Optimization Theory
MMC 52212 / MME 52106

Multivariable Optimization
Non Linear Optimization (Constrained)

by

Dr. Shibayan Sarkar
Department of Mechanical Engg.
Indian School of Mines Dhanbad
1. Initial estimate $x^{(0)}$ of the solution $x^*$ is to be assumed first. The initial estimate may or may not be feasible. A algorithm is to be developed which generate a sequence of points in $\mathbb{R}^N$ from $x^{(0)}$ to $x^{(T)}$, where $x^{(t)}$ is the generic point and $x^{(T)}$, the limit point, is the best estimate of $x^*$ produced by the algorithm.

2. The points $x^{(t)}$, $t = 1, 2, \ldots, T$, approximate stationary points of an associated unconstrained function called a penalty function.

3. The original constrained problem is transformed into a sequence of unconstrained problem via penalty function.
NLP structure

Penalty function methods are classified according to the procedure employed for handling inequality constraints, since essentially all transformation methods treat equality constraints in the same way. We will refer to these methods as *interior* or *exterior* point methods depending upon whether the sequence $x^{(n)}$ contains feasible or infeasible points, respectively. When the sequence of stationary points contains both feasible and infeasible points, we say the method is *mixed*.

Consider the *penalty function*

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

where $R$ is a set of penalty parameters and $\Omega$, the *penalty term*, is a function of $R$ and the constraint functions.

Equality constraints. In every case $\Omega$ is constructed so that solution of the sequence of associated *subproblems* produces a good estimate of the NLP. An interior point method is the result of selecting a form for $\Omega$ that will force stationary points of $P(x, R)$ to be feasible. Such methods are also called Barrier Method.
### Classification of Penalty Terms

| Parabolic Penalty: Discourages +ve or -ve variations of $h(x)$. | $\Omega = R \{h(x)\}^2$
With the increasing value of $R$, the stationary value of $P(x, R)$ will approach to $x^*$, since the limit as $R$ grows large $h(x(T)) = 0$. |
|---|---|
| **Infinite Barrier Penalty:** | $\Omega = 10^{20} \sum_{j \in \bar{J}} |g_j(x)|$
$g_j(x) < 0$ for all $j \in \bar{J}$ |
| **Log Penalty:** Barrier function, -ve penalty can be avoided by setting $\Omega = 0$ for $g(x) > 1$ | $\Omega = -R \ln[g(x)]$
This term produces +ve penalty for $0 < g(x) < 1.0$
This term produces -ve penalty for $1.0 < g(x) < \infty$ |
## Classification of Penalty Terms

<table>
<thead>
<tr>
<th>Inverse Penalty</th>
<th>[ \Omega = R \left[ \frac{1}{g(x)} \right] ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bracket Operator: Produces exterior penalty function.</td>
<td>[ \Omega = R \langle g(x) \rangle^2 ]</td>
</tr>
</tbody>
</table>

\[
\langle \alpha \rangle = \begin{cases} 
\alpha & \text{if } \alpha \leq 0 \\
0 & \text{if } \alpha > 0 
\end{cases}
\]
Penalty Function Algorithm

Step 1. Define \( N, J, K, \quad \varepsilon_3, x^{(0)}, \) and \( R^{(0)}, \)

\[ \varepsilon_3 = \text{penalty termination criterion} \]
\[ x^{(0)} = \text{initial estimate of } x^* \]
\[ R^{(0)} = \text{initial set of penalty parameters} \]

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

Step 3. Find \( x^{(r+1)} \) such that \( P(x^{(r+1)}, R^{(r)}) \rightarrow \min, \) with \( R^{(r)} \) fixed. Terminate, using \( \varepsilon_2, \) and use \( x^{(r)} \) to begin next search.

Step 4. Is \( |P(x^{(r+1)}, R^{(r)}) - P(x^{(r)}, R^{(r-1)})| \leq \varepsilon_3? \)

Yes: Set \( x^{(r+1)} = x^{(T)} \) and terminate
No: Continue.

Step 5. Choose \( R^{(r+1)} = R^{(r)} + \Delta R^{(r)} \) according to a prescribed update rule, and go to 2.
Penalty Function Algorithm: Example 1 (Parabolic)

Minimize \( f(x) = (x_1 - 4)^2 + (x_2 - 4)^2 \)

Subject to \( h(x) = x_1 + x_2 - 5 = 0 \)

Form penalty function using parabolic penalty term
\[
P(x, R) = (x_1 - 4)^2 + (x_2 - 4)^2 + R(x_1 + x_2 - 5)^2
\]

Investigate the stationary points of \( P(x, R) \) as a function of \( R \).

\[
\frac{\partial P}{\partial x_1} = 2(x_1 - 4) + 2R(x_1 + x_2 - 5) = 0
\]

\[
\frac{\partial P}{\partial x_2} = 2(x_2 - 4) + 2R(x_1 + x_2 - 5) = 0
\]

\[
x_1 = x_2 = \frac{5R + 4}{2R + 1}
\]

L-Hospital's rule produces following limit:
\[
\lim_{R \to \infty} \frac{5R + 4}{2R + 1} = \frac{5}{2} = 2.5
\]

\( R \) tends to weight the constraint function against the objective function. So, \( R \) grows large, stationary point of \( P(x, R) \) will tend to satisfy \( h(x) \) more closely and approach \( x^* \).

<table>
<thead>
<tr>
<th>( R )</th>
<th>( x_1^{(t)} = x_2^{(t)} )</th>
<th>( f(x^{(t)}) )</th>
<th>( h(x^{(t)}) )</th>
<th>( P(x^{(t)}, R) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.0000</td>
<td>0.0000</td>
<td>3.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>3.7500</td>
<td>0.1250</td>
<td>2.5000</td>
<td>0.7500</td>
</tr>
<tr>
<td>1</td>
<td>3.0000</td>
<td>2.0000</td>
<td>1.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>10</td>
<td>2.5714</td>
<td>4.0818</td>
<td>0.1428</td>
<td>4.2857</td>
</tr>
<tr>
<td>100</td>
<td>2.5075</td>
<td>4.4551</td>
<td>0.0150</td>
<td>4.4776</td>
</tr>
<tr>
<td>( \infty )</td>
<td>2.5000</td>
<td>4.5000</td>
<td>0.0000</td>
<td>4.5000</td>
</tr>
</tbody>
</table>

\( \varepsilon_3 \)
Penalty Function Algorithm: Example 1 (Parabolic)

\[ f(x) = (x_1 - 4)^2 + (x_2 - 4)^2 \]
\[ h(x) = x_1 + x_2 - 5 = 0 \]

<table>
<thead>
<tr>
<th>( R )</th>
<th>( x_1^{(t)} )</th>
<th>( f(x^{(t)}) )</th>
<th>( h(x^{(t)}) )</th>
<th>( P(x^{(t)}, R) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.0000</td>
<td>0.0000</td>
<td>3.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>3.7500</td>
<td>0.1250</td>
<td>2.5000</td>
<td>0.7500</td>
</tr>
<tr>
<td>1</td>
<td>3.0000</td>
<td>2.0000</td>
<td>1.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>10</td>
<td>2.5714</td>
<td>4.0818</td>
<td>0.1428</td>
<td>4.2857</td>
</tr>
<tr>
<td>100</td>
<td>2.5075</td>
<td>4.4551</td>
<td>0.0150</td>
<td>4.4776</td>
</tr>
<tr>
<td>( \infty )</td>
<td>2.5000</td>
<td>4.5000</td>
<td>0.0000</td>
<td>( \varepsilon_3 )</td>
</tr>
</tbody>
</table>
Penalty Function Algorithm: Example 2 (Bracket Operator)

Minimize \( f(x) = (x_1 - 4)^2 + (x_2 - 4)^2 \)

Subject to \( g(x) = 5 - x_1 - x_2 \geq 0 \)

Form penalty function using parabolic penalty term:
\[
P(x, R) = (x_1 - 4)^2 + (x_2 - 4)^2 + R(5 - x_1 - x_2)^2
\]

Investigate the stationary points of \( P(x, R) \) as a function of \( R \).

\[
\frac{\partial P}{\partial x_1} = 2(x_1 - 4) + 2R(5 - x_1 - x_2)(-1) = 0
\]

\[
\frac{\partial P}{\partial x_2} = 2(x_2 - 4) + 2R(5 - x_1 - x_2)(-1) = 0
\]

\[
x_1 = \frac{5R + 4}{2R + 1} = x_2
\]

L-Hospital’s rule produces following limit:

\[
\lim_{R \to \infty} \frac{5R + 4}{2R + 1} = \frac{5}{2} = 2.5
\]

R tends to weight the constraint function against the objective function. So, R grows large, stationary point of \( P(x, R) \) will tend to satisfy \( h(x) \) more closely and approach \( x^* \).

### Table

<table>
<thead>
<tr>
<th>( R )</th>
<th>( x_1^{(n)} = x_2^{(n)} )</th>
<th>( f(x^{(n)}) )</th>
<th>( h(x^{(n)}) )</th>
<th>( P(x^{(n)}, R) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.0000</td>
<td>0.0000</td>
<td>3.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.1</td>
<td>3.7500</td>
<td>0.1250</td>
<td>2.5000</td>
<td>0.7500</td>
</tr>
<tr>
<td>1</td>
<td>3.0000</td>
<td>2.0000</td>
<td>1.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>10</td>
<td>2.5714</td>
<td>4.0818</td>
<td>0.1428</td>
<td>4.2857</td>
</tr>
<tr>
<td>100</td>
<td>2.5075</td>
<td>4.4551</td>
<td>0.0150</td>
<td>4.4776</td>
</tr>
<tr>
<td>( \infty )</td>
<td>2.5000</td>
<td>4.5000</td>
<td>0.0000</td>
<td>4.5000</td>
</tr>
</tbody>
</table>
Penalty Function Algorithm: Example 3 (Log Penalty)

Minimize \[ f(x) = (x_1 - 4)^2 + (x_2 - 4)^2 \]

Subject to \[ g(x) = 5 - x_1 - x_2 \geq 0 \]

Form penalty function using parabolic penalty term
\[ P(x, R) = (x_1 - 4)^2 - R \ln(5 - x_1 - x_2) \]

Investigate the stationary points of \( P(x, R) \) as a function of \( R \).

\[ \frac{\partial P}{\partial x_1} = 2(x_1 - 4) + R \left[ \frac{1}{5 - x_1 - x_2} \right] = 0 \]

\[ \frac{\partial P}{\partial x_2} = 2(x_2 - 4) + R \left[ \frac{1}{5 - x_2 - x_2} \right] = 0 \]

\[ x_1 = x_2 \quad 2(x_1 - 4) + \left[ \frac{R}{5 - 2x_1} \right] = 0 \]

\[ 2x_1^2 - 13x_1 + 20 - \frac{R}{2} = 0 \]

\[ x_1 = \frac{13}{4} - \frac{1}{4} \sqrt{9 + 4R} \]

L-Hospital's rule produces following limit:

\[ \lim_{R \to 0} x_1 = 2.5 \quad \text{R begins to large and incremented to zero} \]

\[ f(x^{(T)}) = 4.5 \]

\[ \Omega = -R \ln[g(x)] \]

\[ 0 < g(x) < 1.0 \]
\[ 1.0 < g(x) < \infty \]

<table>
<thead>
<tr>
<th>( R )</th>
<th>( x_1 = x_2 )</th>
<th>( f(x) )</th>
<th>( g(x) )</th>
<th>(-R \ln(g(x)))</th>
<th>( P(x, R) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>-1.8059</td>
<td>67.4170</td>
<td>8.6118</td>
<td>-215.3133</td>
<td>-147.8963</td>
</tr>
<tr>
<td>10</td>
<td>1.5000</td>
<td>12.5000</td>
<td>2.0000</td>
<td>-6.9315</td>
<td>5.5685</td>
</tr>
<tr>
<td>1</td>
<td>2.3486</td>
<td>5.4542</td>
<td>0.3028</td>
<td>+1.1947</td>
<td>6.6489</td>
</tr>
<tr>
<td>0.1</td>
<td>2.4835</td>
<td>4.5995</td>
<td>0.0034</td>
<td>+0.3411</td>
<td>4.9406</td>
</tr>
<tr>
<td>0.01</td>
<td>2.4983</td>
<td>4.5100</td>
<td>0.0034</td>
<td>+0.0568</td>
<td>4.5668</td>
</tr>
<tr>
<td>0</td>
<td>2.5000</td>
<td>4.5000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>4.5000</td>
</tr>
</tbody>
</table>
Penalty Function Algorithm: Example 4 (Inverse Penalty)

Minimize \( f(x) = (x_1 - 4)^2 + (x_2 - 4)^2 \)

Subject to \( g(x) = 5 - x_1 - x_2 \geq 0 \)

Form penalty function using parabolic penalty term

\[
P(x, R) = (x_1 - 4)^2 + (x_2 - 4)^2 + R \left[ \frac{1}{5 - x_1 - x_2} \right]
\]

Investigate the stationary points of \( P(x, R) \) as a function of \( R \).

\[
\frac{\partial P}{\partial x_1} = 2(x_1 - 4) + \left[ \frac{R}{(5 - x_1 - x_2)^2} \right] = 0
\]

\[
\frac{\partial P}{\partial x_2} = 2(x_2 - 4) + \left[ \frac{R}{(5 - x_1 - x_2)^2} \right] = 0
\]

\[
x_1 = x_2 \quad 4x_1^3 - 36x_1^2 + 105x_1 - 100 + \frac{R}{2} = 0
\]

L-Hospital's rule produces following limit:

\[
\lim_{R \to \infty} x_1 = 2.5
\]

\[
f(x^{(T)}) = 4.5
\]

<table>
<thead>
<tr>
<th>( R )</th>
<th>( x_1 = x_2 )</th>
<th>( f(x) )</th>
<th>( g(x) )</th>
<th>( R/g(x) )</th>
<th>( P(x, R) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.5864</td>
<td>23.3053</td>
<td>3.8272</td>
<td>26.1288</td>
<td>49.4341</td>
</tr>
<tr>
<td>10</td>
<td>1.7540</td>
<td>10.0890</td>
<td>1.4920</td>
<td>6.7024</td>
<td>16.7914</td>
</tr>
<tr>
<td>1</td>
<td>2.2340</td>
<td>6.2375</td>
<td>0.5320</td>
<td>1.8797</td>
<td>8.1172</td>
</tr>
<tr>
<td>0.1</td>
<td>2.4113</td>
<td>5.0479</td>
<td>0.1774</td>
<td>0.5637</td>
<td>5.6116</td>
</tr>
<tr>
<td>0.01</td>
<td>2.4714</td>
<td>4.6732</td>
<td>0.0572</td>
<td>0.1748</td>
<td>4.8480</td>
</tr>
<tr>
<td>0.001</td>
<td>2.4909</td>
<td>4.5548</td>
<td>0.0182</td>
<td>0.0549</td>
<td>4.6097</td>
</tr>
<tr>
<td>0</td>
<td>2.5000</td>
<td>4.5000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>4.5000</td>
</tr>
</tbody>
</table>
Solving NLP(Constrained) in MATLAB

Nonlinear Inequality Constraints

This example shows how to solve a scalar minimization problem with nonlinear inequality constraints. The problem is to find $x$ that solves

$$
\min_{x} f(x) = e^{x_1 \left( 4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1 \right)}.
$$

subject to the constraints

$$
x_1x_2 - x_1 - x_2 \leq -1.5,
\quad x_1x_2 \geq -10.
$$

Because neither of the constraints is linear, you cannot pass the constraints to `fmincon` at the command line. Instead you can create a second file, `confun.m`, that returns the value at both constraints at the current $x$ in a vector $c$. The constrained optimizer, `fmincon`, is then invoked. Because `fmincon` expects the constraints to be written in the form $c(x) \leq 0$, you must rewrite your constraints in the form

$$
\begin{align*}
x_1x_2 - x_1 - x_2 + 1.5 &\leq 0, \\
-x_1x_2 -10 &\leq 0.
\end{align*}
$$
1. Start the Optimization app by typing `optimtool` at the command line.

2. The default **Solver fmincon - Constrained nonlinear minimization** is selected. This solver is appropriate for this problem, since Rosenbrock's function is nonlinear, and the problem has a constraint. Select **Derivatives**: Approximated by solver.

3. In the Algorithm pop-up menu choose **Interior point**, which is the default.

4. For **Objective function** enter `@rosenbrock`. The `@` character indicates that this is a function handle of the file `rosenbrock.m`.

5. For **Start point** enter `[0 0]`. This is the initial point where `fmincon` begins its search for a minimum.

6. For **Nonlinear constraint function** enter `@unitdisk`, the function handle of `unitdisk.m`.

7. In the Options pane (center bottom), select **iterative** in the **Level of display** pop-up menu. (If you don't see the option, click **Display to command window**.) This shows the progress of `fmincon` in the command window.

To visualize the progress, click different options of plotting.....
Choosing a Solver

Optimization Decision Table
The following table is designed to help you choose a solver. It does not address multiobjective optimization or equation solving. There are more details on all the solvers in Problems Handled by Optimization Toolbox Functions.

Use the table as follows:

Identify your objective function as one of five types:
- Linear
- Quadratic
- Sum-of-squares (Least squares)
- Smooth nonlinear
- Nonsmooth

Identify your constraints as one of five types:
- None (unconstrained)
- Bound
- Linear (including bound)
- General smooth
- Discrete (integer)

Use the table to identify a relevant solver

<table>
<thead>
<tr>
<th>Objective Function</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>None</td>
</tr>
<tr>
<td>Quadratic</td>
<td>Bound</td>
</tr>
<tr>
<td>Sum-of-squares</td>
<td>Linear</td>
</tr>
<tr>
<td>Smooth nonlinear</td>
<td>General</td>
</tr>
<tr>
<td>Nonsmooth</td>
<td>Discrete</td>
</tr>
</tbody>
</table>

Next page
<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Least Squares</th>
<th>Smooth nonlinear</th>
<th>Non smooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>n/a (f = const, or min = −∞)</td>
<td>quadprog, Information</td>
<td>lsqcurvefit, Information, lsqnonlin, Information</td>
<td>fminsearch, fminunc, Information</td>
<td>fminsearch, *</td>
</tr>
<tr>
<td>Bound</td>
<td>linprog, Information</td>
<td>quadprog, Information</td>
<td>lsqcurvefit, lsqnonlin, lsqnonneg, Information</td>
<td>fminbnd, fmincon, fseminf, Information</td>
<td>fminbnd, *</td>
</tr>
<tr>
<td>Linear</td>
<td>linprog, Information</td>
<td>quadprog, Information</td>
<td>lsqlin, Information</td>
<td>fmincon, fseminf, Information</td>
<td>*</td>
</tr>
<tr>
<td>General smooth</td>
<td>fmincon, Information</td>
<td>fmincon, Information</td>
<td>fmincon, Information</td>
<td>fmincon, fseminf, Information</td>
<td>*</td>
</tr>
<tr>
<td>Discrete</td>
<td>intlinprog, Information</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

This table does not list multiobjective solvers nor equation solvers.