both these files contain functions and subroutines for the specific methods used in the algorithm.

A) Implementation of algorithm.

A computer code implementing the penalty function method and the method of multipliers is presented here. DFP method is used for multi-variable optimization and a combination of the bounding phase and the golden section search is used to achieve a unidirectional search. We now briefly demonstrate each of the functions and subroutines used:

a) multi_f
This function takes an input vector x (a point in search space) and returns the function value (penalized function value) at that point.

b) grad_multi_f
This function employs the central difference method to compute the gradient vector at a particular point in search space.

c) bracketing_
This function is an implementation of the bounding phase method used to bracket the alphastar (a minima obtained by performing the unidirectional search). It takes a vector x and vector s (search direction) as input and finds an alphastar such that ***f(x + alphastar*s)*** is minimum. The output represents the modified bounds on the basis of which alpha can be evaluated.

d) secant_minima
This function takes the bounds found from the bounding phase method, a point x and a search direction s as input and evaluates the alphastar.

e) compute_z
This function is used to compute the formula used in secant method:

$$z = x_2 - f'(x_2) \frac{x_2 - x_1}{f'(x_2) - f'(x_1)} \quad (9)$$

f) f_dash
This function is used to calculate the first order differential for a single variable function using central difference method. (represents ***f'***).

g) DFP
This function implements the DFP algorithm for the optimization of multi variable function *P(x, R), P(x, σ, τ) {equation 2, 4}.*This function is called inside the *main* function for some number of sequences until the termination condition is met. It takes only the input vector *x* as the argument and returns a new solution vector.

It starts by first finding a search direction from the input vector *x.* Then it performs a unidirectional search by calling the bounding phase and the secant method to find the optimum alphastar. It then goes on to find new search directions by computing the equations in *(6)* and *(7).* The process continues until the termination condition for DFP is met and we have found the optimum solution of this unconstrained problem for this particular sequence.

h) main
This function implements the penalty function method and the method of multipliers. The algorithm is generalized and asks the user to input the number of variables he wants to work with. The algorithm starts in a loop that runs for a specific user input number of sequences, until any termination condition is met. The value of the penalty parameter *R* is updated during this process and new solutions close to optimum are generated using the DFP method.

Normalization – In the presence of multiple constraints the penalty function can be constructed using normalizing all constraints. Normalization came out to be quite useful in our analysis. The constrained equations are written in the form:

$$1 - \frac{g_j(x)}{b_j} \geq 0 \quad (10)$$

Restart – We have an option of restart in the function DFP. 'Restart' is a global variable in our case and the user has to initially specify 0 (restart) or 1(meaning no restart). We restart the unconstrained optimization after the very 4th iteration in a given sequence.

We now present a flowchart describing the flow of our algorithm.

FLOW CHART OF THE ALGORITHM

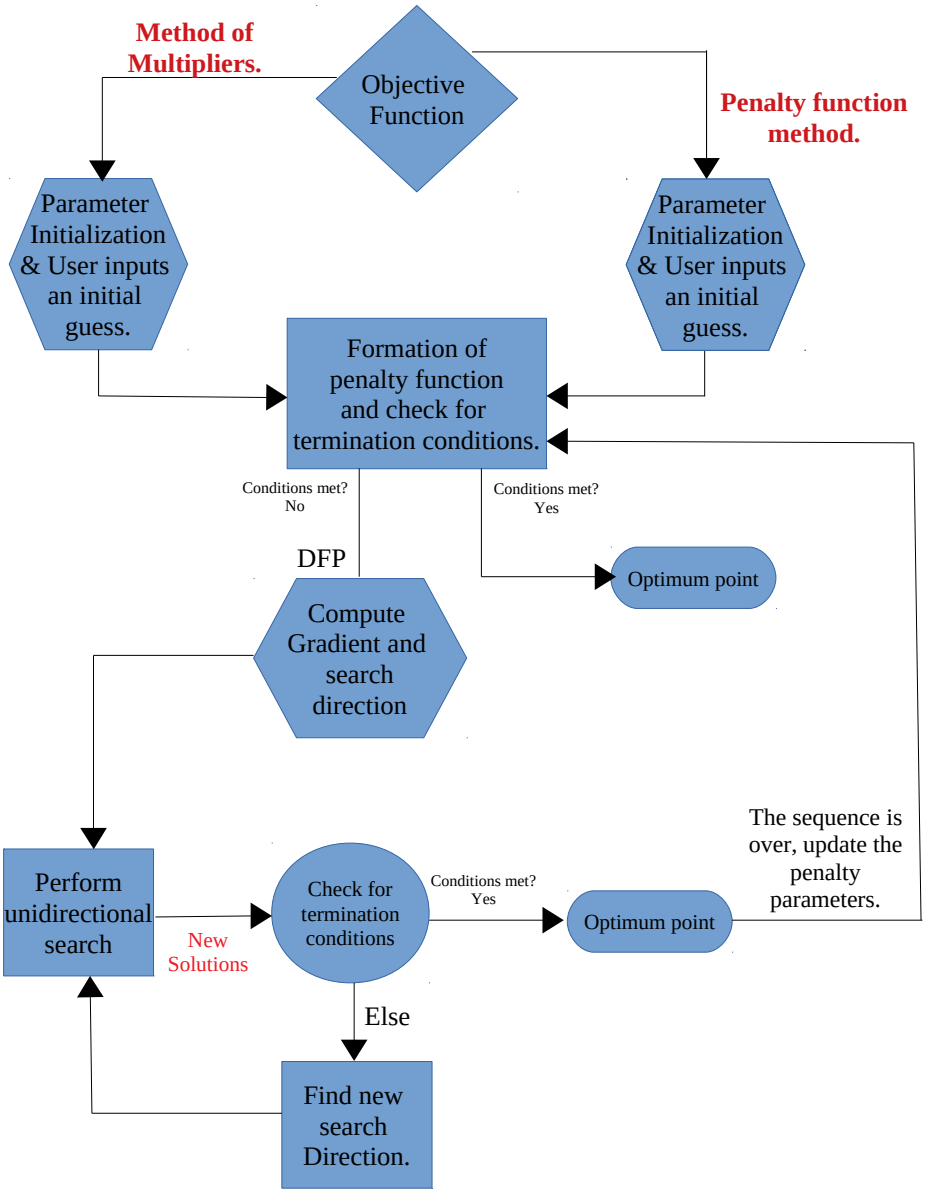

Fig.1: Flow-chart of the Algorithm.

Computational Complexity of the Algorithm.

The computational complexity of the algorithm can be approximated by:

$$O = numseq * (JN + KN + (n_{dfp} + 1)(n_{bracket} + n_{secant}N) + n_{dfp}N)$$

where, ***J*** = number of inequality constraints, ***K*** = number of equality constraints, n_{dfp} = number of iterations in DFP method, $n_{bracket}$ = number of iterations required to bracket the minima in a uni-directional search (Bounding phase method), ***N*** = number of decision variables, n_{secant} = number of iterations used in secant method and ***numseq*** = number of sequential runs in the penalty function method.

IV. RESULTS AND DISCUSSIONS

The algorithm is written on a C++ programming platform. Somestandard libraries like <iostream>, <fstream> (for reading and writing data from a text file), <tuple>, <vector> are used which are inbuilt in any C++ programming platform. Below are the results and a detailed analysis for each question separately. The values corresponding to column "F_global" have been found from the ***optimize*** module of the ***scipy*** library in ***python***.

Hardware used - Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor with 8GB RAM and 2GB graphics card.

A) Himmelblau test case.{refer Appendix}

Method 1: Bracket operator penalty method Parameters setting: R = 0.07, c = 1.55. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001.

Initial Guess	Obtained optima	Number of Sequences	Function value	F_global
3.001 0.115	0.897945 2.93661	9	60.1153	60.372
0 0	0.835535 2.93429	14	60.3343	60.372
0.22 0.9	0.897945 2.93661	9	60.1153	60.372
0.22 2.9	0.897945 2.93661	9	60.1153	60.372
2.62 4.9	0.897945 2.93661	9	60.1153	60.372
Function Value				
Best	Worst	Median	Mean	Std. Dev.
60.3343	60.1153	60.1153	60.1591	0.087

Method 2: Method of Multipliers. Parameters setting: R=0.1. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001.

Initial Guess	Obtained optima	Number of Sequences	Function value	F_global
3.001 0.115	0.828717 2.93394	4	60.4013	60.372
0 0	0.834829 2.94209	8	60.3881	60.372
0.22 0.8	0.832493 2.93855	5	60.3824	60.372
4.25 2.66	0.832961 2.94157	4	60.4165	60.372
4.88 2.66	0.833486 2.94249	4	60.4197	60.372
Function Value				
Best	Worst	Median	Mean	Std. Dev.
60.4197	60.3824	60.4013	60.4015	0.014

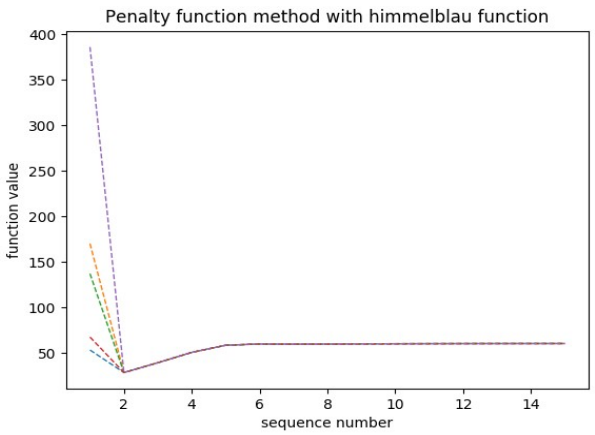

Fig. 2. Function value vs sequence number for Himmelblau using Penalty function method.

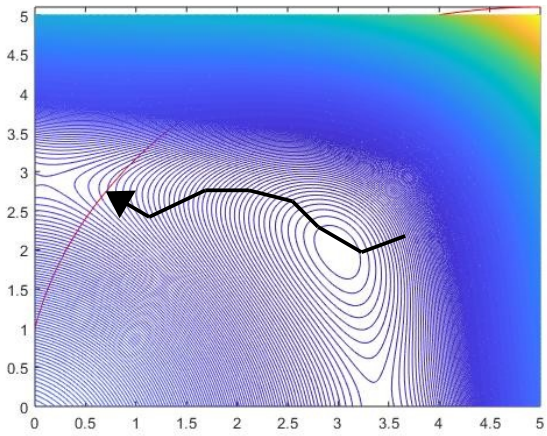

Fig. 3. Contour plot for Himmelblau.

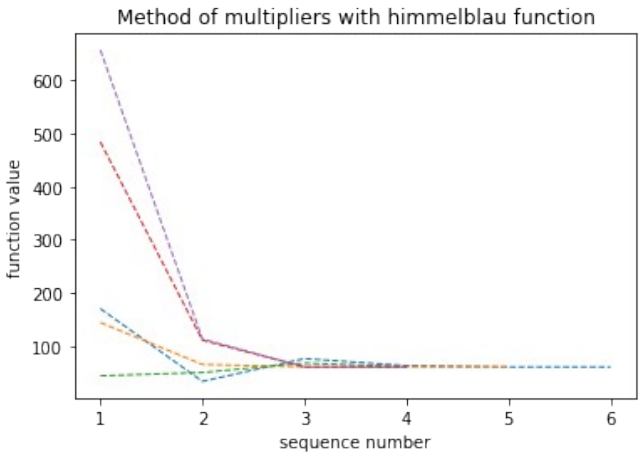

Fig. 4. Function value vs sequence number for Himmelblau using method of multipliers.

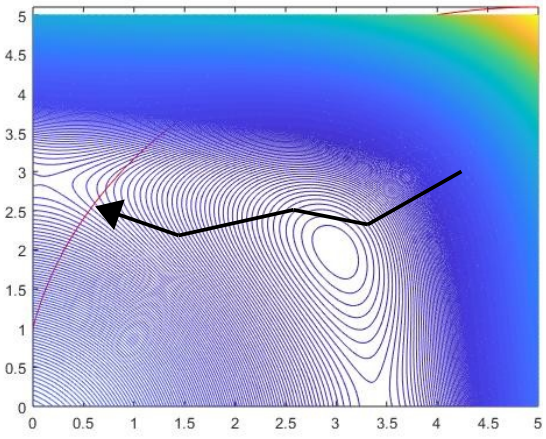

Fig. 5. Contour plot for Himmelblau.

Clearly, we can see the convergence of Method of multiplier is faster than penalty function method, this basically implies the presence of artificial local optima and a **distortions** of the contours in penalty function method. Also, **restart** helps in faster convergence to the minima.

B) ***Problem 1.****{refer Appendix}*

Method 1: Bracket operator penalty method Parameters used: R = 0.05, c = 2. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001.

Initial Guess	Obtained optima	Number of Sequences	Function value	F_global
20 33	13.6401 0.0	6	-7951.38	-7973.0
65 2	13.8511 0.0	18	-7942.89	-7973.0
99 55	13.0084 0.0	3	-7925.55	-7973.0
22.0 22.2	14.8116 0.0	10	-7888.61	-7973.0
52.03 1.66	13.8515 0.0	28	-7942.87	-7973.0
14.055 13.0	13.6433 0.0	6	-7951.36	-7973.0
14 0.3	15.8741 0.0	26	-7797.32	-7973.0
33.89 56.98	13.6521 0.0	13	-7943.19	-7973.0
100 0	13.8515 0.0	27	-7942.87	-7973.0
66 32	13.6435 0.0	5	-7951.5	-7973.0
Function Value				
Best	Worst	Median	Mean	Std. Dev.
-7951.5	-7797.32	-7942.88	-7923.753	41.8189

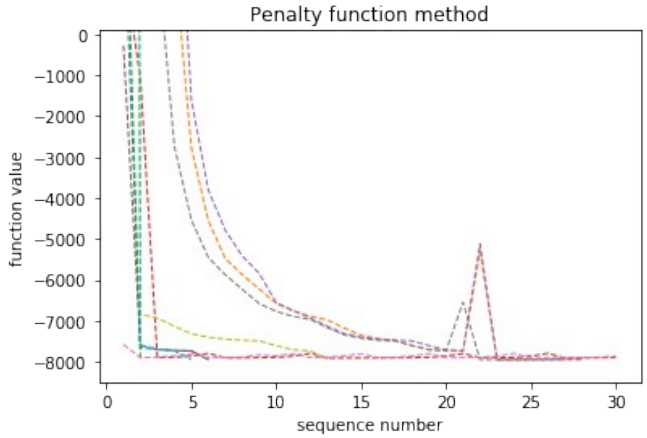

Fig. 6. Function value vs sequence number for problem 1 using penalty function method.

Method 2: Method of Multipliers. Parameters used:R=0.1. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001.

Initial Guess	Obtained optima	Number of Sequences	Function value	F_global
22.22 11.11	13.0946 0.0	3	-7932.77	-7973.0
100 0	13.5994 0.0	21	-7953.57	-7973.0
14.0 0.5	14.5455 0.0	30	-7908.44	-7973.0
30.33 12.05	13.0097 0.0	5	-7960.9	-7973.0
50 50	13.8473 0.0	8	-7943.06	-7973.0
87.099 0.99	13.6039 0.0	21	-7953.41	-7973.0
48.00 15.66	13.0097 0.0	4	-7960.9	-7973.0
99.66 33.99	13.6071 0.0	21	-7953.3	-7973.0
13 10	13.8516 0.0	12	-7942.91	-7973.0
23.32 12.21	13.6604 0.0	8	-7950.23	-7973.0
Function Value				
Best	Worst	Median	Mean	Std. Dev.
-7960.9	-7908.44	-7951.764	-7945949	14.937

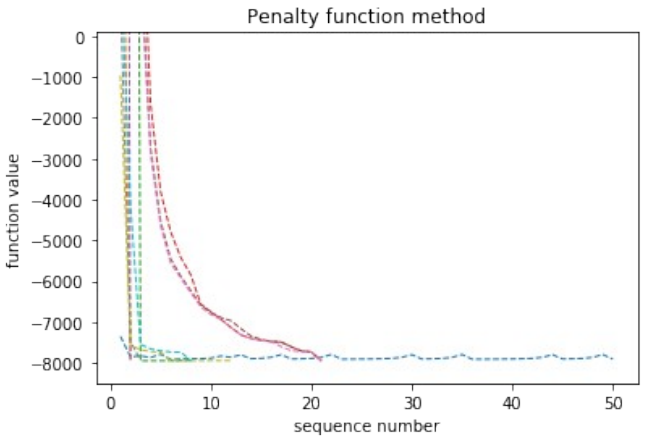

Fig. 7. Function value vs sequence number for problem 1 using method of multipliers.

Here, the DFP method was not even started, at every sequence the unidirectional search converges to the boundary, because of which our algorithm returns to the main function and goes on to the next sequence. Also, in this question we did not take into account the second inequality constraint. Constraint 2 is always active whenever constraint 1 is active.

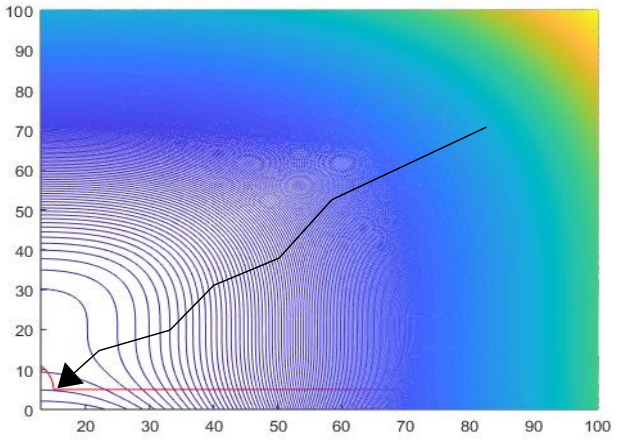

Fig. 8. Contour plot for problem 1.

C) ***Problem 2.****{refer Appendix}*

It is a maximization problem, so we use the Duality principle and convert it to a minimization problem by putting a minus sign.

Method 1: Bracket operator penalty method Parameters used: R = 0.05, c = 10. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001.

Initial Guess	Obtained optima	Number of Sequences	Function value	F_global
1.1 1.1	1.23116 4.2544	7	2.87713	2.87072
6.1 1.5	1.73675 4.75387	8	1.22582	2.87072
6.1 0.1	1.73675 4.75381	4	1.22581	2.87072
5.1 3.1	1.73672 4.75376	4	1.22581	2.87072
0.5 0.5	1.73686 4.75357	5	1.2258	2.87072
1.0 7.5	1.23166 4.25408	10	2.87707	2.87072
2.1 2.9	1.73676 4.7539	7	1.22582	2.87072
1.5 2.06	1.23112 4.25453	8	2.87714	2.87072
1.05 8.95	1.73675 4.75386	4	1.22581	2.87072
1.0001 1.9999	1.73678 4.75388	5	1.22581	2.87072
Function Value				
Best	Worst	Median	Mean	Std. Dev.
2.87714	1.2258	1.22581499	1.721202	0.75672

Our algorithm also gets converged to some local optimum solutions in the vicinity of the true solution both for penalty function method and the method of multipliers.

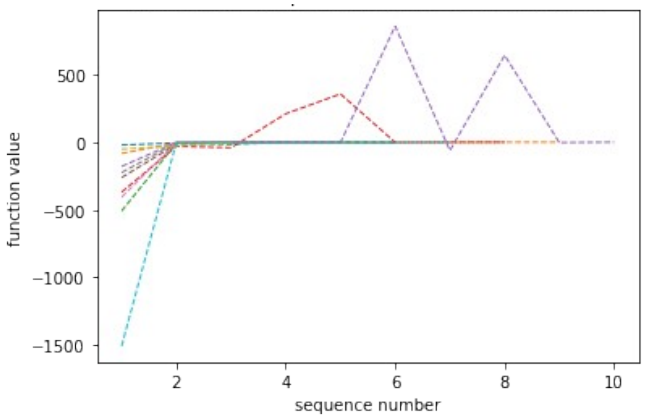

Fig. 9. Function value vs sequence number for problem 3 using penalty function method.

Method 2: Method of Multipliers. Parameters used:R=1.15. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001.

Initial Guess	Obtained optima	Number of Sequences	Function value	F_global
1.0001 1.9999	1.49213 3.2997	6	0.0001667	2.87072
1.1 1.1	1.23113 4.2545	9	2.87714	2.87072
5 5	1.00007 3.99168	6	0	2.87072
4.25 1.47	1.23101 4.25443	8	2.87714	2.87072
0.33 0.66	1.23106 4.25452	10	2.87714	2.87072
3.55 0.78	1.73676 4.75379	4	1.22581	2.87072
4.5 2.5	1.73676 4.7539	5	1.22582	2.87072
0.5 7.66	1.00009 3.99803	6	0	2.87072
2.654 1.999	1.73673 4.75381	3	1.22582	2.87072
6 1	1.73675 4.7539	7	1.22581	2.87072
Function Values				
Best	Worst	Median	Mean	Std. Dev.
2.87714	0.0	1.22582	1.4919	1.1534

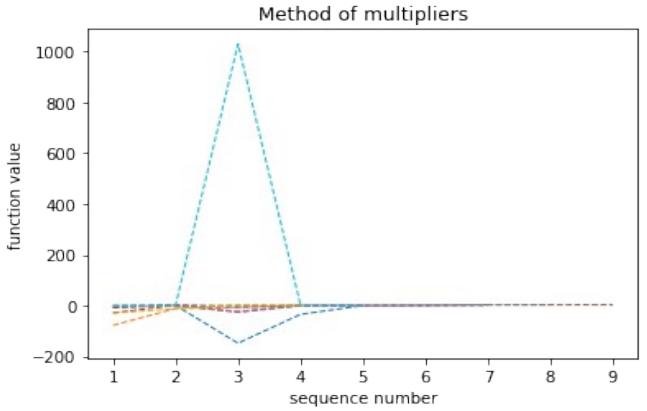

Fig. 10. Function value vs sequence number for problem 1 using method of multipliers.

D) ***Problem 3.****{Refer Appendix}*

Bracket operator penalty method and method of multipliers: Parameters used: R = 0.00000005, c = 10. The initial guess and delta for the Bounding phase method is: ***delta*** = 0.0000001, ***guess*** = 0.5 and the ***terminating factor*** in all cases is 0.0001. There is a problem in the convergence of the algorithm. In is this problem, even a small change in the initial guess or any other parameters like R and c, lead to a drastic change in the function values. So, we try normalization and several values of R and c, but still the algorithm starts with some good initial values but after some sequence, the function values just explode.

CONCLUSIONS

In this report, we present our approch to solving a constrained optimization problem using DFP method for unconstrained optimization and a combination of Secant and bounding phase methods for unidirectional searches. We perform our analysis on a variety of constrained optimization problems and get to know the efficacy of our algorithm. We also came to know the advantage that a “restart” and “normalization” option has in the faster convergence to the optima. The tabulated results in this report, demonstrate the validity of the algorithm to quite a good extent.

APPENDIX: Constraint Optimization Problems.

P: Test Problem.

$$\text{Minimize} \quad (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

subject to

$$(x_1 - 5)^2 + x_2^2 - 26 \geq 0, \qquad x_1, x_2 \geq 0.$$

Fig: Problem Himmelblau

P1: Problem 1.

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

$$\text{subject to } g_1(x) = (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0,$$

$$g_2(x) = 82.81 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0,$$

$$13 \leq x_1 \leq 100, \qquad 0 \leq x_2 \leq 100.$$

- Number of variables: 2 variables.
- The global minima: x* = (14.095,0.84296), f(x*) = -6961.81388.

Fig: Problem 1

P2: Problem 2.

Fig: Problem 2

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

$$\text{subject to } g_1(x) = x_1^2 - x_2 + 1 \leq 0,$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0,$$

$$0 \leq x_1 \leq 10, \qquad 0 \leq x_2 \leq 10$$

- Number of variables: 2 variables.
- The global minima: x∗ = (1.2279713, 4.2453733) , f (x∗) = 0.095825.

P3: Problem 3.

Fig: Problem 3

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

$$\text{subject to } g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0,$$

$$g_2(x) = -1 + 0.0025(-x_4 + x_5 + x_7) \leq 0,$$

$$g_3(x) = -1 + 0.01(-x_6 + x_8) \leq 0,$$

$$g_4(x) = 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0,$$

$$g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0,$$

$$g_6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0,$$

$$100 \leq x_1 \leq 10000$$

$$1000 \leq x_i \leq 10000, i = 2, 3$$

$$10 \leq x_i \leq 1000, i = 4, 5, \ldots, 8$$

- Number of variables: 8 variables.
- The global minima: x∗ = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162,395.5979), f (x∗) = 7049.3307.

REFERENCES

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