The Barrier Method

Lecture 15, Continuous Optimisation
Oxford University Computing Laboratory, HT 2006
Notes by Dr Raphael Hauser (hauser@comlab.ox.ac.uk)

Today we are going to learn two more methods for the general nonlinear programming problem

$$\text{(NLP)} \qquad \min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad g_{\mathcal{E}}(x) = 0, \quad g_{\mathcal{I}}(x) \geq 0.$$

In the previous two Lectures we learned that nonlinear constraints can be dealt with by incorporating a term forcing asymptotic feasibility into the objective function, which is then called merit function.

So far we used quadratic penalty terms to construct merit functions.

The *barrier method* makes another choice,

$$P(x, \mu) = f(x) - \mu \sum_{j \in \mathcal{I}} \ln g_j(x) + \frac{1}{2\mu} \sum_{i \in \mathcal{E}} g_i^2(x),$$

where $\mu > 0$ is a homotopy parameter.

- Equality constraints are again enforced using quadratic penalty terms that become gradually more stringent. The penalty terms are defined for all x.

- Inequality constraints are managed via the *barrier term*

$$-\mu \sum_{j \in \mathcal{I}} \ln g_j(x)$$

  which is only defined when all the $g_j(x)$ are strictly positive.

Definition 1: A point $x \in \mathbb{R}^n$ is *admissible* for (NLP) if all inequality constraints are satisfied.

The point is called *strictly admissible* if all inequality constraints are strictly satisfied, that is,

$$g_j(x) > 0 \qquad (j = \mathcal{I})$$

holds.

Note that admissible points may violate some or all of the equality constraints.

The sets of admissible and strictly admissible points respectively are called *admissible* and *strictly admissible domain*.

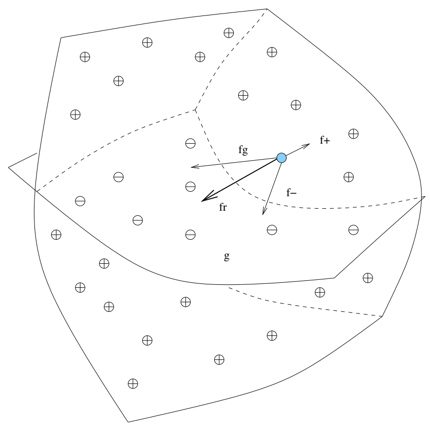
The barrier term is thus *only defined for strictly admissible points*. We can extend it outside the admissible domain by the convention

$$P(x,\mu) = \begin{cases} P(x,\mu) & (x \text{ admissible}), \\ +\infty & (x \text{ inadmissible}). \end{cases}$$

Note also that if $(x_k)_{\mathbb{N}}$ is a sequence of admissible points such that $x_k \to x^*$, where x^* is on the boundary of the admissible domain, then there exists $j \in \mathcal{I}$ such that $g_j(x_k) \to g_j(x^*) = 0$, and then

$$\lim_{k\to\infty} P(x_k,\mu) = +\infty.$$

A mechanistic interpretation ...



I. The Primal Barrier Method

We are ready to formulate a first algorithm based on the merit function introduced above.

Algorithm (PBM): Primal Barrier.

S0 Initialisation

choose $\mu_{-1} > \mu_0 > 0$, $\epsilon_0 > 0$;

choose x_0 strictly admissible;

S1 For $k = 0, 1, 2, \ldots$ repeat

solve $D^2_{xx}P(x_k,\mu_{k-1})\dot{x} + \frac{\partial}{\partial\mu}\nabla_x P(x_k,\mu_{k-1}) = 0$;

set $y^{[0]} := x_k + (\mu_k - \mu_{k-1})\dot{x}$; $\quad l := 0$;

until $\|\nabla_x P(y^{[l]},\mu_k)\| \leq \epsilon_k$ repeat

find $y^{[l+1]}$ s.t. $P(y^{[l+1]},\mu_k) < P(y^{[l]},\mu_k)$; $\quad l \leftarrow l+1$;

end (inner loop)

$x_{k+1} := y^{[l]}$; choose $\mu_{k+1} < \mu_k$, $\epsilon_{k+1} < \epsilon_k$;

end (outer loop)

Finding an Admissible Starting Point:

In step S0 we need to find a strictly admissible point x_0.

This is not a problem, because we can use the same algorithm to solve the *phase I* (or *auxiliary*) *problem*

$$(\text{AUX}) \quad \min_{(x,t)} t \quad \text{s.t. } g_j(x) + t \geq 0, \qquad (j \in \mathcal{I}).$$

A strictly admissible solution of (AUX) is readily available,

$$x_0 \in \mathbb{R}^n \quad \text{arbitrary},$$
$$t_0 := -\min\{g_j(x_0) : j \in \mathcal{I}\} + 1.$$

Convergence of the Algorithm:

The termination criterion in the inner loop guarantees that

$$\nabla_x P(x_k, \mu_{k-1}) = \nabla f(x_k) - \sum_{j \in \mathcal{I}} \frac{\mu_{k-1}}{g_j(x_k)} \nabla g_j(x_k) + \sum_{i \in \mathcal{E}} \frac{g_i(x_k)}{\mu_{k-1}} \nabla g_i(x_k) = O(\epsilon_{k-1}). \quad (1)$$

Arguments similar to those used Lecture 13 show the following convergence result.

Proposition 1: (NLP) has strictly admissible points if and only if the optimal solution (x^*, t^*) of (AUX) satisfies $t^* < 0$. $\square$

Note that if $t^* < 0$ then $t_k < 0$ after finite time k, and then x_k is strictly admissible for (NLP).

If $t^* > 0$ then this yields a certificate of *inadmissibility* for (NLP).

Theorem 1: Let $x^* = \lim_{l \to \infty} x_{k_l}$ be an accumulation point of the sequence $(x_k)_{\mathbb{N}_0}$ generated by Algorithm (PBM) and such that $\{\nabla g_i(x^*) : i \in \mathcal{A}(x^*)\}$ is linearly independent. Then

i) x^* is feasible,

ii) the LICQ holds at x^*,

iii) the limit $\lambda^* = \lim_{l \to \infty} \lambda^{[k_l]}$ exists, where

$$\lambda_i^{[k]} = \begin{cases} \frac{\mu_{k-1}}{g_i(x_k)}, & (i \in \mathcal{I}) \\ -\frac{g_i(x_k)}{\mu_{k-1}} & (i \in \mathcal{E}), \end{cases} \qquad (2)$$

iv) (x^*, λ^*) is a KKT point of (NLP).

Selection of the Starting Point in the Inner Loop:

Algorithm (PBM) determines a starting point $y^{[0]}$ for the inner loop in a somewhat intricate way. Why not use $y^{[0]} = x_k$ instead?

When a Newton step is applied to the unconstrained subproblem

$$\min_{y\in\mathbb{R}^n} P(y,\mu_k)$$

at $y^{[0]} := x_k$, then

$$g_j(x_k+\Delta_x) \approx g_j(x_k) + \nabla g_j^{\mathsf{T}}(x_k)\Delta_x \approx \left(2-\frac{\mu_{k-1}}{\mu_k}\right) g_j(x_k). \quad (3)$$

But note that $2-\mu_{k-1}/\mu_k < 0$ for $\mu_k < \mu_{k-1}/2$.

This shows that only a modest reduction of μ_k is possible in each iteration, because otherwise the Newton step takes the iterate outside of the admissible domain.

The choice $y^{[0]} = x_k$ thus leads to poor convergence of the algorithm.

On the other hand, the choice made by Algorithm (PBM),

$$y^{[0]} := x_k + (\mu_k - \mu_{k-1})\dot{x},$$

can be shown to behave much better.

Let us now give an hand-waving argument for why (3) is true.

Suppose we apply a Newton-Raphson update to the starting point $y^{[0]} = x_k$. The update $\Delta_x \in \mathbb{R}^n$ satisfies the system

$$D^2_{xx}P(x_k,\mu_k)\Delta_x = -\nabla_x P(x_k,\mu_k). \quad (4)$$

We have

$$\begin{aligned} D^2_{xx}P(x_k,\mu_k) = & \\ = & \left[D^2 f(x_k) - \frac{\mu_k}{\mu_{k-1}}\sum_{j\in\mathcal{I}}\left(\frac{\mu_{k-1}}{g_j(x_k)}\right)D^2 g_i(x_k) - \frac{\mu_{k-1}}{\mu_k}\sum_{i\in\mathcal{E}}\left(-\frac{g_i(x_k)}{\mu_{k-1}}\right)D^2 g_i(x_k)\right] \\ & + \left[\sum_{j\in\mathcal{I}}\frac{\mu_k}{g_j^2(x_k)}\nabla g_j(x_k)\nabla g_j(x_k)^{\mathsf{T}} + \frac{1}{\mu_k}\sum_{i\in\mathcal{E}}\nabla g_i(x_k)\nabla g_i(x_k)^{\mathsf{T}}\right] \\ = & \; C(x_k,\mu_k) + B(x_k,\mu_k). \end{aligned}$$

Now

$$\lambda_i^* \approx \lambda_i^{[k]} = \begin{cases} \frac{\mu_{k-1}}{g_i(x_k)}, & (i\in\mathcal{I}) \\ -\frac{g_i(x_k)}{\mu_{k-1}} & (i\in\mathcal{E}) \end{cases}$$

implies

$$\begin{aligned} \|C(x_k,\mu_k)\| &= O(1), \\ \|B(x_k,\mu_k)\| &= O\left(\max\left(\mu_k^{-1}, \max_i\left(g_i(x_k)^{-1}\right)\right)\right). \end{aligned}$$

This means that the Newton system is ill-conditioned whenever

$$\operatorname{rank}\Big(B(x^*,0)\Big) := \operatorname{rank}\Big(\lim_{l\to\infty} B(x_{k_l},\mu_{k_l})\Big) \notin \{0,n\},$$

and then we have

$$
\begin{aligned}
&\sum_{j\in\mathcal{A}(x^*)\cap\mathcal{I}} \frac{\mu_k \nabla g_j^{\mathsf{T}}(x_k)\Delta_x}{g_j^2(x_k)}\nabla g_j(x_k) + \sum_{i\in\mathcal{E}} \frac{\nabla g_i^{\mathsf{T}}(x_k)\Delta_x}{\mu_k}\nabla g_i(x_k)\\
&\approx B(x_k,\mu_k)\Delta_x\\
&\approx D^2_{xx}P(x_k,\mu_k)\Delta_x\\
&= -\nabla_x P(x_k,\mu_k)\\
&\approx -\nabla f(x_k) + \sum_{j\in\mathcal{A}(x^*)\cap\mathcal{I}} \frac{\mu_k}{g_j(x_k)}\nabla g_j(x_k) - \sum_{i\in\mathcal{E}} \frac{g_i(x_k)}{\mu_k}\nabla g_i(x_k)\\
&\overset{(1)}{\approx} \sum_{j\in\mathcal{A}(x^*)\cap\mathcal{I}} \frac{\mu_k-\mu_{k-1}}{g_j(x_k)}\nabla g_j(x_k) + \sum_{i\in\mathcal{E}} \frac{(\mu_k-\mu_{k-1})g_i(x_k)}{\mu_k\mu_{k-1}}\nabla g_i(x_k).
\end{aligned}
$$

Since the LICQ holds at x^*, this implies that

$$\nabla g_j^{\mathsf{T}}(x_k)\Delta_x \approx \left(1-\frac{\mu_{k-1}}{\mu_k}\right) g_j(x_k), \qquad (j\in\mathcal{I}\cap\mathcal{A}(x^*)),$$

and hence,

$$g_j(x_k+\Delta_x) \approx g_j(x_k) + \nabla g_j^{\mathsf{T}}(x_k)\Delta_x \approx \left(2-\frac{\mu_{k-1}}{\mu_k}\right) g_j(x_k),$$

as claimed in (3).

Note that the ill-conditioned Newton system also creates numerical problems similar to those encountered in the quadratic penalty function method.

Although we are not going to show this here, the choice

$$y^{[0]} := x_k + (\mu_k - \mu_{k-1})\dot{x}$$

resolves these problems.

The Primal-Dual Barrier Method:

The bad scaling of the primal barrier method can be overcome by exploiting the fact that Lagrange multiplier estimates become available as μ is decreased. This leads to the *primal-dual barrier method* which we will describe next.

The relationship between the primal barrier method and the primal-dual barrier method is in some ways similar to the relationship between the quadratic penalty function method and the augmented Lagrangian method of Lectures 13 and 14.

Recall the KKT conditions for problem (NLP),

$$
\begin{aligned}
\nabla f(x) - \sum_i \lambda_i \nabla g_i(x) &= 0 && (5)\\
g_{\mathcal{E}}(x) &= 0 && (6)\\
\lambda_j g_j(x) &= 0 \qquad (j\in\mathcal{I}) && (7)\\
\lambda_{\mathcal{I}}, g_{\mathcal{I}}(x) &\geq 0. && (8)
\end{aligned}
$$

Motivation of the primal-dual barrier method:

- Guarantee that (8) holds through the application of line searches to prevent iterates to become inadmissible.
- Perturb the right hand side of the complementarity equations (7) by μ.

Therefore, in each iteration of the algorithm one or several damped Newton steps are applied to the nonlinear system of equations

$$
\begin{aligned}
\nabla f(x) - \sum_i \lambda_i \nabla g_i(x) &= 0 \\
g_{\mathcal{E}}(x) &= 0 \\
\lambda_j g_j(x) &= \mu. \qquad (j \in \mathcal{I})
\end{aligned}
\tag{9}
$$

The corresponding Newton system is

$$
\begin{array}{rlcl}
\left(D^2 f(x) - \sum_i \lambda_i D^2 g_i(x)\right)\Delta_x & -\left(g'_{\mathcal{I}\cup\mathcal{E}}(x)\right)^{\mathsf{T}}\Delta_\lambda & = & -\nabla f(x) + \left(g'_{\mathcal{I}\cup\mathcal{E}}(x)\right)^{\mathsf{T}}\lambda \\
g'_{\mathcal{E}}(x)\Delta_x & & = & -g_{\mathcal{E}}(x) \\
\mathrm{Diag}(\lambda) g'_{\mathcal{I}}(x)\Delta_x & +\mathrm{Diag}(g_{\mathcal{I}}(x))\Delta_\lambda & = & \begin{bmatrix} \mu \\ \vdots \\ \mu \end{bmatrix} - \mathrm{Diag}(\lambda) g_{\mathcal{I}}(x)
\end{array}
\tag{10}
$$

where $\mathrm{Diag}(v)$ denotes a diagonal matrix with a vector v on its diagonal.

Algorithm (PDBM): Primal-Dual Barrier Method.

S0 Initialisation: choose $\mu_{-1} > \mu_0 > 0$, $\epsilon_0 > 0$, $\theta \in (0,1)$, x_0 strictly admissible, $\lambda^{[0]}$ s.t. $\lambda^{[0]}_{\mathcal{I}} \geq 0$;

S1 For $k = 0, 1, 2, \ldots$ repeat

set $y^{[0]} := x_k$, $\eta^{[0]} := \lambda^{[k]}$, $l := 0$;

until $\|\Delta_x, \Delta_\lambda\| \leq \epsilon_k$ repeat

solve (10) for $(\Delta_x, \Delta_\lambda)$ with $(x, \lambda, \mu) := (y^{[l]}, \eta^{[l]}, \mu_k)$;

set $\alpha_{\max} := \max\{\alpha \geq 0 : g_{\mathcal{I}}(y^{[l]} + \alpha\Delta_x) \geq 0, (\eta^{[l]} + \alpha\Delta_\lambda)_{\mathcal{I}} \geq 0\}$;

choose $\alpha_l \in (0, \max\{1, 0.98 \times \alpha_{\max}\}]$;

$(y^{[l+1]}, \eta^{[l+1]}) = (y^{[l]}, \eta^{[l]}) + \alpha_l(\Delta_x, \Delta_\lambda)$;

$l \leftarrow l + 1$;

end (inner loop)

$(x_{k+1}, \lambda^{[k+1]}) := (y^{[l]}, \eta^{[l]})$;

$\mu_{k+1} = \theta\mu_k$;

choose $\epsilon_{k+1} \in (0, \epsilon_k)$;

end (outer loop)

Note that we initialised the starting vector for the inner loop by $y^{[0]} = x_k$. In contrast to the primal barrier method, the primal-dual barrier method works fine with this choice, as a somewhat intricate analysis shows.

In Lecture 16 we will analyse the primal-dual barrier method for linear programming in further detail.

Reading Assignment: Lecture-Note 15.