Unit 14

Iterative Solution Algorithms for Nonlinear Optimization

14.1 Iterative Solution Algorithm for Constrained Nonlinear Optimization Problems

Let us remember that a **constrained NLPP** can be generically written as:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in C, \end{array}$$

where $C := \{x \in \mathbb{R}^n \ ; \ h_1(x) = 0, \ldots, h_p(x) = 0, g_1(x) \leq 0, \ldots, g_q(x) \leq 0\}$ is the feasible set, $h_1, \ldots, h_p : \mathbb{R}^n \to \mathbb{R}$ are the equality-constraint functions, $g_1, \ldots, g_q : \mathbb{R}^n \to \mathbb{R}$ are the less-than-or-equal-to-constraint functions, $p, q \in \{0, 1, 2, \ldots\}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function.

14.1.1 Multiplier Method

Example 14.1. Consider $f, h, g : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x_1, x_2) = (x_1 + 2)^2 + (x_2 - 3)^2$, $h(x_1, x_2) = x_1 + 2 \cdot x_2 - 4$ and $g(x_1, x_2) = -x_1$, respectively. Solve the problem by the **Multiplier Method** using:

1. a tolerance of 0.001;

2. $x^0 = (-1, -1, -1)$;

3. initial λ equal to $(1, 1)$;

4. initial ρ equal to $(1, 1)$;

5. the steepest descent search direction with the Armijo rule with $\sigma = 0.1$, $\beta = 0.1$ and $s = 1$;

And if a constraint is violated, multiply the penalty weight on that constraint by 3.

Solution. First, we convert the inequalities to equalities; thus, we have $h_1, h_2 : \mathbb{R}^3 \to \mathbb{R}$ defined by $h_1(x_1, x_2, x_3) = x_1 + 2 \cdot x_2 - 4$ and $h_2(x_1, x_2, x_3) = -x_1 + {x_3}^2$, respectively. Second, we denote the augmented Lagrangian function

$$A_{\lambda,\rho}(x) := f(x_1, x_2) + \lambda_1 \cdot h_1(x) + \lambda_2 \cdot h_2(x) + \rho_1 \cdot h_1(x)^2 + \rho_2 \cdot h_2(x)^2,$$

where $x := (x_1, x_2, x_3)$. Third, after conducting the following:

Algorithm 1: Multiplier method for problem stated in Example 14.1

$k \leftarrow 0$
$x^k := ({x^k}_1, {x^k}_2, {x^k}_3) \leftarrow (-1, -1, -1)$
$\lambda := (\lambda_1, \lambda_2) \leftarrow (1, 1)$
$\rho := (\rho_1, \rho_2) \leftarrow (1, 1)$
Update augmented Lagrangian function $A_{\lambda,\rho}$
$grad \leftarrow \nabla A_{\lambda,\rho}(x^k)$
while $grad \neq 0$ or $h({x^k}_1, {x^k}_2) \neq 0$ or $g({x^k}_1, {x^k}_2) > 0$ **do**
    $d^k \leftarrow -grad$
    Determine step size, α^k
    $x^{k+1} \leftarrow x^k + \alpha^k \cdot d^k$
    $k \leftarrow k + 1$
    for $j \leftarrow 1, 2$ **do**
        $\lambda_j \leftarrow \lambda_j + 2 \cdot \rho_j \cdot h_j(x^k)$
        if $h_j(x^k) \neq 0$ **then**
            $\rho_j \leftarrow 3 \cdot \rho_j$
        end
    end
    Update augmented Lagrangian function $A_{\lambda,\rho}$
    $grad \leftarrow \nabla A_{\lambda,\rho}(x^k)$
end

We have:

```
Iteration 0
x                       :       Matrix([[-1], [-1], [-1]])
f(x[:2])                :       17.0000000000000
direction               :       Matrix([[16.0000000000000], [34.0000000000000], [10.0000000000000]])
alpha                   :       0.1

Iteration 1
x                       :       Matrix([[0.600000000000000], [2.40000000000000], [0]])
f(x[:2])                :       7.12000000000000
direction               :       Matrix([[-21.2000000000000], [-23.2000000000000], [0]])
alpha                   :       0.010000000000000002

Iteration 2
x                       :       Matrix([[0.388000000000000], [2.16800000000000], [0]])
f(x[:2])                :       6.39476800000000
direction               :       Matrix([[-35.4640000000000], [-40.6880000000000], [0]])
alpha                   :       0.010000000000000002
```

Figure 14.1: Results of first iterations.

```
Iteration 123
x                       :       Matrix([[1.50493221790030e-10], [1.99999999984951], [0]])
f(x[:2])                :       5.00000000090296
direction               :       Matrix([[0.00328433096398357], [-0.00328433313472496], [0]])
alpha                   :       1.0000000000000005e-07

Iteration 124
x                       :       Matrix([[4.78926318188388e-10], [1.99999999952107], [0]])
f(x[:2])                :       5.00000000287356
direction               :       Matrix([[0.00156807579155078], [-0.00156807585644030], [0]])
alpha                   :       1.0000000000000005e-07

x                       :       Matrix([[6.35733897343466e-10], [1.99999999936427], [0]])
f(x[:2])                :       5.00000000381440
```

Figure 14.2: Results of last iterations.