IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

OPTIMIZATION IN CHEMICAL ENGINEERING

Constrained Nonlinear Programming

(WEEK – 10: LECTURE – 46)

Dr. Debasis Sarkar

Department of Chemical Engineering

Indian Institute of Technology Kharagpur

Constrained Nonlinear Programming

Week 10:

- Successive Linear Programming Method
- Successive Quadratic Programming Method
- Penalty Function Method
- Augmented Lagrange Multiplier Method

- **Today's Topic:**

- Successive Linear Programming Method

Karush – Kuhn – Tucker (KKT) Conditions: Review

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to } g_j(\mathbf{x}) \le 0 \quad \text{for } j = 1, 2, ..., p$$

$$h_k(\mathbf{x}) = 0 \quad \text{for } k = 1, 2, ..., m$$

The Lagrangian:

$$L(\mathbf{x}, \boldsymbol{u}, \boldsymbol{v}) = f(\mathbf{x}) + \sum_{i=1}^{m} v_i h_i(\mathbf{x}) + \sum_{j=1}^{p} u_j g_j(\mathbf{x})$$

$$\Rightarrow L(\mathbf{x}, \boldsymbol{u}, \boldsymbol{v}) = f(\mathbf{x}) + \boldsymbol{v}^T \mathbf{h}(\mathbf{x}) + \boldsymbol{u}^T \mathbf{g}(\mathbf{x})$$

1. Gradient conditions:

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{u}^*, \boldsymbol{v}^*)}{\partial x_i} = \frac{\partial f(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{m} v_j^* \frac{\partial h_j(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{p} u_j^* \frac{\partial g_j(\mathbf{x}^*)}{\partial x_i} = 0, \; i = 1, 2, ..., n$$

2. Feasibility check:

$$g_j(\mathbf{x}^*) \le 0, \quad j = 1, 2, ..., p$$

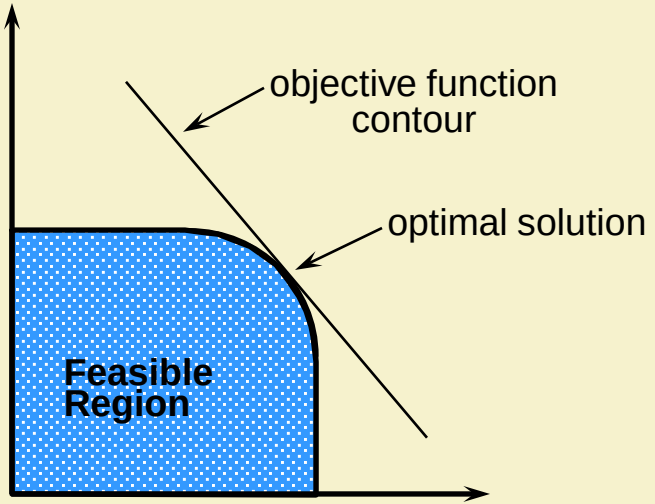
$$h_i(\mathbf{x}^*) = 0, \quad i = 1, 2, ..., m$$

3. Switching conditions: $u_j^* g_j = 0, \quad j = 1, 2, ..., p$

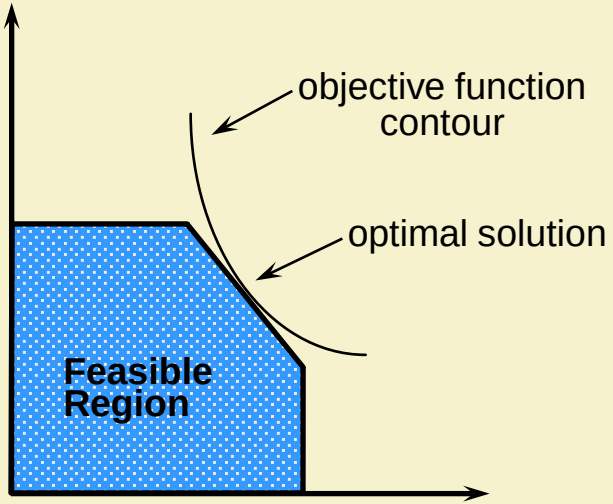
4. Non-negativity of Lagrange multipliers for inequalities: $u_j^* \ge 0, \quad j = 1, 2, ..., p$

5. Regularity check: Gradients of active constraints must be linearly independent.

Nonlinear Programming Problems

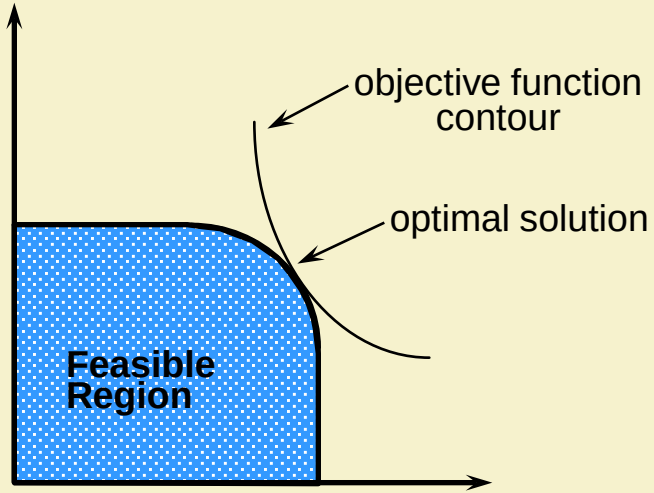


**linear objective function,
nonlinear constraints**

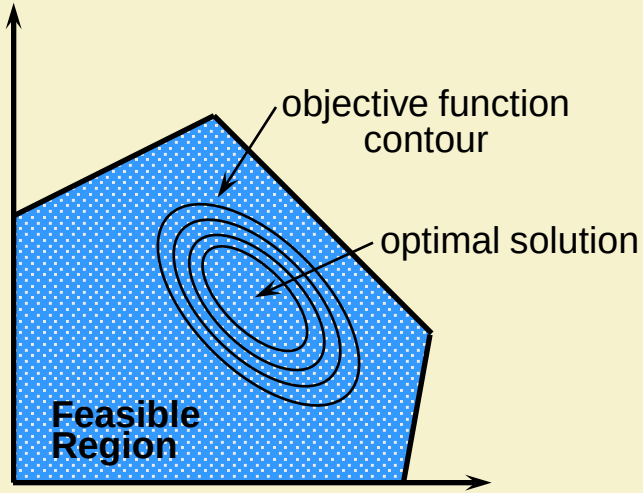


**nonlinear objective function,
linear constraints**

Nonlinear Programming Problems



**nonlinear objective function,
nonlinear constraints**



**nonlinear objective function,
linear constraints**

Constrained Nonlinear Programming: Techniques

The methods available for the solution of a constrained nonlinear programming problems can be classified into two broad categories: direct methods and indirect methods. Direct methods handle the constraints in an explicit manner. Generally, indirect methods solve the constrained problem as a sequence of unconstrained problems.

Direct Methods

- Random search methods
- Heuristic search methods
- Sequential linear programming method
- Sequential quadratic programming method
- Generalized reduced gradient method

Indirect Methods

- Transformation of variables technique
- Sequential unconstrained minimization
- Penalty function method
- Augmented Lagrange multiplier method

Characteristics of a Constrained Optimization Problem

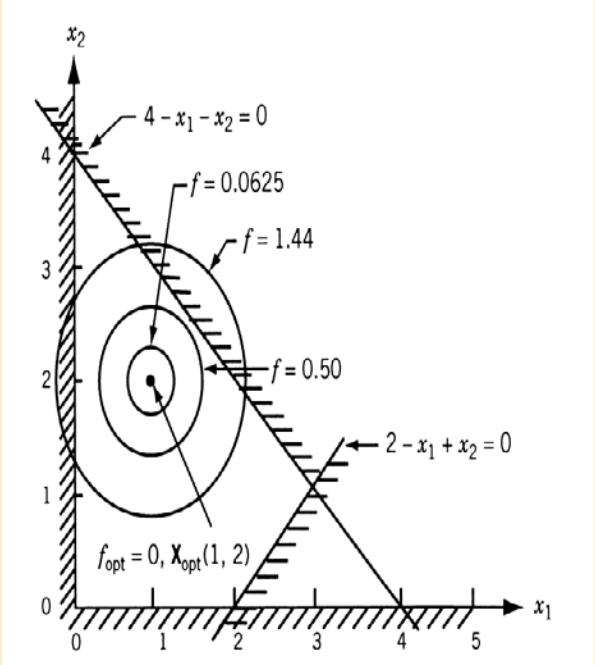
1. No effect of constraints:

The constrainst may have no effect on the optimum point. In this case, the constrained optimum is the same as the unconstrained optimum.

The minimum point X* can be found by using the necessary and sufficient conditions.

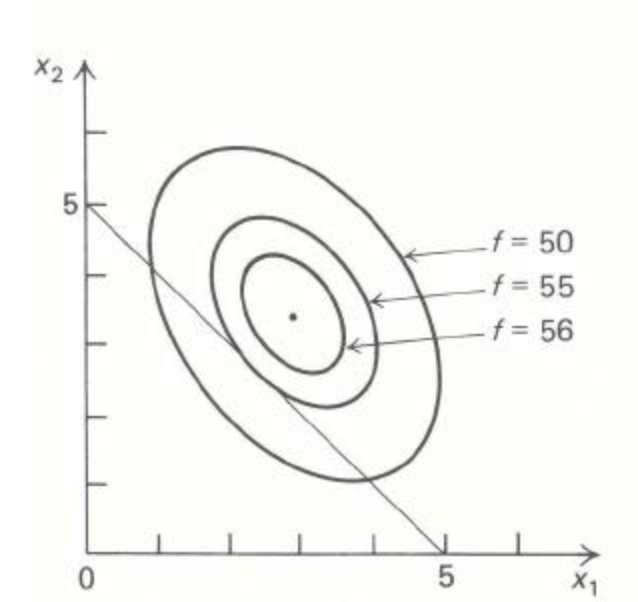
$$\nabla f(X^*) = 0$$

$$\nabla^2 f(X^*) = \text{Positive definite}$$



Characteristics of a Constrained Optimization Problem

2. Constrained Optimum Will Usually be Different



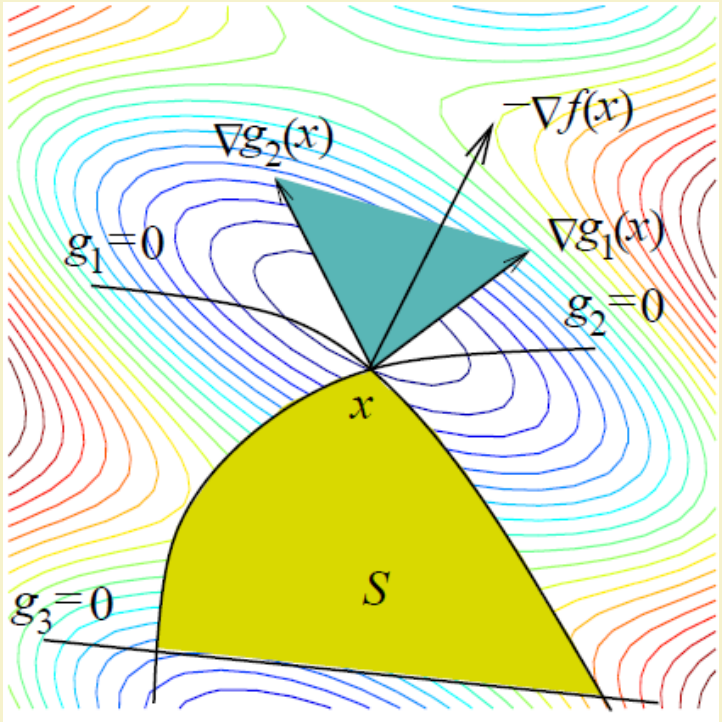
Maximization problem

Characteristics of a Constrained Optimization Problem

3. Optimum Occurs on Constraint Boundary

According to the Karush – Kuhn – Tucker necessary conditions, the negative of the gradient can be expressed as a positive linear combination of the gradients of the active constraints.

$$-\nabla f(\mathbf{x}^*) = \sum_{j=1}^{m} v_j^* \nabla h(\mathbf{x}^*) + \sum_{j=1}^{p} u_j^* \nabla g(\mathbf{x}^*)$$

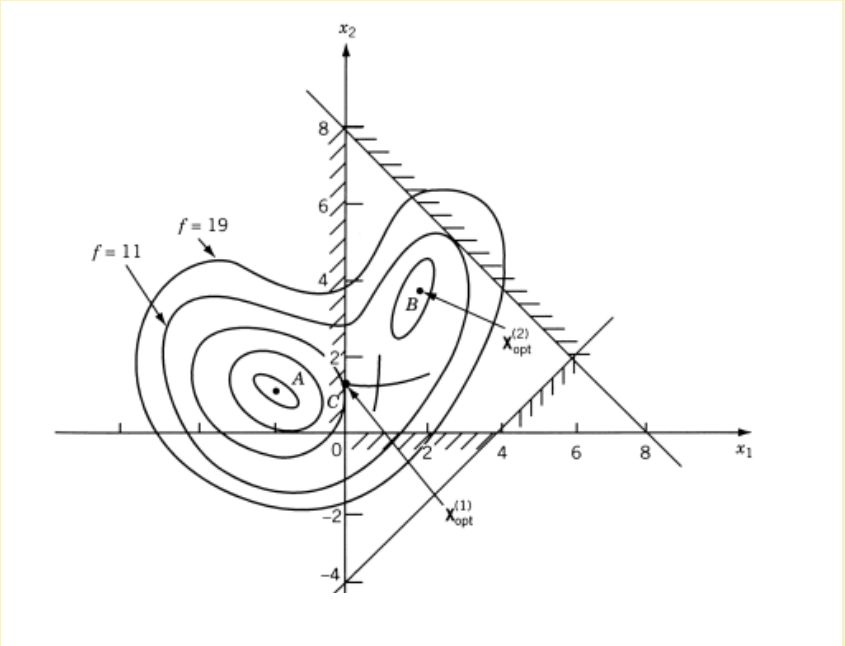


Characteristics of a Constrained Optimization Problem

4. Multiple Optima

If the objective function has multiple unconstrained local minima, the constrained problem may also have multiple minima.

Sometimes, even if the objective function has a single unconstrained minimum, the constraints may introduce additional local minima.



Successive/Sequential Linear Programming Method

- The original nonlinear programming problem is solved by solving a series of LPP.
- Each LPP is obtained by approximating the nonlinear objective and constraint functions using first-order Taylor series expansions about the current estimate of the optimum (decision/design variable vector) $\mathbf{X}_i$.
- The LPP can be solved by simplex method to find the new design vector $\mathbf{X}_{i+1}$.
- If $\mathbf{X}_{i+1}$ is not the desired (converged) solution, the problem is relinearized about the point $\mathbf{X}_{i+1}$ and the procedure is repeated until we find the optimum solution $\mathbf{X}^*$.

Sequential Linear Programming: Linearly Constrained Nonlinear Program

Consider the linearly constrained NLP:

This differs from standard LPP only in the presence of nonlinear objective function $f(x)$.

$$\text{Minimize } f(x)$$
$$\text{Subject to } Ax \le b$$
$$x \ge 0$$

Now the optimal solution can lie anywhere within the feasible region – not necessarily on the corner points only.

The linearization of the above problem around some feasible point x^0 is clearly an LPP and it will have an optimal solution at a feasible corner point, if the feasible region is bounded.

$$\text{Minimize } \tilde{f}(\bar{x}; x^0)$$
$$\text{Subject to } Ax \le b$$
$$x \ge 0$$

How the solution of the approximate problem is related to the solution of the original problem? Some type of line search is required in order to approach the solution of the original problem from the approximate solution.

Sequential Linear Programming: Linearly Constrained NLP: Frank-Wolfe Algorithm

$Step-1$: Define starting point $x^{(0)}$; Termination parameters $\varepsilon > 0,\ \delta > 0$

Compute $\nabla f(x^{(t)})$. If $\left\|\nabla f(x^{(t)})\right\| \le \varepsilon$, *Stop*, Else go to $Step-2$

$Step-2$: Solve LP subproblem: Minimize $\nabla f(x^{(t)})y$, Subject to $Ay \le b$, $y \ge 0$

Let $y^{(t)}$ be the optimal solution to the LPP.

$Step-3$: Find $\alpha^{(t)}$ which solves: Minimize $f(x^{(t)} + \alpha(y^{(t)} - x^{(t)}))$, $0 \le \alpha \le 1$

$Step-4$: Calculate: $x^{(t+1)} = x^{(t)} + \alpha^{(t)}(y^{(t)} - x^{(t)})$

$Step-5$: Convergence check: If $\left\|x^{(t+1)} - x^{(t)}\right\| < \delta \left\|x^{(t+1)}\right\|$ and if $\left\|f(x^{(t+1)} - x^{(t)})\right\| \le \varepsilon \left\|f(x^{(t+1)})\right\|$

then *Stop*, Else go to $Step-1$.

Sequential Linear Programming: General NLP

Consider the following NLP:

$$\text{Minimize } f(x)$$

$$\text{Subject to: } g_j(x) \geq 0, \quad j = 1, \ldots, J$$

$$h_k(x) = 0, \quad k = 1, \ldots, K$$

$$x_i^{LB} \leq x_i \leq x_i^{UB}, \quad i = 1, \ldots, N$$

Given an estimate $x^{(t)}$, the following linear approximation problem can be constructed at $x^{(t)}$.

$$\text{Minimize } f(x^{(t)}) + \nabla f(x^{(t)})(x - x^{(t)})$$

$$\text{Subject to: } g_j(x^{(t)}) + \nabla g_j(x^{(t)})(x - x^{(t)}) \geq 0, \quad j = 1, \ldots, J$$

$$h_k(x^{(t)}) + \nabla h_k(x^{(t)})(x - x^{(t)}) = 0, \quad k = 1, \ldots, K$$

$$x_i^{LB} \leq x_i \leq x_i^{UB}, \quad i = 1, \ldots, N$$

Solution to this LPP gives us $x^{(t+1)}$. If $x^{(t+1)}$ is infeasible, reinitialize at $x^{(t+1)}$.

Sequential Linear Programming: General NLP: Example

Consider the following NLP:

$$\text{Minimize } f(x) = x_1^2 + x_2^2$$

$$\text{Subject to: } g(x) = x_1^2 - x_2 \geq 0$$

$$h(x) = 2 - x_1 - x_2^2 = 0$$

$$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$$

$$\nabla f = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \; f(2,1) = 5$$

$$\text{Now, } f(x^{(t)}) + \nabla f(x^{(t)})(x - x^{(t)})$$

$$= 5 + \begin{bmatrix} 4 & 2 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix}$$

$$= 5 + 4(x_1 - 2) + 2(x_2 - 1)$$

$$\nabla g = \begin{bmatrix} 2x_1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}, \; g(2,1) = 3$$

$$\text{Now, } g(x^{(t)}) + \nabla g(x^{(t)})(x - x^{(t)})$$

$$= 3 + \begin{bmatrix} 4 & -1 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix}$$

$$= 3 + 4(x_1 - 2) - (x_2 - 1)$$

We want to construct the linear approximation to this problem at $x^0 = (2, 1)$.

$$\nabla h = \begin{bmatrix} -1 \\ -2x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \; h(2,1) = -1$$

$$\text{Now, } h(x^{(t)}) + \nabla h(x^{(t)})(x - x^{(t)}) = -1 + \begin{bmatrix} -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix} = -1 - (x_1 - 2) - 2(x_2 - 1)$$

Sequential Linear Programming: General NLP: Example

Consider the following NLP:

$$\text{Minimize } f(x) = x_1^2 + x_2^2$$

$$\text{Subject to: } g(x) = x_1^2 - x_2 \geq 0$$

$$h(x) = 2 - x_1 - x_2^2 = 0$$

$$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$$

$$\text{Minimize } f(\bar{x}; x^0) = 5 + 4(x_1 - 2) + 2(x_2 - 1)$$

$$\text{Subject to: } \tilde{g}(\bar{x}, x^0) = 3 + 4(x_1 - 2) - (x_2 - 1)$$

$$\tilde{h}(\bar{x}, x^0) = -1 - (x_1 - 2) - 2(x_2 - 1)$$

$$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$$

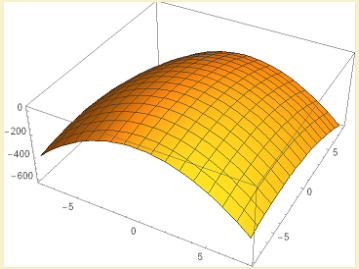
The solution to this LPP is $x^{(1)}$ = (11/9, 8/9)

$g(x^{(1)}) = 0.6049 > 0$, $h(x^{(1)}) = -0.0123 \neq 0$

Therefore, we reinitialize the problem at $x^{(1)}$.

Continue until we get x^* = (1, 1) as optimal solution.

Optimization in Chemical Engineering



Thank You

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

OPTIMIZATION IN CHEMICAL ENGINEERING

Constrained Nonlinear Programming

(WEEK – 10: LECTURE – 47)

Dr. Debasis Sarkar

Department of Chemical Engineering

Indian Institute of Technology Kharagpur

Constrained Nonlinear Programming

Week 10:

- Successive Linear Programming Method
- Successive Quadratic Programming Method
- Penalty Function Method
- Augmented Lagrange Multiplier Method

Today's Topic:

- Successive Quadratic Programming Method

Karush – Kuhn – Tucker (KKT) Conditions: Review

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to } g_j(\mathbf{x}) \le 0 \quad \text{for } j = 1, 2, ..., p$$

$$h_k(\mathbf{x}) = 0 \quad \text{for } k = 1, 2, ..., m$$

The Lagrangian:

$$L(\mathbf{x}, \boldsymbol{u}, \boldsymbol{v}) = f(\mathbf{x}) + \sum_{i=1}^{m} v_i h_i(\mathbf{x}) + \sum_{j=1}^{p} u_j g_j(\mathbf{x})$$

$$\Rightarrow L(\mathbf{x}, \boldsymbol{u}, \boldsymbol{v}) = f(\mathbf{x}) + \boldsymbol{v}^T \mathbf{h}(\mathbf{x}) + \boldsymbol{u}^T \mathbf{g}(\mathbf{x})$$

1. Gradient conditions:

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{u}^*, \boldsymbol{v}^*)}{\partial x_i} = \frac{\partial f(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{m} v_j^* \frac{\partial h_j(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{p} u_j^* \frac{\partial g_j(\mathbf{x}^*)}{\partial x_i} = 0, \; i = 1, 2, ..., n$$

2. Feasibility check:

$$g_j(\mathbf{x}^*) \le 0, \quad j = 1, 2, ..., p$$

$$h_i(\mathbf{x}^*) = 0, \quad i = 1, 2, ..., m$$

3. Switching conditions: $u_j^* g_j = 0, \quad j = 1, 2, ..., p$

4. Non-negativity of Lagrange multipliers for inequalities: $u_j^* \ge 0, \quad j = 1, 2, ..., p$

5. Regularity check: Gradients of active constraints must be linearly independent.

Quadratic Programming Problems

A general NLP:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to} \quad g_j(\mathbf{x}) \le 0 \quad \text{for} \quad j = 1, 2, ..., p$$

$$h_k(\mathbf{x}) = 0 \quad \text{for} \quad k = 1, 2, ..., m$$

Here, $f(x)$, $g(x)$, $h(x)$ all are, in general, nonlinear functions.

The quadratic programming problem:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to} \quad g_j(\mathbf{x}) \le 0 \quad \text{for} \quad j = 1, 2, ..., p$$

$$h_k(\mathbf{x}) = 0 \quad \text{for} \quad k = 1, 2, ..., m$$

Here, the objective function $f(x)$ is quadratic function.
The constraints $g(x)$, $h(x)$ are linear functions.

Quadratic Programming Problems: Applications

- Least Square approximations and estimation
- Portfolio optimization
- Signal and image processing, computer vision, etc.
- Optimal control, linear model predictive control, etc
- PDE-constrained optimization problems in CFD, shape optimization, etc
- Sequential quadratic programming (SQP) methods for NLP
- etc.

Quadratic Programming Problems

Let us define a general quadratic programming (QP) problem as follows:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad q(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x}$$

$$\text{Subject to} \quad A^T\mathbf{x} \le \mathbf{b}$$

$$B^T\mathbf{x} = \mathbf{e}$$

$$\mathbf{x} \ge 0$$

$$\mathbf{x} = [x_1, x_2, ..., x_n]^T, \qquad \mathbf{c} = [c_1, c_2, ..., c_n]^T$$

$$\mathbf{b} = [b_1, b_2, ..., b_m]^T, \qquad \mathbf{e} = [e_1, e_2, ..., e_p]^T$$

$H = n \times n$ Hessian matrix, $A = n \times m$ Constant matrix, $B = n \times p$ Constant matrix

Quadratic Programming Problems

$$\text{Min } f(x) = -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2$$

$$s.t. \quad x_1 + x_2 = 2$$

$$x_1, x_2 \geq 0$$

$$\text{Min } f(x) = \begin{bmatrix} -6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$s.t. \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2$$

Compare with:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad q(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x}$$

$$\text{Subject to } A^T\mathbf{x} \leq \mathbf{b}$$

$$B^T\mathbf{x} = \mathbf{e}$$

$$\mathbf{x} \geq 0$$

$$c = \begin{bmatrix} -6 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$e = \begin{bmatrix} 2 \end{bmatrix}$$

Quadratic Programming Problems: KKT Conditions

$$\underset{\mathbf{x}}{\text{Minimize}} \quad q(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x}$$

$$\text{Subject to} \quad A^T\mathbf{x} \le \mathbf{b}$$

$$B^T\mathbf{x} = \mathbf{e}$$

$$\mathbf{x} \ge 0$$

Introduce slack variables **S** to transform inequalities to equalities.

$$A^T\mathbf{x} + \mathbf{S} = \mathbf{b},\ \mathbf{S} \ge \mathbf{0}$$

Define Lagrangian:

$$L = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{u}^T(A^T\mathbf{x} + \mathbf{S} - \mathbf{b}) + \mathbf{v}^T(B^T\mathbf{x} - \mathbf{e}) - \mathbf{w}^T\mathbf{x}$$

Note that the non-negativity constraint $x \ge 0$ has been written as $-x \le 0$.

Quadratic Programming Problems: KKT Conditions

$$\underset{\mathbf{x}}{\text{Minimize}} \quad q(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x}$$

Subject to $A^T\mathbf{x} \le \mathbf{b}$

$B^T\mathbf{x} = \mathbf{e}$

$\mathbf{x} \ge 0$

$$L = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{u}^T(A^T\mathbf{x} + \mathbf{S} - \mathbf{b}) + \mathbf{v}^T(B^T\mathbf{x} - \mathbf{e}) - \mathbf{w}^T\mathbf{x}$$

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{c} + H\mathbf{x} + A\mathbf{u} + B\mathbf{v} - \mathbf{w} = \mathbf{0}$$

$$A^T\mathbf{x} + \mathbf{S} - \mathbf{b} = 0$$

$$B^T\mathbf{x} - \mathbf{e} = 0$$

Complimentary slackness:

$$u_i S_i = 0,\ i = 1, \ldots, m; \qquad S_i, u_i \ge 0,\ i = 1, \ldots, m$$

$$w_i x_i = 0, i = 1, \ldots, n; \qquad w_i \ge 0,\ i = 1, \ldots, n$$

Solve these conditions for **x**, **u**, **v**, **S**, and **w**.

Transformation of KKT Conditions

Lagrangian multiplier for equality constraint is free in sign. Write it as: $\mathbf{v} = \mathbf{y} - \mathbf{z}$ with $\mathbf{y}, \mathbf{z} \geq 0$

We can define the following matrix and vectors:

$$N = \begin{bmatrix} H & A & -I_{(n)} & 0_{(n\times m)} & B & -B \\ A^T & 0_{(m\times m)} & 0_{(m\times n)} & I_{(n)} & 0_{(m\times n)} & 0_{(m\times n)} \\ B^T & 0_{(p\times m)} & 0_{(p\times n)} & 0_{(p\times m)} & 0_{(p\times p)} & 0_{(p\times p)} \end{bmatrix}_{[(n+m+p)\times(2n+2m+2p)]}$$

$$X = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{w} \\ \mathbf{S} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix}_{(2n+2m+2p)} \qquad D = \begin{bmatrix} \mathbf{-c} \\ \mathbf{b} \\ \mathbf{e} \end{bmatrix}_{(n+m+p)}$$

Now, the KKT conditions can be written as:

$$NX = D$$

Complimentary slackness conditions reduces to:

$$X_i X_{n+m+i} = 0,\ i = 1,\ldots..,(n+m) \quad \text{and} \quad X_i \geq 0,\ i = 1,\ldots..,(2n+2m+2p)$$

Note only complimentary slackness condition is nonlinear in variable X_i

The Simplex Method for Solving QP Problem

Except the complimentary slackness condition, all the equations in the transformed KKT conditions are linear. A solution of the linear equations in NX = D that satisfy the complimentary slackness condition and the non-negativity constraint, is a solution to the original QP problem.

The complimentary slackness condition is nonlinear in variable X_i, i = 1 to $(n+m)$. Thus, the Simplex procedure needs to be changed to accommodate this. A procedure developed by Wolfe (1959) and later refined by Hadley (1964) can be used and the Simplex method will converge in a finite number of steps provided H is a positive definite matrix. This method is based on the Phase I of the Two-Phase Simplex method.

Solution of NLP by Quadratic Approximation

Similar to sequential linear programming strategy, we can solve a general NLP by successively approximating it as a QP problem around the current estimate of the solution (x^0).

Note that a function $f(x)$ can be approximated by its quadratic approximation around x^0 as follows:

$$q(x; x^0) = f(x^0) + \nabla f(x^0)^T (x - x^0) + \frac{1}{2}(x - x^0)^T \nabla^2 f(x^0)(x - x^0)$$

Solution of NLP by Successive Quadratic Approximation: Algorithm

Step – 1: Formulate the QP problem

$$\text{Minimize} \quad \nabla f(x^{(t)})^T d + \frac{1}{2} d^T \nabla^2 f(x^{(t)}) d$$

$$\text{Subject to:} \quad g_j(x^{(t)}) + \nabla g_j(x^{(t)}) d \geq 0, \qquad j = 1, ..., J$$

$$h_k(x^{(t)}) + \nabla h_k(x^{(t)}) d = 0, \qquad k = 1, ..., K$$

$$x_i \geq 0, \qquad i = 1, ..., N$$

Step – 2: Solve the QP problem and set: $x^{(t+1)} = x^{(t)} + d$

Step – 3: Check for convergence. If converged, *Stop*. Else go to *Step* – 1

Solution of NLP by Successive Quadratic Programming: Example

Solve by successive QP strategy.

Minimize $f(x) = 6\frac{x_1}{x_2} + \frac{x_2}{x_1^2}$

Subject to: $g(x) = x_1 + x_2 - 1 \geq 0$

$h(x) = x_1 x_2 - 2 = 0$

Initial feasible estimate: $x^0 = (2, 0)$

$f(x^0) = 12.25,\ h(x^0) = 0,\ g(x^0) = 2 > 0$

$$\nabla f(x) = \begin{bmatrix} 6x_2^{-1} - 2x_2 x_1^{-3} \\ -6x_1 x_2^{-2} + x_1^{-2} \end{bmatrix}$$

$$\nabla^2 f(x) = \begin{bmatrix} 6x_2 x_1^{-4} & -6x_2^{-2} - 2x_1^{-3} \\ -6x_2^{-2} - 2x_1^{-3} & 12x_1 x_2^{-3} \end{bmatrix}$$

$$\nabla h(x) = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$$

Solution of NLP by Successive Quadratic Programming: Example

The first QP sub-problem will be:

$$\text{Minimize} \quad \left(\frac{23}{4}, \quad -\frac{47}{4}\right)d + \frac{1}{2}d^T \begin{bmatrix} \frac{3}{8} & -\frac{25}{4} \\ -\frac{25}{4} & 24 \end{bmatrix} d$$

$$\text{Subject to: } (1, \quad 1)d + 2 \geq 0$$

$$(1, \quad 2)d = 0$$

The second constraints can be use to write: $d_1 = -2d_2$

Now the solution of single variable problem can be obtained as:

$d^0 = (-0.9207, 0.4604)$

The new point becomes:

$x^{(1)} = x^0 + d^0 = (1.0793, 1.4604)$

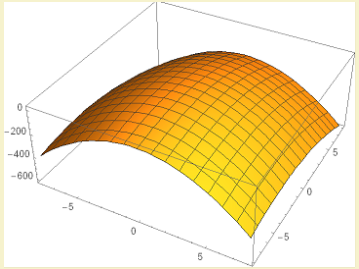
$f(x^{(1)}) = 5.687$

$h(x^{(1)}) = -0.424$

$g(x^{(1)}) > 0$

The objective function has improved. However, equality constraint is violated. We can continue the procedure and find the optimum with high accuracy. True optimum is: x^* = (1, 2), $f(x^*)$ = 5. We are already not very far in just one iteration.

Optimization in Chemical Engineering



Thank You

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

OPTIMIZATION IN CHEMICAL ENGINEERING

Constrained Nonlinear Programming

(WEEK – 10: LECTURE – 48)

Dr. Debasis Sarkar

Department of Chemical Engineering

Indian Institute of Technology Kharagpur

Constrained Nonlinear Programming

Week 10:

- ✓ Successive Linear Programming Method
- ✓ Successive Quadratic Programming Method
- Penalty Function Method
- Augmented Lagrange Multiplier Method

Today's Topic:

- Penalty Function Method

Constrained Nonlinear Programming: Techniques

Direct Methods

- Sequential linear programming method
- Sequential quadratic programming method

Indirect Methods

- Penalty function method
- Augmented Lagrange multiplier method

The Penalty Function Method and Augmented Lagrangian Multiplier Method are called "Transformation Methods" as these methods transform the original constrained problem to a sequence of unconstrained problems.

The motivation for the transformation method is that we can solve the constrained optimization problem using algorithms for unconstrained problems which can be solved efficiently and reliably.

Penalty Function Method

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

Penalty function methods are classified depending on how they handle inequality constraints. All transformation methods treat equality constraint in similar way.

Let $x^{(t)}$, t = 1, 2, 3, ….. approximate the stationary points of the sequence of associated unconstrained problems.

Interior Point Methods: The sequence $x^{(t)}$ contains feasible points. Such methods are also called Barrier Methods.

Exterior Point Methods: The sequence $x^{(t)}$ contains infeasible points.

Penalty Function Method

The penalty function method solves a nonlinear programming problem by transforming a constrained problem into a sequence of unconstrained problems.

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to} \quad g_j(\mathbf{x}) \le 0 \quad \text{for} \quad j = 1, 2, ..., p$$

$$h_k(\mathbf{x}) = 0 \quad \text{for} \quad k = 1, 2, ..., m$$

$$\text{Minimize} \quad P(x, R) = f(x) + \Omega(R, g(x), h(x))$$

where $P(x, R)$ is a penalty function, and R is a positive penalty parameter. Ω is known as penalty term. After the penalty function is formulated, it is minimized for a series of values of increasing R-values, which force the sequence of minima to approach the optimum of the constrained problem.

Penalty Function Method

The penalty function method solves a nonlinear programming problem by transforming a constrained problem into a sequence of unconstrained problems. Each time we modify a set of penalty parameters and start a sequence with the solution obtained in the previous sequence. At any sequence, the following problem is solved (say for minimization):

$$\text{Minimize} \quad P(x,R) = f(x) + \Omega(R, g(x), h(x))$$

The above unconstrained problem can be efficiently solved using any method for unconstrained minimization.

Penalty Function Method: Different Penalty Terms

Different penalty terms are used for equality and inequality constraints.

Parabolic Penalty for Equality Constraints: $\Omega = R\{h(x)\}^2$

The feasible points always satisfy $h(x) = 0$. Any infeasible point is penalized by an amount proportionate to the square of constraint violation. The extent of penalty is controlled by penalty parameter R. Initially, a small value of R is used and then it is increased gradually.

Since all infeasible points are penalized, this is an exterior penalty term.

Penalty Function Method: Different Penalty Terms

Infinite Barrier Penalty for Inequality Constraints: $\Omega = R\sum_{j\in\bar{J}}\left|g_j(x)\right|$

Here, the parameter R is a large number (say, 10^{20}). $\bar{J}$ represents the set of violated constraints at the current point.

$$g_j(x) > 0 \quad \text{for all} \quad j \in \bar{J}$$

Thus a penalty proportionate to the constraint violation is added to the objective function. Since all infeasible points are penalized, this is an exterior penalty term.

Penalty Function Method: Different Penalty Terms

Log Penalty for Inequality Constraints: $\Omega = R \ln\left[-g(x)\right]$

For infeasible points, $g(x) > 0$.

Thus, this penalty cannot assign penalty to infeasible points. More penalty is added to the points that are close to the constraint boundary [points with very small $g(x)$]. Here we start with large R and then gradually reduce the value of R.

Since all feasible points are penalize, this is an interior penalty term.

Penalty Function Method: Different Penalty Terms

Bracket Operator: Handles both Constraints:

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

$$\text{Where, } \langle g(x)\rangle = \max(0,\ g(x))$$

This is mostly used to handle inequality constraints.

Here we start with a small value of R and then gradually increase the value of R.

Since all infeasible points are penalized, this is an exterior penalty term.

Penalty Function Method: Different Penalty Terms

Inverse Penalty for Inequality Constraints:

$$\Omega = -R\frac{1}{g(x)}$$

Here we start with a large value of R and then gradually reduce the value of R.

Since all feasible points are penalized, this is an interior penalty term.

Penalty Function Method: Exterior Penalty

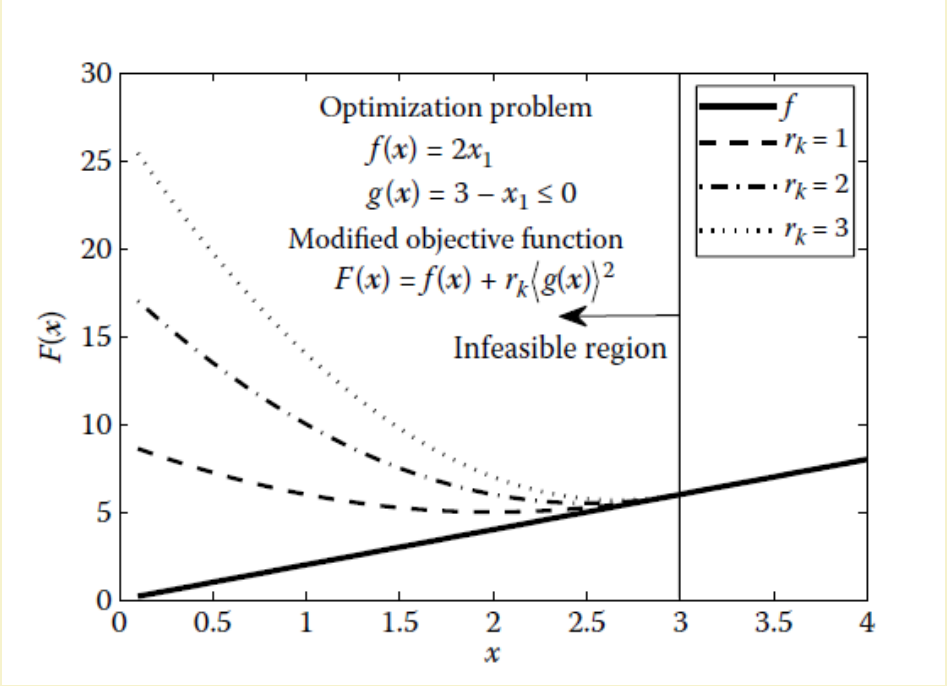
The modified objective function with penalty terms is written as

$$P(x,R) = f(x) + R\sum_{k=1}^{m}\{h_k(x)\}^2 + R\sum_{i=1}^{p}\langle g_i(x)\rangle^2 \qquad \text{where} \quad \langle g_i(x)\rangle = \max[0,\ g_i(x)]$$

In case constraints are satisfied ($g_i(x) \le 0$), $\langle g_i(x)\rangle$ will be zero and there will be no penalty on the objective function. In case constraints are violated ($g_i(x) > 0$), $\langle g_i(x)\rangle$ will be a positive value resulting in a penalty on the objective function. The penalty will be higher for higher infeasibility of the constraints.

The function $P(x, R)$ can be optimized using the algorithms for unconstrained problems. The penalty function method of this form is called the exterior penalty function methods.

Penalty Function Method: Exterior Penalty



Start the algorithm with a small $R = r_k$ and increase it to a larger value for the purpose of tightening the constraints at later iterations.

Penalty Function Method: Interior Penalty

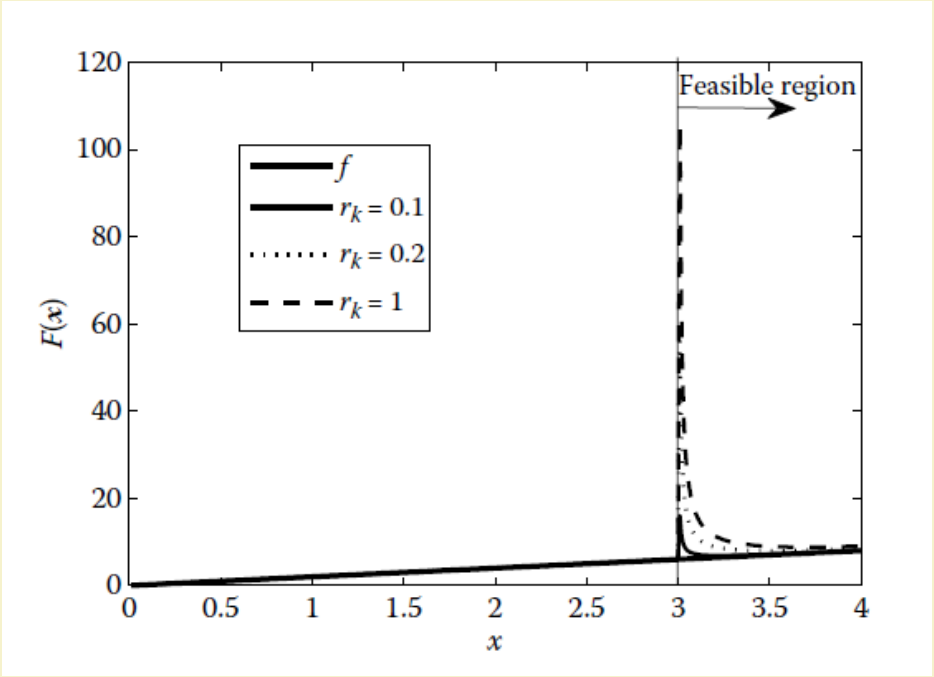
In the interior penalty function method, a feasible point is first selected. The objective function is modified in such a way that it does not leave the feasible boundary. They are therefore frequently referred to as barrier function methods.

The modified objective function in the interior penalty function approach would be

$$P(x,R) = f(x) - R\sum_{i=1}^{m}\frac{1}{g_i(x)}$$

This is known as inverse penalty for inequality constraints. This is also interior penalty.

Penalty Function Method: Interior Penalty



Here we start the algorithm with a large value of $R = r_k$ and increase it to a smaller value for the purpose of tightening the constraints at later iterations.

Penalty Function Method: Advantages and Disadvantages

Advantages:

- It can be started from an infeasible point.
- Unconstrained optimization methods can be directly used.

Disadvantages:

- The function becomes ill-conditioned as the value of the penalty terms is increased. The gradient value may become large and the algorithm may show divergence.
- This method does not satisfy the constraints exactly, it is not suitable for optimization problems where feasibility must be ensured in all iterations.

Penalty Function Method: Example

Minimize $f(x) = -x_1 x_2$

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

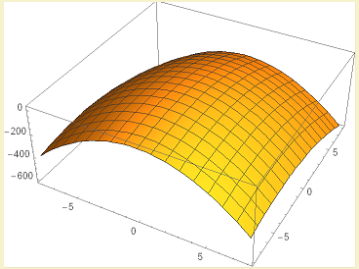
$g_2(x) = -x_1 - x_2 \leq 0$

Initial estimate: $x^0 = (1, 5)$

$$P(x, R) = f(x) + R\sum_{k=1}^{m}\{h_k(x)\}^2 + R\sum_{i=1}^{p}\langle g_i(x)\rangle^2$$

Set R = 1, 10, 100 etc in successive iterations.

Optimization in Chemical Engineering



Thank You

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

OPTIMIZATION IN CHEMICAL ENGINEERING

Constrained Nonlinear Programming

(WEEK – 10: LECTURE – 49)

Dr. Debasis Sarkar

Department of Chemical Engineering

Indian Institute of Technology Kharagpur

Constrained Nonlinear Programming

Week 10:

- ✓ Successive Linear Programming Method
- ✓ Successive Quadratic Programming Method
- ✓ Penalty Function Method
- Augmented Lagrange Multiplier Method

Today's Topic:

- Augmented Lagrange Multiplier Method

Constrained Nonlinear Programming: Techniques

Direct Methods

- Sequential linear programming method
- Sequential quadratic programming method

Indirect Methods

- Penalty function method
- Augmented Lagrange multiplier method

The Penalty Function Method and Augmented Lagrangian Multiplier Method are called "Transformation Methods" as these methods transform the original constrained problem to a sequence of unconstrained problems.

The motivation for the transformation method is that we can solve the constrained optimization problem using algorithms for unconstrained problems which can be solved efficiently and reliably.

Features of Penalty Function Method: Exterior Point

1. Applicable to both equality and inequality constrained problems.

2. The stating point can be infeasible.

3. The method iterates through the infeasible region. In this infeasible region, the objective function and the constraint functions may or may not be defined.

4. If the iterative process terminates prematurely, the final point may not be feasible and hence not usable.

Features of Penalty Function Method: Interior Point

1. Applicable to only inequality constrained problems.

2. The stating point must be feasible.

3. The method iterates through the feasible region. If the iterative process terminates prematurely, the final point will be feasible and hence usable.

Weakness of Penalty Function Method

1. The penalty function method tend to be ill-behaved near the boundary of the feasible set, where the optimum point usually lie.

2. The choice of the penalty parameter R at successive steps is not easy. Bad choice of the parameter R can seriously affect the computational effort.

3. With increased value of R, the Hessian matrix of the of the transformed unconstrained function becomes ill-defined as $R \rightarrow \infty$

Augmented Lagrangian Multiplier Method: Equality Constraints Only

Augmented Lagrangian Method attempt to alleviate some of the problems of the Penalty Function Method. In this method, there is no need for the Penalty Parameter R to go to infinity.

The augmented Lagrange multipliers method combines both Lagrange multipliers and penalty function methods. Let us first consider an optimization problem with only equality constraints.

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to} \quad h_k(\mathbf{x}) = 0 \quad \text{for} \quad k = 1, 2, ..., m$$

Augmented Lagrangian Multiplier Method: Equality Constraints Only

The Lagrangian function for the equality constrained problem is: $L(\mathbf{x}, \boldsymbol{\lambda}) = f(x) + \sum_{k=1}^{m} \lambda_k h_k(\mathbf{x})$

where λ_j , j = 1, 2, . . . , m, are the Lagrange multipliers.

The Augmented LagrangianFunction is defined by using exterior penalty function approach to define the new objective function AL(x,λ,R) as

$$AL(\mathbf{x}, \boldsymbol{\lambda}, R) = f(x) + \sum_{k=1}^{m} \lambda_k h_k(\mathbf{x}) + R \sum_{k=1}^{m} \{h_k(\mathbf{x})\}^2$$

where R is the penalty parameter. Note that the function AL reduces to the Lagrangian if R = 0 and to the the classical penalty function if all λ_j = 0.

Augmented Lagrangian Multiplier Method: Equality Constraints Only

Augmented Lagrangian Function:

$$AL(\mathbf{x}, \boldsymbol{\lambda}, R) = f(x) + \sum_{k=1}^{m} \lambda_k h_k(\mathbf{x}) + R \sum_{k=1}^{m} \{h_k(\mathbf{x})\}^2$$

If the Lagrange multipliers are fixed at their optimum values λ^*_j , the minimization of AL gives the solution of the original constrained problem in one step for any value of R.

Then there is no need to minimize the function AL for an increasing sequence of values of R. Since the values of λ^*_j are not known in advance, an iterative scheme is used to find the solution of the problem. In the first iteration (k = 1), the values of $\lambda^{(k)}_j$ are chosen as zero, the value of R is set equal to an arbitrary constant, and the function AL is minimized with respect to **x** to find $X^{*(k)}$. The values of $\lambda^{(k)}_j$ and R are then updated to start next iteration.

Augmented Lagrangian Multiplier Method: Equality Constraints Only

Update rules for λ_j : $\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2R_k h_j(\mathbf{x}^{(k)}), \quad j = 1, ..., m$

Here $\mathbf{x}^{(k)}$ is the starting vector used in the minimization of AL.

The value of R_k is updated as: $R_{k+1} = C\, R_k$ with $C > 1$

Augmented Lagrangian Multiplier Method: Inequality Constraints Only

Consider:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to } g_j(\mathbf{x}) \le 0 \quad \text{for } j = 1, 2, ..., p$$

Lagrangian function:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(x) + \sum_{j=1}^{p} \lambda_j \left[g_j(x) + \sigma_j^2 \right]^2$$

Note the slack variables

Augmented Lagrangian Function:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(x) + \sum_{j=1}^{p} \lambda_j \left[g_j(x) + \sigma_j^2 \right]^2 + \sum_{j=1}^{p} R \left[g_j(x) + \sigma_j^2 \right]^2$$

Augmented Lagrangian Multiplier Method: Inequality Constraints Only

Augmented Lagrangian Function: $L(\mathbf{x},\boldsymbol{\lambda}) = f(x) + \sum_{j=1}^{p} \lambda_j \left[g_j(x) + \sigma_j^2 \right]^2 + \sum_{j=1}^{p} R \left[g_j(x) + \sigma_j^2 \right]^2$

The above form is equivalent to:

$$L(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^{p} \lambda_j \alpha_j + R \sum_{j=1}^{p} \alpha_j^2$$

where, $\alpha_j = \max \left\{ g_j(\mathbf{x}), -\frac{\lambda j}{2R_k} \right\}$

Update rules for λ_j :

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2R_k h_j(\mathbf{x}^{(k)}), \qquad j = 1, \ldots\ldots, p$$

Augmented Lagrangian Multiplier Method: Both Equality and Inequality Constraints

Consider:

$$\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) \qquad \mathbf{x} = [x_1, x_2, ..., x_n]^T$$

$$\text{Subject to } g_j(\mathbf{x}) \le 0 \quad \text{for } j = 1, 2, ..., p$$

$$h_k(\mathbf{x}) = 0 \quad \text{for } k = 1, 2, ..., m$$

Augmented Lagrangian Function:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^{p} \lambda_j \alpha_j + \sum_{k=1}^{m} \lambda_{p+k} h_k(\mathbf{x}) + R \sum_{j=1}^{p} \alpha_j^2 + R \sum_{k=1}^{m} \{h_k(\mathbf{x})\}^2$$

where, $\alpha_j = \max\left\{ g_j(\mathbf{x}), -\frac{\lambda j}{2R_k} \right\}$

Augmented Lagrangian Multiplier Method: Both Equality and Inequality Constraints

Update rules for λ_j :

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2R_k \max\left\{ g_j(\mathbf{x}), -\frac{\lambda_j^{(k)}}{2R_k} \right\}, \qquad j = 1, ..., p$$

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + 2R_k h_j(\mathbf{x}^{(k)}), \qquad j = 1, ..., m$$

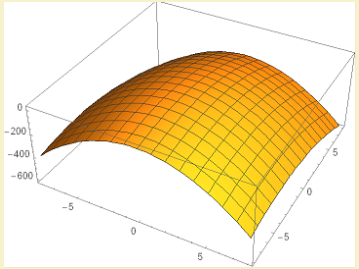
Augmented Lagrangian Multiplier Method: Advantages

1. The value of Penalty parameter R need not be increased to infinity for convergence.

2. The starting vector $x^{(1)}$ need not be feasible.

3. It is possible to achieve $g(x) = 0$ and $h(x) = 0$ precisely and non-zero values of Lagrange multipliers for the active constraints.

Optimization in Chemical Engineering



Thank You

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

OPTIMIZATION IN CHEMICAL ENGINEERING

Constrained Nonlinear Programming

(WEEK – 10: LECTURE – 50)

Dr. Debasis Sarkar

Department of Chemical Engineering

Indian Institute of Technology Kharagpur

Constrained Nonlinear Programming

Week 10:

- Successive Linear Programming Method
- Successive Quadratic Programming Method
- Penalty Function Method
- Augmented Lagrange Multiplier Method

Today's Topic:

- MATLAB Examples

MATLAB Example

MATLAB function `fmincon` solves a General Nonlinear Programming Problem as follows:

$$\underset{x}{\text{Minimize}} \quad f(x)$$

$$\text{Such that:} \quad c(x) \le 0$$

$$c_{eq}(x) = 0$$

$$A \cdot x \le b$$

$$A_{eq} = b_{eq}$$

$$LB \le x \le UB$$

We will solve NLP using MATLAB functions:

`linprog`

`quadprog`

`fminsearch`

Successive Linear Programming: Example

Solve the following NLP using SLP starting from (2, 1).

$$\text{Minimize} \quad f(x) = x_1^2 + x_2^2$$

$$\text{Subject to} \quad g(x) = x_1^2 - x_2 \geq 0$$

$$h(x) = 2 - x_1 - x_2^2 = 0$$

$$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$$

Let us go through the steps of SLP and solve the LPP at each step using MATLAB function `linprog`

MATLAB solves the following LPP:

$$\underset{x}{\text{Minimize}} \quad f^T x$$

$$\text{Such that: } A \cdot x \leq b$$

$$A_{eq} = b_{eq}$$

$$LB \leq x \leq UB$$

```
x = linprog(f,A,b,Aeq,beq,lb,ub)
```

Sequential Linear Programming: General NLP: Example

Minimize $f(x) = x_1^2 + x_2^2$

Subject to $g(x) = x_1^2 - x_2 \geq 0$

$h(x) = 2 - x_1 - x_2^2 = 0$

$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$

$$\nabla f = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \quad f(2,1) = 5 \qquad \nabla g = \begin{bmatrix} 2x_1 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}, \quad g(2,1) = 3$$

Now, $f\left(x^{(t)}\right) + \nabla f\left(x^{(t)}\right)\left(x - x^{(t)}\right)$

$$= 5 + \begin{bmatrix} 4 & 2 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix}$$

$$= 5 + 4(x_1 - 2) + 2(x_2 - 1)$$

Now, $g\left(x^{(t)}\right) + \nabla g\left(x^{(t)}\right)\left(x - x^{(t)}\right)$

$$= 3 + \begin{bmatrix} 4 & -1 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix}$$

$$= 3 + 4(x_1 - 2) - (x_2 - 1)$$

We want to construct the linear approximation to this problem at $x^0 = (2,1)$

$$\nabla h = \begin{bmatrix} -1 \\ -2x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \quad h(2,1) = -1$$

Now, $h\left(x^{(t)}\right) + \nabla h\left(x^{(t)}\right)\left(x - x^{(t)}\right) = -1 + \begin{bmatrix} -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix} = -1 - (x_1 - 2) - 2(x_2 - 1)$

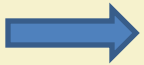
Sequential Linear Programming: General NLP: Example

Minimize $f(x) = x_1^2 + x_2^2$

Subject to $g(x) = x_1^2 - x_2 \geq 0$

$h(x) = 2 - x_1 - x_2^2 = 0$

$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$



Minimize $\tilde{f}(\bar{x}, x^0) = 5 + 4(x_1 - 2) + 2(x_2 - 1)$

Subject to $\tilde{g}(\bar{x}, x^0) = 3 + 4(x_1 - 2) - (x_2 - 1) \geq 0$

$\tilde{h}(\bar{x}, x^0) = -1 - (x_1 - 2) - 2(x_2 - 1) = 0$

$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$

MATLAB

```
f = [4  2];
A = [-4 1];
b = -4;
lb = [0.5 0];
ub = [2.5 3];
Aeq = [1 2];
beq = 3;
x = linprog(f,A,b,Aeq,beq,lb,ub);
```

The solution to this LPP is $x^{(1)} = \left(11/9, 8/9\right)$

$g\left(x^{(1)}\right) = 0.6049 > 0, \quad h\left(x^{(1)}\right) = -0.0123 \neq 0$

Therefore, we reinitialize the problem at $x^{(1)}$

Sequential Linear Programming: General NLP: Example

We reinitialize the problem at

$$x^{(1)} = \left(11/9, 8/9\right)$$

$$\nabla f = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 22/9 \\ 16/9 \end{bmatrix}, \quad f\left(11/9, 8/9\right) = 185/81$$

Now, $f\left(x^{(t)}\right) + \nabla f\left(x^{(t)}\right)\left(x - x^{(t)}\right)$

$$= 185/81 + \begin{bmatrix} 22/9 & 16/9 \end{bmatrix} \begin{bmatrix} x_1 - 11/9 \\ x_2 - 8/9 \end{bmatrix}$$

$$= 185/81 + 22/9\left(x_1 - 11/9\right) + 16/9\left(x_2 - 8/9\right)$$

$$\nabla g = \begin{bmatrix} 2x_1 \\ -1 \end{bmatrix} = \begin{bmatrix} 22/9 \\ -1 \end{bmatrix}$$

$$g\left(11/9, 8/9\right) = 49/81$$

Now, $g\left(x^{(t)}\right) + \nabla g\left(x^{(t)}\right)\left(x - x^{(t)}\right)$

$$= 49/81 + \begin{bmatrix} 22/9 & -1 \end{bmatrix} \begin{bmatrix} x_1 - 2 \\ x_2 - 1 \end{bmatrix}$$

$$= 49/81 + 22/9\left(x_1 - 11/9\right) - 1\left(x_2 - 8/9\right)$$

$$\nabla h = \begin{bmatrix} -1 \\ -2x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -16/9 \end{bmatrix},$$

$$h\left(11/9, 8/9\right) = -1/81$$

Now, $h\left(x^{(t)}\right) + \nabla h\left(x^{(t)}\right)\left(x - x^{(t)}\right)$

$$= -1/81 + \begin{bmatrix} -1 & -16/9 \end{bmatrix} \begin{bmatrix} x_1 - 11/9 \\ x_2 - 8/9 \end{bmatrix}$$

$$= -1/81 - 1\left(x_1 - 11/9\right) - 16/9\left(x_2 - 8/9\right)$$

Sequential Linear Programming: General NLP: Example

Minimize $\tilde{f}(\bar{x}, x^1) = 185/81 + 22/9\left(x_1 - 11/9\right) + 16/9\left(x_2 - 8/9\right)$

Subject to $\tilde{g}(\bar{x}, x^1) = 49/81 + 22/9\left(x_1 - 11/9\right) - 1\left(x_2 - 8/9\right) \geq 0$

$\tilde{h}(\bar{x}, x^1) = -1/81 - 1\left(x_1 - 11/9\right) - 16/9\left(x_2 - 8/9\right) = 0$

$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$

MATLAB

```
f = [22/9  16/9];
A = [-22/9 1];
b = -121/81;
lb = [0.5 0];
ub = [2.5 3];
Aeq = [1 16/9];
beq = 226/81;
x = linprog(f,A,b,Aeq,beq,lb,ub)
```

The solution to this LPP is $x^{(2)} = (1.0187, 0.9964)$

$g\left(x^{(2)}\right) = \quad 0.0414 > 0, \quad h\left(x^{(2)}\right) = -0.0116 \neq 0$

Therefore, we reinitialize the problem at $x^{(2)}$

Sequential Linear Programming: General NLP: Example

We reinitialize the problem at

$$x^{(2)} = (1.0187, 0.9964)$$

$$\nabla f = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 2.0375 \\ 1.9928 \end{bmatrix}, \; f(1.0187, 0.9964) = 2.0306$$

$$\text{Now, } f\left(x^{(t)}\right) + \nabla f\left(x^{(t)}\right)\left(x - x^{(t)}\right)$$

$$= 2.0306 + \begin{bmatrix} 2.0375 & 1.9928 \end{bmatrix} \begin{bmatrix} x_1 - 1.0187 \\ x_2 - 0.9964 \end{bmatrix}$$

$$= 2.0306 + 2.0375(x_1 - 1.0187) + 1.9928(x_2 - 0.9964)$$

$$\nabla g = \begin{bmatrix} 2x_1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2.0375 \\ -1 \end{bmatrix}, \; g(1.0187, 0.9964) = 0.0414$$

$$\text{Now, } g\left(x^{(t)}\right) + \nabla g\left(x^{(t)}\right)\left(x - x^{(t)}\right)$$

$$= 0.0414 + \begin{bmatrix} 2.0375 & -1 \end{bmatrix} \begin{bmatrix} x_1 - 1.0187 \\ x_2 - 0.9964 \end{bmatrix}$$

$$= 0.0414 + 2.0375(x_1 - 1.0187) - 1(x_2 - 0.9964)$$

$$\nabla h = \begin{bmatrix} -1 \\ -2x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -1.9929 \end{bmatrix}, \; h(1.0187, 0.9964) = -0.0116$$

$$\text{Now, } h\left(x^{(t)}\right) + \nabla h\left(x^{(t)}\right)\left(x - x^{(t)}\right)$$

$$= -0.0116 + \begin{bmatrix} -1 & -1.9928 \end{bmatrix} \begin{bmatrix} x_1 - 1.0187 \\ x_2 - 0.9964 \end{bmatrix}$$

$$= -0.0116 - 1(x_1 - 1.0187) - 1.9928(x_2 - 0.9964)$$

Sequential Linear Programming: General NLP: Example

$$\text{Minimize} \quad \tilde{f}(\bar{x}, x^2) = 2.0306 + 2.0375(x_1 - 1.0187) + 1.9928(x_2 - 0.9964)$$

$$\text{Subject to} \quad \tilde{g}(\bar{x}, x^2) = 0.0414 + 2.0375(x_1 - 1.0187) - 1(x_2 - 0.9964) \geq 0$$

$$\tilde{h}(\bar{x}, x^2) = -0.0116 - 1(x_1 - 1.0187) - 1.9928(x_2 - 0.9964) = 0$$

$$0.5 \leq x_1 \leq 2.5, \quad 0 \leq x_2 \leq 3$$

The solution to this LPP is $x^{(3)} = (1.0001,\ 0.9999)$

$$g\left(x^{(3)}\right) = 2.9632\text{e}-04, \quad h\left(x^{(3)}\right) = 1.0071\text{e}-04$$

MATLAB

```
f = [2.0375  1.9928];
A = [-2.0375 1];
b = -1.0378;
lb = [0.5 0];
ub = [2.5 3];
Aeq = [1 1.9928];
beq = 2.9927;
x = linprog(f,A,b,Aeq,beq,lb,ub);
```

Successive Quadratic Programming: Example

Solve the following NLP by SQP:

Minimize $f(x) = 6\frac{x_1}{x_2} + \frac{x_2}{x_1^2}$

Subject to $g(x) = x_1 + x_2 - 1 \geq 0$

$h(x) = x_1 x_2 - 2 = 0$

Initial feasible estimate $x^0 = (2,1)$

True solution: $x_1 = 1$, $x_2 = 2$, $f_{opt} = 5$

MATLAB solves the following QP:

$$\underset{x}{\text{Minimize}} \quad f^T x + \frac{1}{2} x^T H x$$

Such that: $A \cdot x \leq b$

$A_{eq} = b_{eq}$

$LB \leq x \leq UB$

```
x = quadprog(H,f,A,b,Aeq,beq)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub)
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)
```

Solution of NLP by Sequential Quadratic Programming: Example

Minimize $f(x) = 6\frac{x_1}{x_2} + \frac{x_2}{x_1^2}$

Subject to $g(x) = x_1 + x_2 - 1 \geq 0$

$h(x) = x_1 x_2 - 2 = 0$

Initial feasible estimate $x^0 = (2,1)$

$$f\left(x^0\right) = 12.25, \quad h\left(x^0\right) = 0, \quad g\left(x^0\right) = 2 > 0$$

$$\nabla f = \begin{bmatrix} 6x_2^{-1} - 2x_2 x_1^{-3} \\ -6x_1 x_2^{-2} + x_1^{-2} \end{bmatrix}$$

$$\nabla^2 f = \begin{bmatrix} 6x_2 x_1^{-4} & -6x_2^{-2} - 2x_1^{-3} \\ -6x_2^{-2} - 2x_1^{-3} & 12x_1 x_2^{-3} \end{bmatrix}$$

$$\nabla h(x) = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$$

Solution of NLP by Sequential Quadratic Programming: Example

The first QP sub-problem will be:

$$\text{Minimize} \quad \begin{bmatrix} \frac{23}{4} \\ -\frac{47}{4} \end{bmatrix}^T d + \frac{1}{2} d^T \begin{bmatrix} \frac{3}{8} & -\frac{25}{4} \\ -\frac{25}{4} & 24 \end{bmatrix}$$

$$\text{Subject to} \quad (1,1)d + 2 \geq 0$$

$$(1,2)d = 0$$

Write this on MATLAB command window (or write a script):

```
H = [3/8 -25/4; -25/4 24];
f = [23/4; -47/4];
A = [-1 -1];
b = 2;
Aeq = [1 2];
beq = 0;

d = quadprog(H,f,A,b,Aeq,beq)
```

Now the solution can be obtained as:

$$d^0 = (-0.9207, 0.4604)$$

Solution of NLP by Sequential Quadratic Programming: Example

The new point becomes:

$$x^{(1)} = x^0 + d^0 = (1.0793, 1.4604)$$

$$f\left(x^{(1)}\right) = 5.687$$

$$h\left(x^{(1)}\right) = -0.4238$$

$$g\left(x^{(1)}\right) = 1.5397$$

$$\text{Minimize} \quad \begin{bmatrix} 0.4362 \\ -2.1779 \end{bmatrix}^T d + \frac{1}{2} d^T \begin{bmatrix} 6.4574 & -4.4040 \\ -4.4040 & 4.1582 \end{bmatrix}$$

$$\text{Subject to} \quad (1,1)d + 1.5397 \geq 0$$

$$(1,2)d - 0.4238 = 0$$

Now the solution can be obtained as: $d^1 = (-0.0127, 0.2183)$

The new point becomes: $x^{(2)} = x^1 + d^1 = (1.0666,\ 1.6787)$

$$f\left(x^{(2)}\right) = 5.2878$$

$$h\left(x^{(2)}\right) = -0.2094$$

$$g\left(x^{(2)}\right) = 1.7454$$

```
d = quadprog(H,f,A,b,Aeq,beq)
```

Solution of NLP by Sequential Quadratic Programming: Example

$$\text{Minimize} \quad \begin{bmatrix} -0.5002 \\ -1.3920 \end{bmatrix}^T d + \frac{1}{2} d^T \begin{bmatrix} 7.7816 & -3.7772 \\ -3.7772 & 2.7055 \end{bmatrix}$$

$$\text{Subject to} \quad (1,1)d + 1.7454 \geq 0$$

$$(1,2)d - 0.2094 = 0$$

```
d = quadprog(H,f,A,b,Aeq,beq)
```

Now the solution can be obtained as: $d^2 = (0.0279, 0.0908)$

The new point becomes: $x^{(3)} = x^2 + d^2 = (\ 1.0945,\ 1.7695)$

$$f\left(x^{(3)}\right) = 5.1884$$

$$h\left(x^{(3)}\right) = -0.0633$$

$$g\left(x^{(3)}\right) = 1.8639$$

Penalty Function Method

Solve the following NLP by Penalty Function Method:

$$\text{Minimize} \quad f(x) = \left(x_1 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

$$\text{Subject to} \quad 26 - \left(x_1 - 5\right)^2 - x_2^2 \leq 0$$

$$x_1,\ x_2 \geq 0$$

Formulate the Penalty Function and solve the successive unconstrained functions using MATLAB function `fminsearch`

Solution of NLP using Penalty Function: Example

$$\text{Minimize} \quad f(x) = \left(x_1 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

$$\text{Subject to} \quad 26 - \left(x_1 - 5\right)^2 - x_2^2 \le 0$$

$$x_1,\ x_2 \ge 0$$

Penalty function $P = \left(x_1 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2 + \mathrm{R}\left\langle 26 - \left(x_1 - 5\right)^2 - x_2^2 \le 0 \right\rangle$

This means $\max\left[0,\ 26 - \left(x_1 - 5\right)^2 - x_2^2 \le 0\right]$

Solution of NLP using Penalty Function: Example

Take $R = 0.1$ and solve using MATLAB function `fminsearch`

Write a function file as follows:

```
function f = Penalty(x)
R = 0.1;
f = (x(1)+x(2)-11)^2 + (x(1)+x(2)^2-7)^2 + R*max(0,26-(x(1)-5)^2-x(2)^2);
```

Now write the following on MATLAB command window (or write a script file):

```
x0 = [0, 0];
options = optimset('Display','iter','TolFun',1e-8);
[x, fval, exitflag] = fminsearch(@Penalty,x0,options)
```

Solution: $x_1^* = 8.9005 \quad x_2^* = 0.4159$ $\qquad f\left(x_1^*, x_2^*\right) = 8.1951$ $\qquad \text{Constraint} = 10.6130$

Solution of NLP using Penalty Function: Example

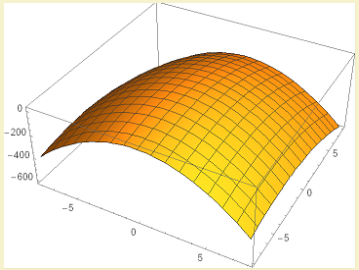
Take $R = 1$ and solve using MATLAB function `fminsearch`

Solution: $x_1^* = 10.0973 \quad x_2^* = 0.1333 \qquad f(x_1^*, x_2^*) = 10.2955 \qquad \text{Constraint} = 2.9818e-09$

Take $R = 10$ and solve using MATLAB function `fminsearch`

Solution: $x_1^* = 0.7759 \quad x_2^* = 2.8561 \qquad f(x_1^*, x_2^*) = 58.0243 \qquad \text{Constraint} = -4.0346e-09$

Optimization in Chemical Engineering



Thank You