Constrained Optimization using Conjugate Gradient method

1st Parmar Harsharajsinh Birendrasinh (150103051)
Department of Mechanical Engineering
IIT Guwahati
Guwahati, India
parmar@iitg.ac.in

2nd Kush Gupta (150103042)
Department of Mechanical Engineering
IIT Guwahati
Guwahati, India
kush.gupta@iitg.ac.in

***Abstract*—Conjugate gradient methods are widely used for unconstrained optimization, especially large scale problems.This paper presents a new version of the conjugate gradient method, which converges conjugate gradient method with bisection method and bounding phase method.The resulting algorithm is more efficient than the conjugate gradient method as it converges faster with less no. of function evaluations.**

***Index Terms*—unconstrained optimization, bisection method, bounding phase method**

I. Introduction

In mathematics, the conjugate gradient method is an algorithm to find the numerical solution of particular systems of linear equations, such as those whose matrix is symmetric and positive-definite. The conjugate gradient method is often implemented as an iterative algorithm, applicable to sparse systems that are too large to be handled by a direct implementation. Large sparse systems often arise when numerically solving partial differential equations or optimization problems.

The conjugate gradient method is widely used to solve unconstrained optimization problems such as energy minimization. It was developed by Magnus Hestenes and Eduard Stiefel in 1951.

Our goal is to develop a modified conjugate gradient method which is more efficient than the original one. The project was divided into three phases. The first phase involved solving the single variable optimization problems using Bisection cum Bounding phase and Secant cum Bounding phase method. In second phase, we tackled unconstrained multi-variable problems using modified Conjugate Gradient algorithm. In phase three, we solved the multi-variable constrained optimization problems using Method of Multiplier (MoM) and Bracket Operator Penalty Function Method.

II. Conjugate Gradient Method

The conjugate gradient method, also known as Fletcher-Reeves method, is similar to the conjugate direction method which uses a history of previous solutions to create new search directions. Assuming that the objective function is quadratic (which is a valid assumption at the vicinity of the minimum for many functions), conjugate search directions can be found using only the first order derivatives. The following conjugate search directions were suggested by Fletcher and Reeves who proved that $s^{(k)}$ is conjugate to all previous search directions $s^{(i)}$ for $i = 1, 2, ..., (k-1)$:

$s^{(k)} = -\nabla f(x^{(k)}) + \frac{\|\nabla f(x^{(k)})\|^2}{\|\nabla f(x^{(k-1)})\|^2} s^{(k-1)}$ with $s^{(0)} = -\nabla f(x^{(0)})$. This recursive equation for search direction $s^{(k)}$ requires only first-order derivatives at two points $x^{(k)}$ and $x^{(k-1)}$. The initial search direction $s^{(0)}$ is assumed to be the steepest descent direction at the initial point. Thereafter, the subsequent search directions are found by using the above recursive equation.

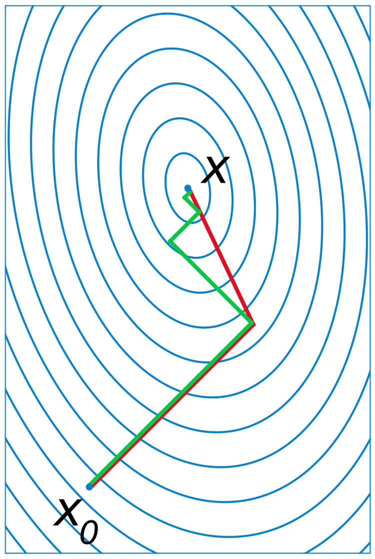


Fig. 1. Conjugate Gradient Operation

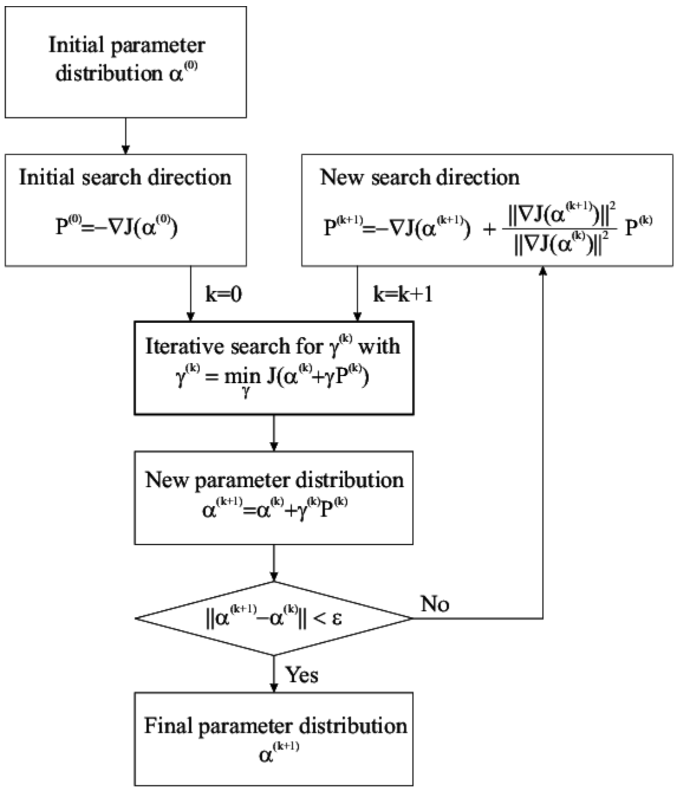
A. Algorithm

- **Step-1 Initialization**: Choose $x^{(0)}$ using bounding phase method. Also, choose the termination parameters $\epsilon_1, \epsilon_2, \epsilon_3$.
- **Step-2 Computation**: Find $\nabla f(x^{(0)})$ and set $s^{(0)} = -\nabla f(x^{(0)})$.
- **Step-3 Uni-directional search**: Find $\lambda^{(0)}$ using bisection search method, such that $f((x^{(0)}) + \lambda^{(0)} s^{(0)})$ is minimum with termination parameter ϵ_1. Set $x^{(1)} = x^{(0)} + \lambda^{(0)^*} s^{(0)}$ and $k = 1$. Calculate $\nabla f(x^{(1)})$.

- **Step-4 Iteration**: Set $s^{(k)} = -\nabla f(x^{(k)}) + \frac{\|\nabla f(x^{(k)})\|^2}{\|\nabla f(x^{(k-1)})\|^2} s^{(k-1)}$
- **Step-5 Uni-directional search**: Find $\lambda^{(k)}$ such that $f((x^{(k)}) + \lambda^{(k)} s^{(k)})$ is minimum with termination parameter ϵ_1. Set $x^{(k+1)} = x^{(k)} + \lambda^{(k)^*} s^{(k)}$.
- **Step-6 Termination**: If $\frac{\|(x^{(k+1)} - x^{(k)})\|}{\|x^{(k)}\|} \leq \epsilon_2$ or $\|\nabla f(x^{(k+1)})\| \leq \epsilon_3$, Terminate.
  Else set $k = k + 1$ and go to Step-4.

For minimization of linear or quadratic objective functions, two iterations of this algorithm are sufficient to find the minimum. But, for other functions, more iterations through Steps 4 to 6 may be necessary. It is observed that the search directions obtained using Equation above become linearly dependent after a few iterations of the above algorithm. When this happens, the search process becomes slow. In order to make the search faster, the linear dependence of the search directions may be checked at every iteration. One way to compute the extent of linear dependence is to calculate the included angle between two consecutive search directions $s^{(k-1)}$ and $s^{(k)}$. If the included angle is close to zero (less than a small predefined threshold angle), the algorithm is restarted from Step 1 with $x^{(0)}$ being the current point. A restart is usually necessary after every N search directions are created.

B. Flow Chart



Initial parameter distribution $\alpha^{(0)}$
Initial search direction
$P^{(0)} = -\nabla J(\alpha^{(0)})$
New search direction
$P^{(k+1)} = -\nabla J(\alpha^{(k+1)}) + \frac{\|\nabla J(\alpha^{(k+1)})\|^2}{\|\nabla J(\alpha^{(k)})\|^2} P^{(k)}$
k=0
k=k+1
Iterative search for $\gamma^{(k)}$ with
$\gamma^{(k)} = \min_{\gamma} J(\alpha^{(k)} + \gamma P^{(k)})$
New parameter distribution
$\alpha^{(k+1)} = \alpha^{(k)} + \gamma^{(k)} P^{(k)}$
$\|\alpha^{(k+1)} - \alpha^{(k)}\| < \varepsilon$
No
Yes
Final parameter distribution
$\alpha^{(k+1)}$

Fig. 2. Conjugate Gradient Method

Here J represents the function.

C. Computational complexity of the algorithm

The dominating operations during an iteration of Conjugate Gradient are matrix-vector products. In general, matrix-vector multiplication requires O(m) operations, where m is the number of non-zero entries in the matrix. The time complexity is slightly more complex, since there are a number of different components that need to happen through each iteration. The components include: calculating the gradient, calculating the beta, finding the new search direction, calculating the step amount, and updating the guess. Calculating the gradient is an O(mn) operation for both the explicitly defined derivative and the induced derivative. When the derivative function is provided explicitly, first the Jacobian matrix must be computed, which is an O(mn) calculation.Conjugate Gradient Method has a time complexity of $O(m\sqrt{k})$.

III. Results and Discussion

A. Details of the Hardware and Programming platform used

The algorithms have been formulated and run on a machine with the following specifications,

- Intel Core i7-7500 2.90 GHz
- 16 GB DDR4 RAM

Programs have been written in FORTRAN F90 and plotting has been done in Octave.

B. Parameters set for algorithm and test problems

The algorithms use inputs from the user on convergence limits and various parameters like Increment coefficient,r, maximum number of sequences and initial point.

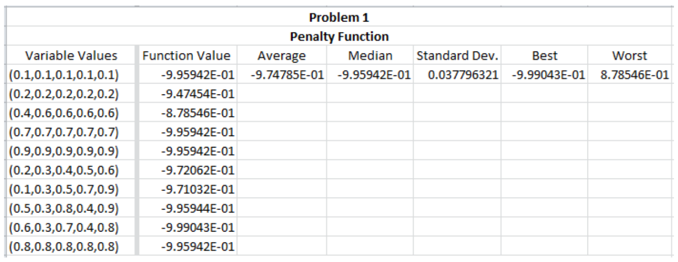
For bracket operator penalty method the following parameters have been used,

- Convergence limits - $1e^{-3}$
- Maximum Sequences - 20
- Increment coefficient (c) - 10
- r - 0.01

For method of multipliers the following parameters have been used,

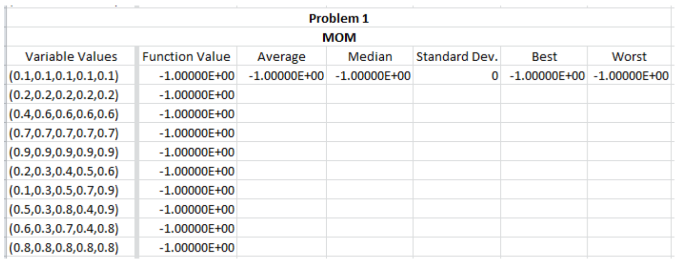
- Convergence limits - $1e^{-3}$
- Maximum Sequences - 20
- r - 10

C. Table of results



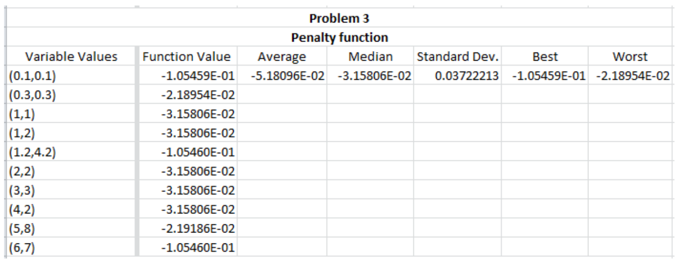
Problem 1						
Penalty Function						
Variable Values	Function Value	Average	Median	Standard Dev.	Best	Worst
(0.1,0.1,0.1,0.1,0.1)	-9.95942E-01	-9.74785E-01	-9.95942E-01	0.037796321	-9.99043E-01	8.78546E-01
(0.2,0.2,0.2,0.2,0.2)	-9.47454E-01					
(0.4,0.6,0.6,0.6,0.6)	-8.78546E-01					
(0.7,0.7,0.7,0.7,0.7)	-9.95942E-01					
(0.9,0.9,0.9,0.9,0.9)	-9.95942E-01					
(0.2,0.3,0.4,0.5,0.6)	-9.72062E-01					
(0.1,0.3,0.5,0.7,0.9)	-9.71032E-01					
(0.5,0.3,0.8,0.4,0.9)	-9.95944E-01					
(0.6,0.3,0.7,0.4,0.8)	-9.99043E-01					
(0.8,0.8,0.8,0.8,0.8)	-9.95942E-01					

Fig. 3. Problem 1 Results using Penalty Function Method



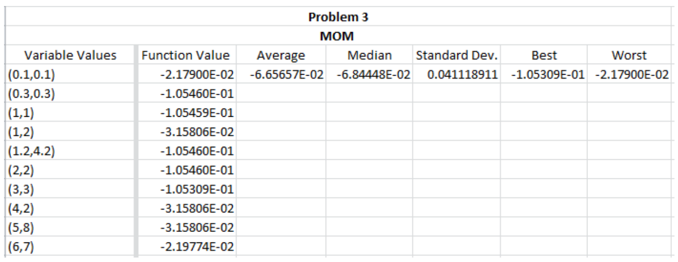
Problem 1						
MOM						
Variable Values	Function Value	Average	Median	Standard Dev.	Best	Worst
(0.1,0.1,0.1,0.1,0.1)	-1.00000E+00	-1.00000E+00	-1.00000E+00	0	-1.00000E+00	-1.00000E+00
(0.2,0.2,0.2,0.2,0.2)	-1.00000E+00					
(0.4,0.6,0.6,0.6,0.6)	-1.00000E+00					
(0.7,0.7,0.7,0.7,0.7)	-1.00000E+00					
(0.9,0.9,0.9,0.9,0.9)	-1.00000E+00					
(0.2,0.3,0.4,0.5,0.6)	-1.00000E+00					
(0.1,0.3,0.5,0.7,0.9)	-1.00000E+00					
(0.5,0.3,0.8,0.4,0.9)	-1.00000E+00					
(0.6,0.3,0.7,0.4,0.8)	-1.00000E+00					
(0.8,0.8,0.8,0.8,0.8)	-1.00000E+00					

Fig. 4. Problem 1 Results using MoM



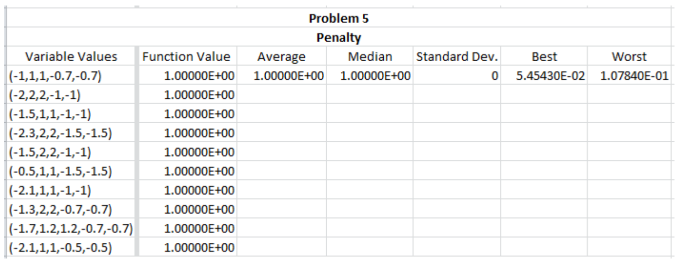
Problem 3						
Penalty function						
Variable Values	Function Value	Average	Median	Standard Dev.	Best	Worst
(0.1,0.1)	-1.05459E-01	-5.18096E-02	-3.15806E-02	0.03722213	-1.05459E-01	-2.18954E-02
(0.3,0.3)	-2.18954E-02					
(1,1)	-3.15806E-02					
(1,2)	-3.15806E-02					
(1.2,4.2)	-1.05460E-01					
(2,2)	-3.15806E-02					
(3,3)	-3.15806E-02					
(4,2)	-3.15806E-02					
(5,8)	-2.19186E-02					
(6,7)	-1.05460E-01					

Fig. 5. Problem 3 Results using Penalty Function Method



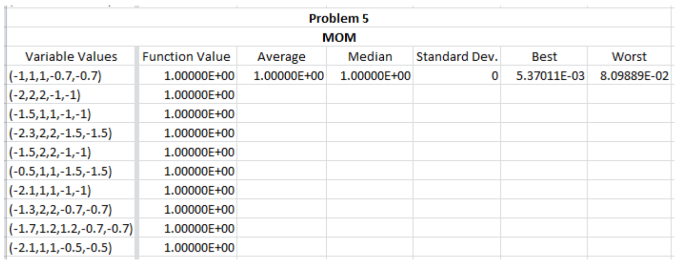
Problem 3						
MOM						
Variable Values	Function Value	Average	Median	Standard Dev.	Best	Worst
(0.1,0.1)	-2.17900E-02	-6.65657E-02	-6.84448E-02	0.041118911	-1.05309E-01	-2.17900E-02
(0.3,0.3)	-1.05460E-01					
(1,1)	-1.05459E-01					
(1,2)	-3.15806E-02					
(1.2,4.2)	-1.05460E-01					
(2,2)	-1.05460E-01					
(3,3)	-1.05309E-01					
(4,2)	-3.15806E-02					
(5,8)	-3.15806E-02					
(6,7)	-2.19774E-02					

Fig. 6. Problem 3 Results using MoM



Problem 5						
Penalty						
Variable Values	Function Value	Average	Median	Standard Dev.	Best	Worst
(-1,1,1,-0.7,-0.7)	1.00000E+00	1.00000E+00	1.00000E+00	0	5.45430E-02	1.07840E-01
(-2,2,2,-1,-1)	1.00000E+00					
(-1.5,1,1,-1,-1)	1.00000E+00					
(-2.3,2,2,-1.5,-1.5)	1.00000E+00					
(-1.5,2,2,-1,-1)	1.00000E+00					
(-0.5,1,1,-1.5,-1.5)	1.00000E+00					
(-2.1,1,1,-1,-1)	1.00000E+00					
(-1.3,2,2,-0.7,-0.7)	1.00000E+00					
(-1.7,1.2,1.2,-0.7,-0.7)	1.00000E+00					
(-2.1,1,1,-0.5,-0.5)	1.00000E+00					

Fig. 7. Problem 5 Results using Penalty Function Method



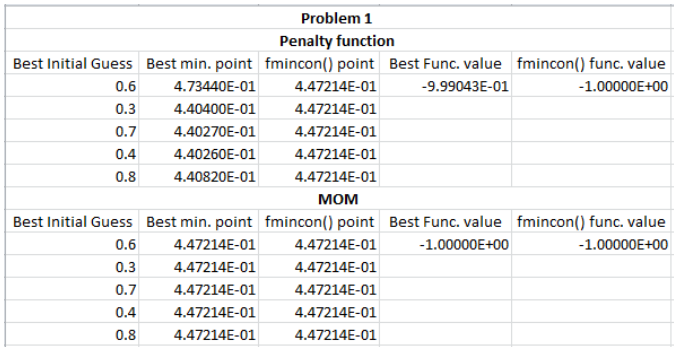
Problem 5						
MOM						
Variable Values	Function Value	Average	Median	Standard Dev.	Best	Worst
(-1,1,1,-0.7,-0.7)	1.00000E+00	1.00000E+00	1.00000E+00	0	5.37011E-03	8.09889E-02
(-2,2,2,-1,-1)	1.00000E+00					
(-1.5,1,1,-1,-1)	1.00000E+00					
(-2.3,2,2,-1.5,-1.5)	1.00000E+00					
(-1.5,2,2,-1,-1)	1.00000E+00					
(-0.5,1,1,-1.5,-1.5)	1.00000E+00					
(-2.1,1,1,-1,-1)	1.00000E+00					
(-1.3,2,2,-0.7,-0.7)	1.00000E+00					
(-1.7,1.2,1.2,-0.7,-0.7)	1.00000E+00					
(-2.1,1,1,-0.5,-0.5)	1.00000E+00					

Fig. 8. Problem 5 Results using MoM

D. Comparison of Results and Plots

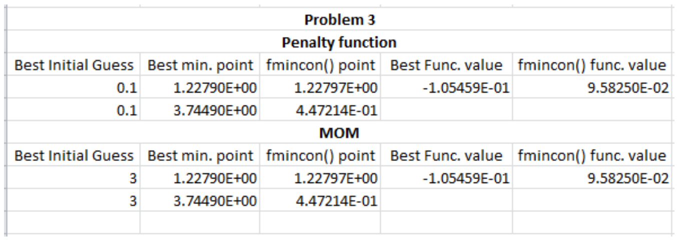
Convergence has been assessed by plotting the function value over each sequence and by observing the number of function calls performed over the sequences. Function values give us a clear view of whether the algorithm is attaining a steady state value or exhibiting fluctuations. It is important to note that this plot does not convey that the steady state value is the correct solution but solely helps us in troubleshooting.

On the same lines, the number of function calls also provide us an alternative approach for analysis. Function calls are made throughout the code and specifically in the multi-variable optimization and uni-directional search subroutines. When the computations attain a steady state implying that the computed solution point is not changing, the number of function calls per sequence become uniform. So near convergence, the plot of function calls with sequence becomes a straight line with the slope remaining constant thereafter.



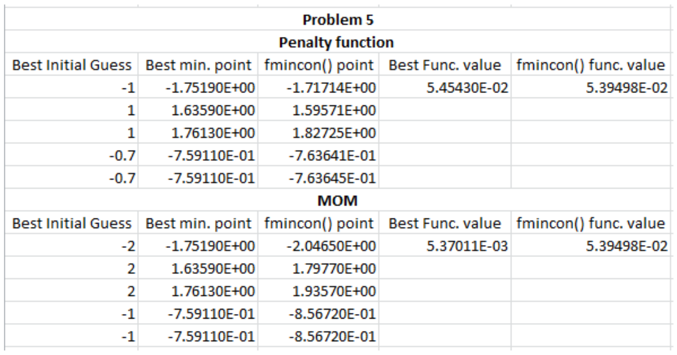
Problem 1				
Penalty function				
Best Initial Guess	Best min. point	fmincon() point	Best Func. value	fmincon() func. value
0.6	4.73440E-01	4.47214E-01	-9.99043E-01	-1.00000E+00
0.3	4.40400E-01	4.47214E-01		
0.7	4.40270E-01	4.47214E-01		
0.4	4.40260E-01	4.47214E-01		
0.8	4.40820E-01	4.47214E-01		
MOM				
Best Initial Guess	Best min. point	fmincon() point	Best Func. value	fmincon() func. value
0.6	4.47214E-01	4.47214E-01	-1.00000E+00	-1.00000E+00
0.3	4.47214E-01	4.47214E-01		
0.7	4.47214E-01	4.47214E-01		
0.4	4.47214E-01	4.47214E-01		
0.8	4.47214E-01	4.47214E-01		

Fig. 9. Problem 1 Result Comparison



Problem 3				
Penalty function				
Best Initial Guess	Best min. point	fmincon() point	Best Func. value	fmincon() func. value
0.1	1.22790E+00	1.22797E+00	-1.05459E-01	9.58250E-02
0.1	3.74490E+00	4.47214E-01		
MOM				
Best Initial Guess	Best min. point	fmincon() point	Best Func. value	fmincon() func. value
3	1.22790E+00	1.22797E+00	-1.05459E-01	9.58250E-02
3	3.74490E+00	4.47214E-01		

Fig. 10. Problem 3 Result Comparison



Problem 5				
Penalty function				
Best Initial Guess	Best min. point	fmincon() point	Best Func. value	fmincon() func. value
-1	-1.75190E+00	-1.71714E+00	5.45430E-02	5.39498E-02
1	1.63590E+00	1.59571E+00		
1	1.76130E+00	1.82725E+00		
-0.7	-7.59110E-01	-7.63641E-01		
-0.7	-7.59110E-01	-7.63645E-01		
MOM				
Best Initial Guess	Best min. point	fmincon() point	Best Func. value	fmincon() func. value
-2	-1.75190E+00	-2.04650E+00	5.37011E-03	5.39498E-02
2	1.63590E+00	1.79770E+00		
2	1.76130E+00	1.93570E+00		
-1	-7.59110E-01	-8.56720E-01		
-1	-7.59110E-01	-8.56720E-01		

Fig. 11. Problem 5 Result Comparison

For problem 1 (detailed in appendix) we have the following results shown in Fig. 12, 13, 14 and 15 using bracket-operator penalty method and method of multipliers.

Similarly for Problem 3 we have the following results as shown in Fig. 16 and 17.

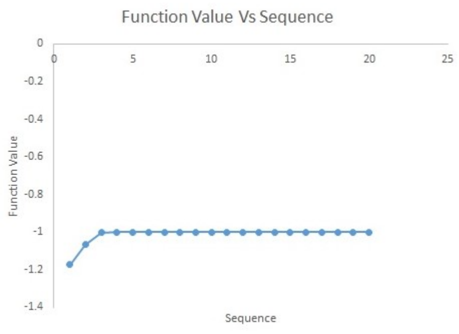


Fig. 12. Method of Multipliers

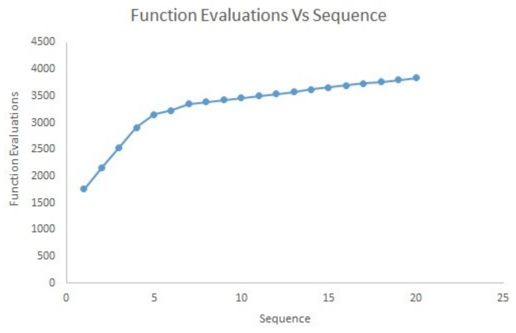


Fig. 13. Method of Multipliers

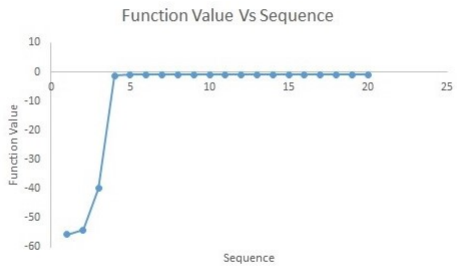


Fig. 14. Bracket-operator penalty method

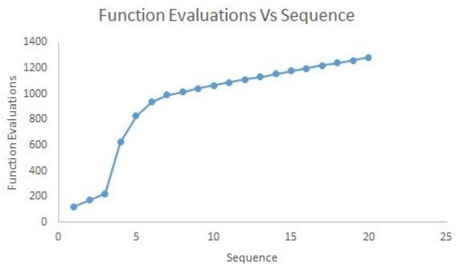


Fig. 15. Bracket-operator penalty method

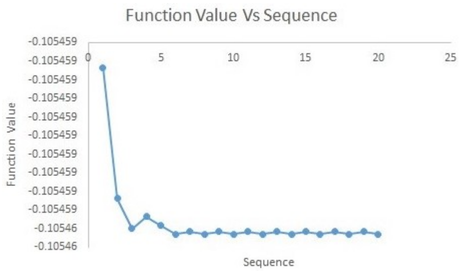


Fig. 16. Method of Multipliers

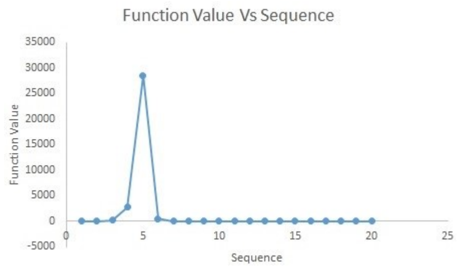


Fig. 17. Bracket-operator penalty method

E. Discussion

Our algorithm works efficiently for most problems using both method of multipliers and bracket-operator penalty method. Results shown above validate this claim for three problems based on comparison with exact solutions and convergence analysis. Interesting observations were made in the context of relative performance of the methods. In a broader sense, bracket-operator penalty method is observed to be giving acceptable solutions over a wider range of initial points. But the increment and value of r needs to be tailored in some cases to the specific problem under consideration. Also, high accuracy solutions are often an outcome of fortitude when using this method as the contours get distorted and may introduce artificial minima.

Method of multipliers enable us to achieve highly accurate solutions to specific problems for instance, problem 1 in this study. The original contours are preserved and so there is no added complication of probabilistic convergence to artificial minima. On the other hand, this method provided better results when the initial point was taken near the actual minima for our analysis. So the versatility in terms of initial point is not as concrete as the bracket-operator method. Finally, the convergence is generally faster using this method in comparison to bracket operator and for certain problems highly accurate solutions are obtained.

Problems 2 and 4 have not been evaluated in this study mainly because of certain egregious difficulties. Problem 2 has two constraints not in accordance with mathematical logic and so if we try to implement them together one would be violated and would unnecessarily distort the contours. We have managed to find the solution to this problem but only in a small range of initial points varying from $(13, 0) - (13, 1)$. Naturally, this special treatment will not be useful for a robust analysis and so we have excluded it. Problem 4 poses a computation challenge in a sense that eight variables are involved with six inequality constraints. Thus over the range of variables given in the problem it was difficult for us to discern where the optima might lie. As a result, a similar strategy of special initial points had to be implemented here and as detailed earlier is not of much practical use.

CONCLUSION

In conclusion we can say that the Bracket-operator penalty method works well over a range of initial points with considerable accuracy. Method of multipliers gives us highly accurate results for certain problems but has a narrower allowance on the initial point values. For unconstrained optimization, conjugate gradient method allows proper refinement from a point away from the minima but may require resetting when the process becomes slow. Finally, unidirectional search using a combination of bisection and bounding phase method is found to work properly for the problems considered in our analysis.

APPENDIX

Problem specifications and their solutions have been listed here for the different cases considered in our analysis.

- Question 1.

$$\max f(x) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i \tag{1}$$

subject to constraint.

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

Search Space: $0 \le x_i \le 1,\ i = 1, 2, \ldots, n.$

- Question 2.

$$\min f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{2}$$

subject to constraints.

$$\begin{aligned} g_1(x) &= (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \le 0, \\ g_2(x) &= (x_1 - 5)^2 + (x_2 - 5)^2 - 82.81 \ge 0. \end{aligned}$$

Search Space: $l \le x_i \le 100,\ i = 1, 2,$ and $l = (13, 0).$

- Question 3.

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

subject to constraints.

$$\begin{aligned} g_1(x) &= x_1^2 - x_2 + 1 \le 0, \\ g_2(x) &= 1 - x_1 + (x_2 - 4)^2 \le 0. \end{aligned}$$

Search Space: $0 \le x_i \le 10,\ i = 1, 2.$

- Question 4.

$$\min f(x) = x_1 + x_2 + x_3, \tag{4}$$

subject to constraints.

$$\begin{aligned} g_1(x) &= -1 + 0.0025(x_4 + x_6) \le 0, \\ g_2(x) &= -1 + 0.0025(-x_4 + x_5 + x_7) \le 0, \\ g_3(x) &= -1 + 0.01(-x_3 + x_8) \le 0, \\ g_4(x) &= 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0, \\ g_5(x) &= x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0, \\ g_6(x) &= x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0. \end{aligned}$$

Search Space: l_i (lower bound) $\le x_i,$

u_i (upper bound), $i = 1, 2, \ldots, 8,$

$$\begin{aligned} l\,(\text{lower bound}) &= 10 * (10, 100, 100, 1, 1, 1, 1, 1), \\ u\,(\text{upper bound}) &= 1000 * (10, 10, 10, 1, 1, 1, 1, 1). \end{aligned}$$

- Question 5.

$$\min f(x) = e^{x_1x_2x_3x_4x_5}, \tag{5}$$

subject to constraints.

$$\begin{aligned} h_1(x) &= x_1^2 + x_2^2 + x_3^2x_4^2x_5^2 - 10 = 0, \\ h_2(x) &= x_2x_3 - 5x_4x_5 = 0, \\ h_3(x) &= x_1^3 + x_2^3 + 1 = 0. \end{aligned}$$

Search Space: l_i (lower bound) $\le x_i\, u_i$ (upper bound), $i = 1, 2, \ldots,$

$$\begin{aligned} u\,(\text{upper bound}) &= (2.3, 2.3, 3.2, 3.2, 3.2), \\ l\,(\text{lower bound}) &= -u\,(\text{upper bound}). \end{aligned}$$