Optimization with Constraints

• Standard formulation

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{such that} & \quad h_i(x) = 0, \quad i = 1, \ldots, n_c, \\
& \quad g_j(x) \geq 0, \quad j = 1, \ldots, n_g.
\end{align*}
\] (5.1)

• Why are constraints a problem?
  – Canyons in design space
  – Equality constraints are particularly bad!
Treatment by derivative-based optimizers

- Following constraint boundaries: Gradient projection (Section 5.5)
- Steering away from constraints (Feasible directions, Section 5.6)
- Using penalties to convert to unconstrained problem (Section 5.7)
- Combining penalty with Lagrange multipliers (Section 5.8)
- Projected Lagrangian methods (Section 5.9)
Gradient projection and reduced gradient methods

• Find good direction in space tangent to active constraints

• Move a distance and then restore to constraint boundaries

*Figure 5.5.1 Projection and restoration moves.*
The Feasible Directions method

- Compromise constraint avoidance and objective reduction

\[ \text{maximize} \quad \beta \]
\[ \text{such that} \quad -s^T \nabla g_j + \theta_j \beta \leq 0, \quad j \in I_A, \]
\[ s^T \nabla f + \beta \leq 0, \quad \theta_j \geq 0, \]
\[ |s_0| \leq 1. \]

Figure 5.6.1 Selection of search direction using the feasible directions method.
Penalty function methods

- Quadratic penalty function

\[
\text{minimize } f(x) \\
\text{such that } h_i(x) = 0, \quad i = 1, \ldots, n_e, \\
g_j(x) \geq 0, \quad j = 1, \ldots, n_g, \\
\]

is replaced by

\[
\text{minimize } \phi(x, r) = f(x) + r \sum_{i=1}^{n_e} h_i^2(x) + r \sum_{j=1}^{n_g} (-g_j)^2 \\
r = r_1, r_2, \ldots, \quad r_i \to \infty, \\
\]

- Gradual rise of penalty parameter leads to sequence of unconstrained minimization technique (SUMT). Why is it important?
Example 5.7.1

Minimize \( f = x_1^2 + 10x_2^2 \)

Such that \( x_1 + x_2 = 4 \)

Augmented function \( \phi = x_1^2 + 10x_2^2 + r(4 - x_1 - x_2)^2 \)

Solution \( x_1 = \frac{40r}{10 + 11r} \quad x_2 = \frac{4r}{10 + 11r} \)

<table>
<thead>
<tr>
<th>( r )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f )</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.905</td>
<td>0.1905</td>
<td>3.992</td>
<td>7.619</td>
</tr>
<tr>
<td>10</td>
<td>3.333</td>
<td>0.3333</td>
<td>12.220</td>
<td>13.333</td>
</tr>
<tr>
<td>100</td>
<td>3.604</td>
<td>0.3604</td>
<td>14.288</td>
<td>14.144</td>
</tr>
<tr>
<td>1000</td>
<td>3.633</td>
<td>0.3633</td>
<td>14.518</td>
<td>14.532</td>
</tr>
</tbody>
</table>
Contours for $r=1$
Contours for r=10 at 12.5:2.5:75
Contours for $r=100$ at [15:5:150]
Contours for $r=1000$ at [15:5:150]
Multiplier methods

- By adding the Lagrange multipliers to penalty term can avoid ill-conditioning associated with high penalty

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{such that} & \quad h_j(x) = 0, \quad j = 1, \ldots, n_e.
\end{align*}
\tag{5.8.1}
\]

The augmented Lagrangian function

\[
\mathcal{L}(x, \lambda, r) = f(x) - \sum_{j=1}^{n_e} \lambda_j h_j(x) + r \sum_{j=1}^{n_e} h_j^2(x). \tag{5.8.2}
\]
Can estimate Lagrange multipliers

• Stationarity
\[ \frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial f}{\partial x_i} - \sum_{j=1}^{n_c} (\lambda_j - 2r h_j) \frac{\partial h_j}{\partial x_i} = 0, \]

• Without penalty
\[ \frac{\partial f}{\partial x_i} - \sum_{j=1}^{n_c} \lambda^*_j \frac{\partial h_j}{\partial x_i} = 0. \]

• So:
\[ \lambda_j - 2r h_j \rightarrow \lambda^*_j, \]

• Iterate
\[ \lambda_j^{(k+1)} = \lambda_j^{(k)} - 2r^{(k)} h_j^{(k)}, \]

See example in textbook
5.9: Projected Lagrangian methods

- Sequential quadratic programming

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{such that} & \quad g_j(x) \geq 0, \quad j = 1, \ldots, n_g .
\end{align*}
\] (5.9.1)

- Convert to

\[
\begin{align*}
\text{minimize} & \quad \phi(s) = f(x_i) + s^T g(x_i) + \frac{1}{2} s^T A(x_i, \lambda_i)s \\
\text{such that} & \quad g_j(x_i) + s^T \nabla g_j(x_i) \geq 0, \quad j = 1, \ldots, n_g ,
\end{align*}
\]

\[x_{i+1} = x_i + \alpha s ,\]

- Find alpha by minimizing

\[\psi(\alpha) = f(x) + \sum_{j=1}^{n_g} \mu_j | \min(0, g_j(x)) | ,\]
Matlab function fmincon

FMINCON attempts to solve problems of the form:

\[
\begin{align*}
\min \ F(X) \ & \text{subject to: } A \cdot X & \leq B, \ Aeq \cdot X = Beq \ \text{(linear cons)} \\
X & \leq C(X) \leq 0, \ Ceq(X) = 0 \ & \text{(nonlinear cons)} \\
LB & \leq X \leq UB
\end{align*}
\]

\[X=FMINCON(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON)\] subjects the minimization to the constraints defined in NONLCON. The function NONLCON accepts \(X\) and returns the vectors \(C\) and \(Ceq\), representing the nonlinear inequalities and equalities respectively. (Set \(LB=[]\) and/or \(UB=[]\) if no bounds exist.)

\[X,FVAL]=FMINCON(FUN,X0,...)\] returns the value of the objective function \(FUN\) at the solution \(X\).
Quadratic function and constraint function $f=\text{quad2}(x)$

Global $a$

$f=x(1)^2+a*x(2)^2$;

function $[c,\text{ceq}]=\text{ring}(x)$

global $r_i$ $\rho$

c(1)=$r_i^2-x(1)^2-x(2)^2$;

c(2)=$x(1)^2+x(2)^2-\rho^2$;

ceq=[];

$x_0=[1,10]$; $a=10$; $r_1=10.$; $r_2=20$;

[x,fval]=fmincon(@quad2,x0,[],[],[],[],[],[],@ring(x))

$x =10.0000$  -0.0000  \hspace{1cm} fval =100.0000

Min $f = x_1^2 + ax_2^2 \hspace{1cm} a > 1$

s.t $r_i^2 \leq x_1^2 + x_2^2 \leq \rho^2$
Making it harder for fmincon

- >> a=1.1;
- [x,fval]=fmincon(@quad2,x0,[],[],[],[],[],[],@ring)
- Warning: Trust-region-reflective method does not currently solve this type of problem,
- using active-set (line search) instead.
- > In fmincon at 437
- Maximum number of function evaluations exceeded;
- increase OPTIONS.MaxFunEvals.
- x =4.6355  8.9430
  fval =109.4628
Restart sometimes helps

```matlab
>> x0 = x
x0 = 4.6355  8.9430
>> [x,fval] = fmincon(@quad2,x0,[],[],[],[],[],[],@ring)

• x = 10.0000  0.0000
• fval = 100.0000
```