Solution of Constraint Optimization problem with Bracket Operator Penalty method

N. V. Sonule
Department of Mechanical Engineering
Indian Institute of Technology, Guwahati
Guwahati, India
s.niraj@iitg.ac.in

R. M. Majumdar
Department of Chemical Engineering
Indian Institute of Technology, Guwahati
Guwahati, India
ritam170107030@iitg.ac.in

***Abstract*—This paper focuses on solving constraint optimization problems using bracket operator penalty method. This method transforms the constrained problem into unconstrained problem which can be solved using various optimization techniques. In this work Marquardt's method is implemented for solving the problem. Many such sequences are performed to obtain the converged solution.**

Keywords—constrained optimization, penalty method, Marquardt's method

I. Introduction

In many practical cases engineers come across problems which involve solving optimization problems to carry out their work and most of them are constrained problems. Few areas are manufacturing, transportation, welding, heat treatment etc. In this paper a solution which can be used to solve such constrained optimization problems is discussed.

In this work we have used bracket operator penalty method to solve the constrained optimization problem. The bracket operator penalty method, which is an exterior penalty method, transforms the constrained problem into unconstrained problem by adding penalty to the objective function whenever the constraint is violated. This transformation also involve a penalty parameter *R* which starts with a small value and then increased by a factor *c* as we move to the next sequence. Many such sequences of penalized objective function are solved using Marquardt's method with implementation of unidirectional search using bounding phase method followed by interval halving method. After a number of such sequences the solution is said to be converged when the difference between the function values of the penalized function of the current sequence and the previous sequence is smaller than set termination parameter. The algorithm for the complete solution is briefly discussed in the next section.

This method changes our objective function which leads to deviation of solution from the best possible solution. This deviation in solution depends on many factors such as parameter setting (*R* and *c),* nature of objective function, number of constraints and their nature. Hence, to make sure of the obtained solution, we have solved a particular problem with 10 different starting points and then after analyzing the obtained data we have concluded the final solution.

A constrained optimization formulation can be written as:
Minimize f(x),
Subject to,
$g_j(x) \geq 0, j = 1,2,\dots, J,$
$h_k(x) = 0,\ k = 1,2,\dots, K,$
$xi^{(L)} \leq xi \leq xi^{(U)}, i = 1, 2, \dots, N.$
Decision variables: $x = \{x_1, x_2, \dots, x_N\}^T$.
f(x) is the objective function.
$g_j(x)$ are inequality constraints.
$h_k(x)$ are equality constraints.

II. Description Of Algorithm

A. *Penalty function method*

Penalty function methods work in a series of sequence, each time modifying a set of penalty parameters and starting a sequence with the point obtained in the previous sequence. Generally penalty function can be written as,

$$P(x, R) = f(x) + \Omega(R, g(x), h(x))$$

where R is a set of penalty parameters, Ω is the penalty term chosen to favor the selection of feasible point over infeasible point, g(x) is inequality constraint and h(x) is equality constraints.

In our solution, we have used Bracket Operator Penalty Function for inequality constraints, which is defined as,

$$\Omega = R \langle g(x)\rangle^2$$

where $\langle\alpha\rangle = \alpha$ when α is negative; zero otherwise. This method assigns positive value to infeasible point; hence it is an exterior penalty term. It starts with small value of R which increases gradually. For equality constraints we use Parabolic Penalty function, which is defined as,

$$\Omega = R\{h(x)\}^2.$$

It is also an exterior penalty term.

The transformed penalty function is an unconstrained minimization objective function which can be solved by any of the unconstrained optimization techniques. In this work we have used Marquardt's method with incorporation of unidirectional search with help of bounding phase followed by interval halving method. After solving such sequences of unconstrained problem, we say that we have found the solution for our constrained problem, when the difference between penalty function value of current and previous sequence is negligible.

Algorithm:

Step 1 Choose two termination parameters ε1, ε2, 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 $c \geq 1$ is used for exterior penalty terms. Set t = 0.

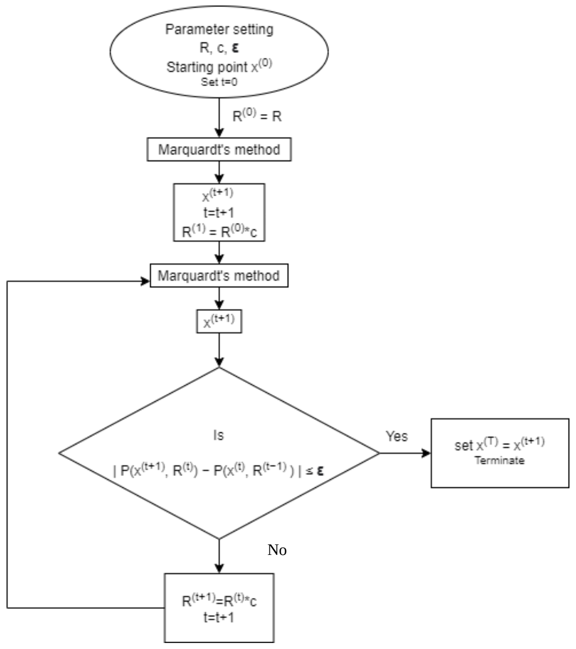
Step 2 Form $P(x(t), R(t)) = f(x(t)) + \Omega(R(t), g(x(t)), h(x(t)))$.

Step 3 Starting with x (t), find x (t+1) such that P(x (t), R(t)) is minimum for a fixed value of R(t). Use ε1 to terminate the unconstrained search.

Step 4 Is $|P(x(t+1), R(t)) - P(x(t), R(t-1))| \leq \varepsilon 2$? If yes, set x (T) = x (t+1) and terminate; Else go to Step 5.

Step 5 Choose R(t+1) = c R(t). Set t = t + 1 and go to Step 2.

Flow Chart:



B. *Marquardt's method:*

Marquardt's method is an unconstrained optimization technique for multivariable problem which uses hessian as well as gradient of the objective function to find a search direction along which a new solution is found from the previous solution. The search direction is found as,

$$s(x^{(k)}) = -[H^{(k)} + \lambda^{(k)}*I]^{-1}\nabla f(x^{(k)})$$

where, s is the direction of change, $H^{(k)}$ represents the Hessian Matrix, and $\nabla f(x^{(k)})$ represents the gradient of objective function. For larger value of $\lambda^{(k)}$, the Hessian $H^{(k)}$ has little effect on the determination of the search direction $s(x^{(k)})$, which means $s(x^{(k)})$ is effectively $-\nabla f(x^{(k)})$, which is Cauchy's steepest descent method, whereas for smaller value $\lambda^{(k)}$, the search direction $s(x^{(k)})$ is same as Newton's method. Hence, Marquardt's method is a combination of Cauchy's method and Newton's method, where Cauchy's method works well when the initial point is far away from the minimum point, and Newton's method works well when the initial point is near to the minimum point.

The new point is calculated as,

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

where $x^{(k+1)}$ is the new point, $x^{(k)}$ is the current point and α is a unidirectional search parameter. α transforms the multivariable function into single variable function. We take that value of α which will minimize our objective function. This single variable objective function can be solved by various techniques. In this work we have used interval having method for which the initial bounds is found by Bounding Phase method.

Algorithm:

Step 1 Choose a starting point, x (0), the maximum number of iterations, M, and a termination parameter, ε. Set k = 0 and λ (0) = 10^4 (a large number).

Step 2 Calculate ∇f(x(k).

Step 3 If ||∇f(x (k))|| ≤ ε or k ≥ M? Terminate; Else go to Step 4.

Step 4 Calculate $s(x^{(k)}) = -[H^{(k)} + \lambda^{(k)}*I]^{-1}\nabla f(x^{(k)})$.

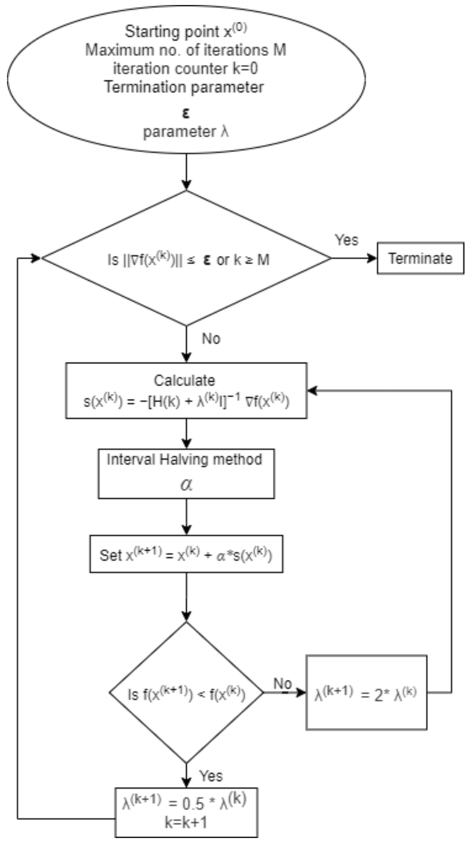
Set $x^{(k+1)} = x^{(k)} + \alpha*s(x^{(k)})$.

Step 5 Is f(x(k+1)) < f(x(k))? If yes, go to Step 6; Else go to Step 7.

Step 6 Set λ(k+1) = 0.5*λ(k), k = k + 1, and go to Step 2.

Step 7 Set λ(k+1) = 2λ(k) and go to Step 4.

Flow chart:



Starting point $x^{(0)}$
Maximum no. of iterations M
iteration counter k=0
Termination parameter
ε
parameter λ
Is $||\nabla f(x^{(k)})|| \le$ ε or k ≥ M
Yes
Terminate
No
Calculate
$s(x^{(k)}) = -[H(k) + \lambda^{(k)}I]^{-1} \nabla f(x^{(k)})$
Interval Halving method
α
Set $x^{(k+1)} = x^{(k)} + \alpha*s(x^{(k)})$
Is $f(x^{(k+1)}) < f(x^{(k)})$
No
$\lambda^{(k+1)} = 2* \lambda^{(k)}$
Yes
$\lambda^{(k+1)} = 0.5 * \lambda^{(k)}$
k=k+1

C. *Interval halving method:*

Interval halving method is a single variable optimization technique in which the interval between the upper and lower bounds is halved in every iteration. The initial bound are obtained from Bounding Phase method. We carry on such iteration until the interval reduces to small value (say 0.001). The optimum solution can be taken as the mean of bounds in the terminating interval.

Algorithm:

Step 1 Choose a lower bound a and an upper bound b. Choose also a small delta. Let $x_m = (a + b)/2$, L = b − a. Compute $f(x_m)$.

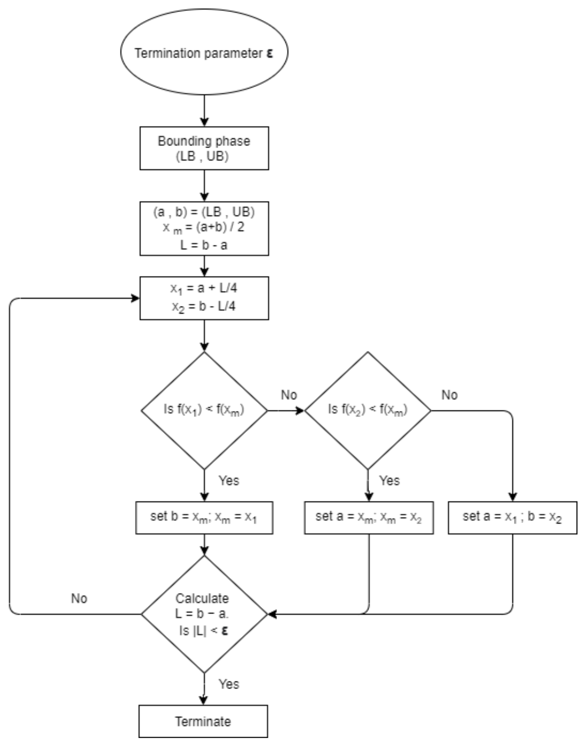
Step 2 Set $x_1 = a + L/4$, $x_2 = b - L/4$. Compute $f(x_1)$ and $f(x_2)$.

Step 3 If $f(x_1) < f(x_m)$ set $b = x_m$; $x_m = x_1$; go to Step 5; Else go to Step 4.

Step 4 If $f(x_2) < f(x_m)$ set $a = x_m$; $x_m = x_2$; go to Step 5; Else set $a = x_1$, $b = x_2$; go to Step 5.

Step 5 Calculate L = b − a. If |L| < ε, $x^*=x_m$ and Terminate; Else go to Step 2.

Flow chart:



D. *Bounding phase method:*

Bounding phase method is used to bound the optimum solution for a single variable objective function by comparing the function value between two points.

Algorithm:

Step 1 Choose an initial guess $x^{(0)}$ 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;

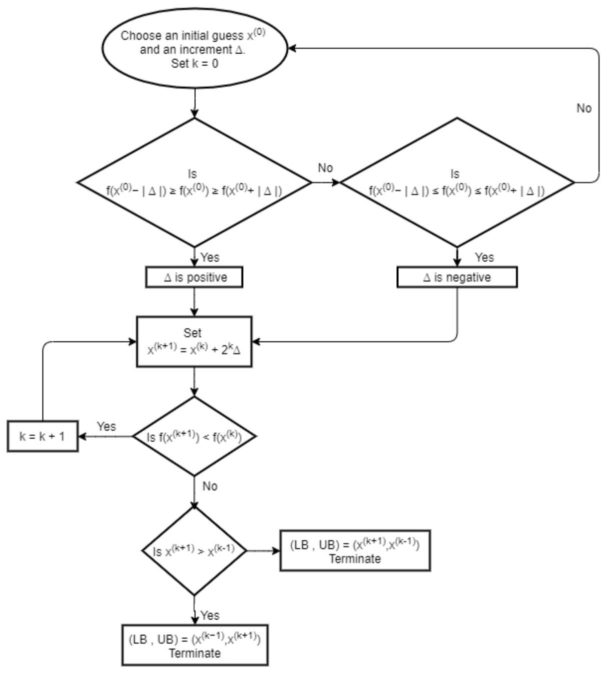
Else if $f(x^{(0)}-|\Delta|) \leq f(x^{(0)}) \leq f(x^{(0)}+| \Delta |)$, then Δ is negative;

Else go to Step 1.

Step 3 Set $x^{(k+1)} = x^{(k)} + 2^k\Delta$, (other exponent can be used).

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

Flow chart:


Choose an initial guess x(0) and an increment Δ. Set k = 0
Is f(x(0)− | Δ |) ≥ f(x(0)) ≥ f(x(0)+ | Δ |)
No
Is f(x(0)− | Δ |) ≤ f(x(0)) ≤ f(x(0)+ | Δ |)
No
Yes
Δ is positive
Yes
Δ is negative
Set x(k+1) = x(k) + 2kΔ
k = k + 1
Yes
Is f(x(k+1)) < f(x(k))
No
Is x(k+1) > x(k-1)
(LB , UB) = (x(k+1),x(k-1)) Terminate
Yes
(LB , UB) = (x(k−1),x(k+1)) Terminate

III. Results And Discussion

A. *System details*

1) Hardware
   a) Operating system: Windows 10
   b) RAM: 8GB
   c) ROM:1TB
2) Programming platform

MATLAB R2019b

B. *Parameter setting*

Penalty parameter:

R=0.1; c=10;

Termination parameter:

ε_1 = 01e-05 (for penalty method)

ε_2 = 01e-03 (for Marquardt's method)

C. *Results*

1) Problem 1

	Starting point	Minima x*	Optimum function value f(x*)
1	(-5.946,1.868)	(14.09511,0.84317)	-6961.576535
2	(10.272,-1.499)	(14.09511,0.84317)	-6961.576575
3	(13.218,-2.885)	(14.09511,0.84317)	-6961.576655
4	(-13.910,-13.692)	(14.09511,0.84317)	-6961.576594
5	(-8.492,8.766)	(14.09511,0.84317)	-6961.57661
6	(-12.727,10.966)	(14.09511,0.84317)	-6961.576677
7	(8.472,3.132)	(14.09511,0.84317)	-6961.576686
8	(6.421,-6.023)	(14.09511,0.84317)	-6961.576584
9	(-11.667,14.951)	(14.09511,0.84317)	-6961.576727
10	(2.662,5.451)	(14.09511,0.84317)	-6961.576674
Best			-6961.576727
Worst			-6961.576535
Mean			-6961.576632
Median			-6961.576633
SD*			6.07032E-05

Table 1: Results for 10 starting points

Convergence plots:

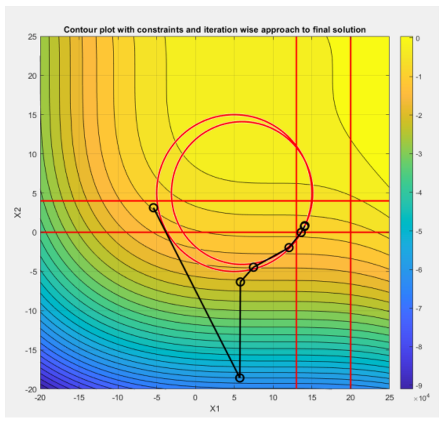


Figure 1

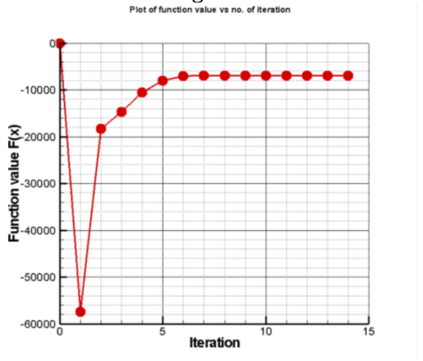


Figure 2

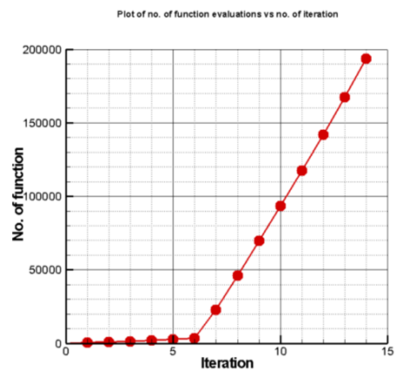


Figure 3

2) Problem 2

This is a maximization problem but for applying the minimization algorithm we change the nature of optimization problem as well constraints according to standard minimization problem.

	Starting point	Minima x*	Optimum function value f(x*)
1	(-1.624,-9.688)	(1.22801,4.24543)	-0.095825027
2	(2.307,1.122)	(1.32422,3.43059)	-0.027263379
3	(-6.592,4.197)	(1.73414,4.74626)	-0.029143789
4	(-5.653,-9.324)	(1.67402,3.80234)	-0.025812452
5	(-0.413,-5.285)	(1.22796,4.24528)	-0.095825023
6	(-3.911,2.391)	(1.22800,4.24522)	-0.095824988
7	(-4.549,0.574)	(1.22792,4.24521)	-0.095824975
8	(4.446,8.567)	(1.22800,4.24553)	-0.095824993
9	(7.960,-3.422)	(1.22796,4.24533)	-0.095825037
10	(2.429,7.110)	(1.73411,4.74621)	-0.029143794
Best			-0.095825037
Worst			-0.025812452
Mean			-0.068631346
Median			-0.095824981
SD*			0.035119275

** SD stands for standard deviation.*

Table 1: Results for 10 starting points

Convergence plots:

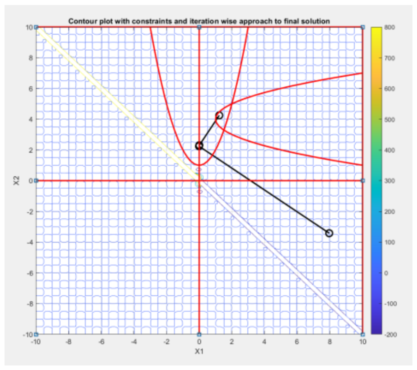


Figure 4

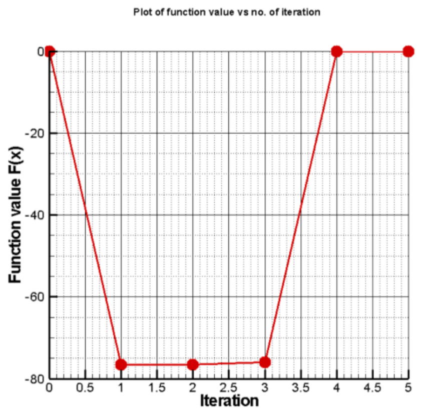


Figure 5

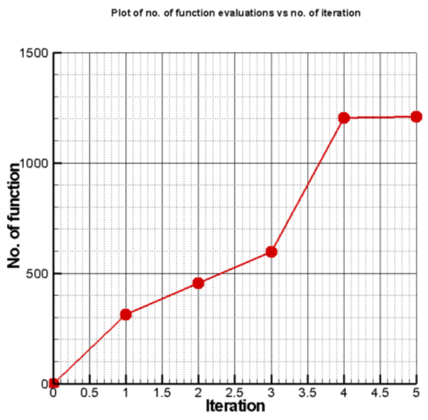


Figure 6

3) Problem 3

This problem is a complex problem with 8 variables and 6 constraints and bounds for all variables. Due to this complexity in the problem it was difficult to find the solution for different starting points. So, we took a starting point near optimum solution as provided in Appendix. By doing so we were able to get some solution near to optima. Few such solutions are mentioned below.

	x*	f(x*)
1	(100,1000,4279.4,108.2,229.3,289.3,259.5,387.4)	5379.4
2	(100,1000,4095.2,113.3,188.2,279.5,206.9,378.5)	5195.2
3	(119.7,1003.4,5103.1,95.6,10,150.1,242.7,250.1)	6226.1
4	(100,1103.6,4227.3,62.2,220.6,285.8,241.6,385.8)	5430.9
5	(100,1000,4073.2,108.6,213.9,290,263.2,289.5)	5173.2

Optimum solution for this problem is

x∗ = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)

f (x∗) = 7049.3307.

Few reasons for this deviation may be:

i) Generation of local minima due addition of penalty terms.

ii) Improper selection of algorithm parameters.

iii) As the problem is solved using Marquardt's method which involves finding inverse of a matrix, the algorithm fails when the inverse of matrix is very close to singularity.

iv) Complexities such as non-linear constraints, many variables and constraints etc.

IV. Conclusions

On the basis of above observations and results we can conclude that

i) The solution discussed in this paper is able to solve problems with lesser complexities.

ii) The solution obtained depends on the choice of penalty parameters. Therefore choice of these parameters must be done wisely.

iii) The variation of solution can be explained by distortion of contours of objective function resulting in creation of local minimas. As we move from one sequence to another, this distortion also increases due increase in the value of R.

iv) To avoid such distortion we can use Method of Multipliers which does not distort the original objective function but shifts it toward the constrained optimum point.

v) As some methods involve finding inverse of matrix in algorithm, the algorithm may collapse when this matrix is very close to singularity.

vi) The solution also depends strongly on selection of starting point as it decides number of sequences required to achieve the solution. If the starting point is far away the algorithm may require more sequences which will lead to more distortion of our objective function.

vii) In case many variables and constraints are present in our optimization problem the complexity increases due to addition of more and more penalty terms. This also may lead to variation in the obtained solution.

V. Appendix

A. Problem 1:

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

Constraints:

$g_1(x) = (x_1-5)^2 + (x_2-5)^2 - 100 \geq 0$

$g_2(x) = (x_1-6)^2 + (x_2-5)^2 - 82.81 \leq 0$

$13 \leq x_1 \leq 20;\ 0 \leq x_2 \leq 4$

Global Minima:

$x^*=(14.095, 0.8429)$

$f(x^*)=-6961.813$

B. Problem2:

Maximize

$f(x) = sin\verb|^|3(2*pi*x1)*sin(2*pi*x2)/(x1\verb|^|3*(x1+x2))$

Constraints:

$g_1(x) = x_1^2 - x_2 + 1 \leq 0$

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

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

Global Minima:

$x^*=(1.227, 4.245)$

$f(x^*) = 0.0958$

C. Problem 3:

Minimize $f(x) = x1 + x2 + x3$

Constraints:

$g_1(x) = -1 + 0.0025(x_4+x_6) \leq 0$

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

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

$g_4(x) = 100*x_1 - x_1*x_6 + 833.33252*x_4 - 83333.52 \leq 0$

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

$g6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0$

$100 \leq x_1 \leq 10000$

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

$10 \leq x_i \leq 10000 \ i=4,5,6,7,8$

Global Minima:

x∗ = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162,395.5979)

f (x∗) = 7049.3307.