Interior point methods; penalty methods; augmented Lagrangian; SQP; Choosing/devising the right solver.

Interior-point method

Interior-point methods (also referred to as **barrier methods** or **IPMs**) are a certain class of algorithms that solve linear and nonlinear convex optimization problems. An interior point method was discovered in 1967 and reinvented in the 80s. Still in the 80s, a nother methed was developed, a method for linear programming called Karmarkar's algorithm, which runs in provably polynomial time and is also very efficient in practice. It enabled solutions of linear programming problems that were beyond the capabilities of the simplex method. Contrary to the simplex method, it reaches a best solution by traversing the interior of the feasible region. The method can be generalized to convex programming based on a self-concordant barrier function used to encode the convex set. Any convex optimization problem can be transformed into minimizing (or maximizing) a linear function over a convex set by converting to the epigraph form. Karmarkar's breakthrough revitalized the study of interior-point methods and barrier problems, showing that it was possible to create an algorithm for linear programming characterized by polynomial complexity and, moreover, that was competitive with the simplex method. The class of primal-dual path-following interior-point methods is considered the most successful. Mehrotra's predictor–corrector algorithm provides the basis for most implementations of this class of methods.

Primal-dual interior-point method for nonlinear optimization

The primal-dual method's idea is easy to demonstrate for constrained nonlinear optimization. For simplicity, consider the all-inequality version of a nonlinear optimization problem:

$$\text{minimize } f(x) \text{ subject to } c_i(x) \geq 0 \text{ for } i = 1, \ldots, m,\ x \in \mathbb{R}^n, \text{ where } f : \mathbb{R}^n \to \mathbb{R}, c_i : \mathbb{R}^n \to \mathbb{R} \quad (1).$$

This inequality-constrained optimization problem is then solved by converting it into an unconstrained objective function whose minimum we hope to find efficiently. Specifically, the logarithmic barrier function associated with (1) is

$$B(x, \mu) = f(x) - \mu \sum_{i=1}^{m} \log(c_i(x)). \quad (2)$$

Here μ is a small positive scalar, sometimes called the "barrier parameter". As μ converges to zero the minimum of $B(x, \mu)$ should converge to a solution of (1).

The barrier function gradient is

$$g_b(x, \mu) := \nabla B(x, \mu) = g(x) - \mu \sum_{i=1}^{m} \frac{1}{c_i(x)} \nabla c_i(x), \quad (3)$$

where $g(x) := \nabla f(x)$ is the gradient of the original function $f(x)$, and ∇c_i is the gradient of c_i.

In addition to the original ("primal") variable x we introduce a Lagrange multiplier-inspired dual variable $\lambda \in \mathbb{R}^m$

$$c_i(x)\lambda_i = \mu, \forall i = 1, \ldots, m. \quad (4)$$

(4) is sometimes called the "perturbed complementarity" condition, for its resemblance to "complementary slackness" in KKT conditions.

We try to find those (x_μ, λ_μ) for which the gradient of the barrier function is zero.

Applying (4) to (3), we get an equation for the gradient:

$$g - A^T \lambda = 0, \quad (5)$$

where the matrix A is the Jacobian of the constraints $c(x)$.

The intuition behind (5) is that the gradient of $f(x)$ should lie in the subspace spanned by the constraints' gradients. The "perturbed complementarity" with small μ (4) can be understood as the condition that the solution should either lie near the boundary $c_i(x) = 0$, or that the projection of the gradient g on the constraint component $c_i(x)$ normal should be almost zero.

Applying Newton's method to (4) and (5), we get an equation for (x, λ) update (p_x, p_λ):

$$\begin{pmatrix} W & -A^T \\ \Lambda A & C \end{pmatrix} \begin{pmatrix} p_x \\ p_\lambda \end{pmatrix} = \begin{pmatrix} -g + A^T \lambda \\ \mu 1 - C\lambda \end{pmatrix},$$

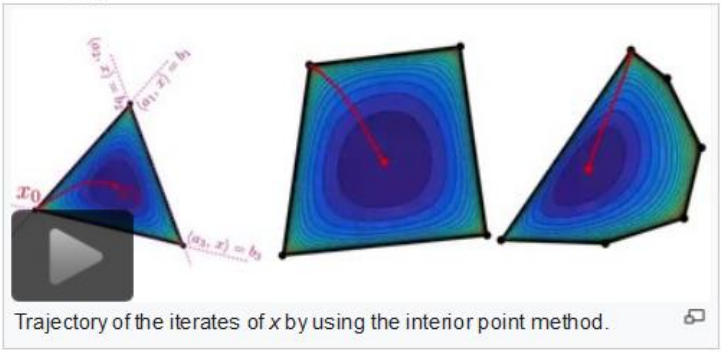
where W is the Hessian matrix of $B(x, \mu)$, Λ is a diagonal matrix of λ, and C is a diagonal matrix with $C_{ii} = c_i(x)$.

Because of (1), (4) the condition

$$\lambda \geq 0$$

should be enforced at each step. This can be done by choosing appropriate α:

$$(x, \lambda) \to (x + \alpha p_x, \lambda + \alpha p_\lambda).$$



Trajectory of the iterates of x by using the interior point method.

Penalty Methods

Problem Setup

Many times we have the constrained optmization problem (**P**):

$$\min_{x \in \mathcal{S}} f(x)$$

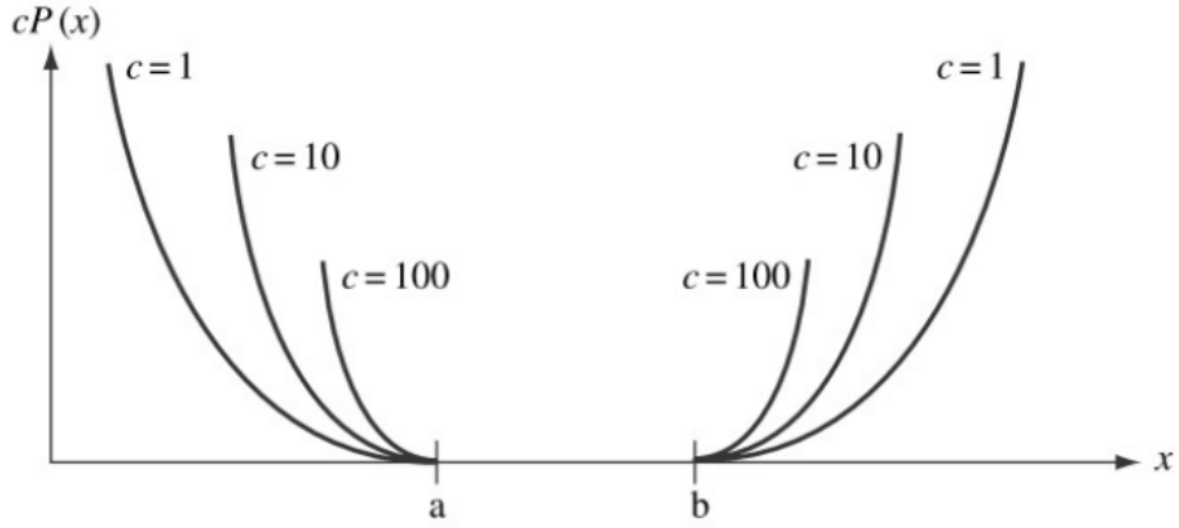
where $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and $\mathcal{S}$ is a constraint set in $\mathbb{R}^n$.

We introduce the **Penalty program**, (**P**(**c**)), the unconstrained problem:

$$\min_{x \in \mathbb{R}^n} f(x) + cp(x)$$

where $c > 0$ and $p : \mathbb{R}^n \to \mathbb{R}$ is the **penalty function** where $p(x) \geq 0 \; \forall \; x \in \mathbb{R}^n$, and $p(x) = 0$ iff $x \in S$.

Intuitively, the **penalty term** is used to give a high cost for violation of the constraints.



Inequality and Equality Constraints

For example, if we are given a set of inequality constraints (i.e. $\mathcal{S} = \{x : g_i(x) \leq 0, i = 1, 2, \ldots, m\}$), a useful penalty function could be $p(x) = \frac{1}{2} \sum_{i=1}^{m} (\max[0, g_i(x)])^2$. That is, if we satisfy the constraint, we don't take any penalty. Otherwise we take a squared penalty. Depending on c, we weight this penalty in $(P(c))$. For equality constraints we can rewrite them as inequality constraints and use them as above. That is, rewrite $h_j(x) = 0$ as two inequality constraints, $h_j(x) \leq 0$ and $-h_j(x) \leq 0$.

For large c, the minimum point of a problem $(P(c))$ is in a region where the penalty p is small. In fact, we will prove below that as $c \to \infty$, the solution of the penalty problem $(P(c))$ will converge to a solution of the constrained problem (P).

Penalty Method Lemmas

Let $0 < c_1 < c_2 < \ldots < c_k < c_{k+1} < \ldots \to \infty$ be our penalty parameter. Let $q(c, k) := f(x) + cp(x)$ be our penalty program. Also, let $x_k = \arg\min_x q(c_k, x) = \arg\min_x f(x) + c_k p(x)$.

With this notation, we will prove the following for **penalty lemmas**:

1. $q(c_k, x_k) \leq q(c_{k+1}, x_{k+1})$
2. $p(x_k) \geq p(x_{k+1})$
3. $f(x_k) \leq f(x_{k+1})$
4. $f(x^*) \geq q(c_k, x_k) \geq f(x_k)$

Below, we provide proofs of each of the above lemmas.

Lemma 16.1 $q(c_k, x_k) \leq q(c_{k+1}, x_{k+1})$

Proof:

$$\begin{aligned} q(c_{k+1}, x_{k+1}) &= f(x_{k+1}) + c_{k+1}p(x_{k+1}) \\ &\geq f(x_{k+1}) + c_k p(x_{k+1}) && (\because c_{k+1} > c_k > 0) \\ &\geq f(x_k) + c_k p(x_{k+1}) && (\because x_k \text{ is the minimizer of } q(c_k, x)) \\ \boxed{\therefore q(c_{k+1}, x_{k+1}) \geq q(c_k, x_k)} && (\because q(c, x_{k+1} = f(x_k) + c_k p(x_{k+1})) \end{aligned}$$

Lemma 16.2 $p(x_k) \geq p(x_{k+1})$

Proof:

$$f(x_k) + c_k p(x_k) \leq f(x_{k+1} + c_k p(x_{k+1}) \quad (\because x_k \text{ is the minimizer of } q(c_k, x)) \tag{16.1}$$

$$f(x_{k+1}) + c_{k+1}p(x_{k+1}) \leq f(x_k) + c_{k+1}p(x_k) \quad (\because x_{k+1} \text{ is the minimizer of } q(c_{k+1}, x)) \tag{16.2}$$

Adding Equation 16.1 and Equation 16.2 together, we get

$$\begin{aligned} & c_k p(x_k) + c_{k_+} p(x_{k+1}) \leq c_k p(x_{k+1}) + c_{k+1} p(x_k) \\ \Rightarrow\ & (c_{k+1} - c_k) p(x_{k+1}) \leq (c_{k+1} - c_k) p(x) \\ & \boxed{\therefore p(x_{k+1}) \leq p(x_k)} \qquad (\because c_{k+1} > c_k \Rightarrow c_{k+1} - c_k > 0) \end{aligned}$$

Lemma 16.3 $f(x_k) \leq f(x_{k+1})$

Proof:

$$\begin{aligned} f(x_{k+1}) + c_k p(x_{k+1}) &\geq f(x_k) + c_k p(x_k) && (\because x_k \text{ is the minimizer of } q(c_k, x)) \\ &\geq f(x_k) + c_k p(x_{k+1}) && (\because \text{Lemma } 16.2) \\ \boxed{\therefore f(x_{k+1}) \geq f(x_k)} \end{aligned}$$

Lemma 16.4 *Let x^* be the optimal value of our original constrained problem (P) with constraint set $\mathcal{S}$. Then, $f(x^*) \geq q(c_{k+1}, x_{k+1}) \geq f(x_k)\ \forall k$.*

Proof:

$$\begin{aligned} f(x^*) &= f(x^*) + c_k p(x^*) && (\because x^* \in \mathcal{S} \Rightarrow p(x^*) = 0) \\ &\geq f(x_k) + c_k p(x_k) \geq f(x_k) && (\because x_k \text{ is the minimizer of } q(c_k, x),\ \text{and } c_k > 0,\ p(x_k) \geq 0) \end{aligned}$$

$$\boxed{\therefore f(x^*) \geq q(c_{k+1}, x_{k+1}) \geq f(x_k)\ \forall k}$$

Convergence of the Penalty Method

Using the lemmas developed in Section 16.2, we prove the Penalty convergence theorem.

Theorem 16.5 *Suppose f, g, p are continuous functions. Let $x_k = \arg\min_x f(x) + c_k p(x)$ for a penalty function $p(x)$ as defined in subsection 16.1.1. Let $0 < c_1 < c_2 < \ldots < c_k < c_{k+1} < \ldots \to \infty$. Let $\bar{x}$ be an arbitrary limit point of $\{x_k\}_{k=1}^{\infty}$.*

Then, $\bar{x}$ solves (P) where (P) is the original constrained problem $\min_x f(x)$ s.t. $g(x) \leq 0$.

Proof: The limit point is defined as $\bar{x} = \lim_{k \in \mathcal{K}} x_k$.

Since f is given as continuous, then $\lim_{k \in \mathcal{K}} f(x_k) = f(\bar{x})$. We then get,

$$\begin{aligned} q^* := \lim_{x \in \mathcal{K}} q(c_k, x_k) \leq f(x^*) \qquad (\because \text{Lemma } 16.4) \\ \Rightarrow q^* = \lim_{x \in \mathcal{K}} f(x_k) + \lim_{x \in \mathcal{K}} c_k p(x_k) \leq f(x^*) \\ \Rightarrow q^* = f(\bar{x}) + \lim_{x \in \mathcal{K}} c_k p(x_k) \leq f(x^*) \\ \Rightarrow q^* - f(\bar{x}) = \lim_{x \in \mathcal{K}} c_k p(x_k) \leq f(x^*) \end{aligned}$$

Since $q^* - f(\bar{x})$ and $f(x^*)$ are finite which means $\lim_{x \in \mathcal{K}} c_k p(x_k)$ has to be a finite quantity. Since we know that $c_k \to \infty$, $p(x_k) \to 0$. This means that $p(\bar{x}) = 0$, which from the definition of p tells us that $\bar{x} \in S$ where S is our constraint set.

Frequently used penalty functions

1. **Polynomial penalty:** $p(x) = \sum_{i=1}^{m} [\max\{0, g_i(x)\}]^q, q \geq 1$
   (a) Linear penalty: $(q = 1) : p(x) = \sum_{i=1}^{m} [\max\{0, g_i(x)\}]$
   (b) Quadratic penalty: $(q = 2) : p(x) = \sum_{i=1}^{m} [\max\{0, g_i(x)\}]$

   For example, if we define $g_i^+(x) = \max\{0, g_i(x)\}$, then $g^+(x) = [g_1^+(x), .., g_m^+(x)]^T$. The penalty function $P(x) = g^+(x)^T g^+(x)$, or $P(x) = g^+(x)^T \Gamma g^+(x)$ where $\Gamma > 0$
2. **Penalty for problem with equality and inequality constraints**

$$
\begin{aligned}
P: \min_x f(x) \\
\text{s.t. } g(x) \leq 0 \\
h(x) = 0 \\
x \in \mathbb{R}^n \\
\text{Need penalty function: } p(x) = 0 \text{ if } g(x) \leq 0 \text{ AND } h(x) = 0 \\
p(x) > 0 \text{ if } g(x) > 0 \text{ OR } h(x) \neq 0 \\
\text{We can use: } p(x) = \sum_{i=1}^{m} [\max\{0, g_i(x)\}]^q + \sum_{i=1}^{k} |h_i(x)|^q, q \geq 1
\end{aligned}
$$

Derivative of the penalty function

Suppose we use $P(x) = \gamma(g^+(x))$, where $g^+(x)$ is as defined previously. An example of $\gamma(x)$ is $\gamma(x) = y^T y$. The difficulty arises when we try to take the derivative of $P(x)$, as the max function $g^+(x)$ is not differentiable. But we will see that if we choose $\gamma(x)$ appropriately, we can make $P(x)$ differentiable.

$$
\begin{aligned}
\frac{\partial P(x)}{\partial x} &= \sum_{i=1}^{m} \frac{\partial \gamma(g^+(x))}{\partial (g_i^+(x))} \frac{\partial g_i^+(x)}{\partial x} \\
\frac{\partial g^+(x)}{\partial x} &= \begin{cases} \frac{\partial g_i(x)}{\partial x} \text{ if } g_i(x) \geq 0 \\ 0 \text{ if } g_i(x) < 0 \end{cases}
\end{aligned}
$$

But $\frac{\partial g_i^+(x)}{\partial x}$ may not be continuous at 0. However, if we choose γ such that $\frac{\partial \gamma(g^+(x))}{\partial y_i} = 0$ whenever $g_i(x) = 0$, then it won't matter if $\frac{\partial g_i^+(x)}{\partial x}$ is discontinuous, because it will be multiplied by 0. One such $\gamma(x)$ is $\sum_{i=1}^{m} [g_i^+(x)]^q, q \geq 1$

KKT in penalty methods

As before, we have:

1. Penalty program: $x_k = \arg\min_x f(x) + c_k P(x)$
2. Penalty function: $P(x) = \gamma(g^+(x))$
3. Derivatives: $\nabla P(x) = \sum_{i=1}^{m} \frac{\partial \gamma(g^+(x))}{\partial (g_i^+(x))} \frac{\partial g_i^+(x)}{\partial x}$

The 1st order condition in local minimum tells us:

$$
\begin{aligned}
0 = \nabla f(x_k) + c_k \nabla P(x_k) = \nabla f(x_k) + \sum_{i=1}^{m} u_{i,k} \nabla g_i(x_k) \text{ where } u_{i,k} = c_k \frac{\partial \gamma(g^+(x_k))}{\partial (g_i^+(x_k))} \\
0 = \nabla f(x_k) + (u_k)^T \nabla g(x_k)
\end{aligned}
$$

u_k now looks like a Lagrange multiplier. Indeed, under some mild conditions, as $x_k \to x^* \implies u_k \to u^*$, where u^* is the Lagrange multiplier at the optimum.

Sequential quadratic programming

Sequential quadratic programming (SQP) is an iterative method for constrained nonlinear optimization. SQP methods are used on mathematical problems for which the objective function and the constraints are twice continuously differentiable. SQP methods solve a sequence of optimization subproblems, each of which optimizes a quadratic model of the objective subject to a linearization of the constraints. If the problem is unconstrained, then the method reduces to Newton's method for finding a point where the gradient of the objective vanishes. If the problem has only equality constraints, then the method is equivalent to applying Newton's method to the first-order optimality conditions, or Karush–Kuhn–Tucker conditions, of the problem.

Algorithm basics

Consider a nonlinear programming problem of the form:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & b(x) \geq 0 \\ & c(x) = 0. \end{aligned}$$

The Lagrangian for this problem is[1]

$$\mathcal{L}(x, \lambda, \sigma) = f(x) - \lambda b(x) - \sigma c(x),$$

where λ and σ are Lagrange multipliers. At an iterate x_k, a basic sequential quadratic programming algorithm defines an appropriate search direction d_k as a solution to the quadratic programming subproblem

$$\begin{aligned} \min_d \quad & f(x_k) + \nabla f(x_k)^T d + \tfrac{1}{2} d^T \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, \sigma_k) d \\ \text{s.t.} \quad & b(x_k) + \nabla b(x_k)^T d \geq 0 \\ & c(x_k) + \nabla c(x_k)^T d = 0. \end{aligned}$$

Note that the term $f(x_k)$ in the expression above may be left out for the minimization problem, since it is constant under the $\min_d$ operator.

Alternative approaches

There are alternative approaches such as the

- Sequential linear programming
- Sequential linear-quadratic programming
- Augmented Lagrangian method

Implementations

SQP methods have been implemented in well known numerical environments such as MATLAB and GNU Octave. There also exist numerous software libraries, including open source:

- SciPy (de facto standard for scientific Python) has scipy.optimize.minimize(method='SLSQP') solver.
- NLopt (C/C++ implementation, with numerous interfaces including Julia, Python, R, MATLAB/Octave), implemented by Dieter Kraft as part of a package for optimal control, and modified by S. G. Johnson.
- LabVIEW
- KNITRO (C, C++, C#, Java, Python, Fortran)
- NPSOL (Fortran)
- SNOPT (Fortran)
- NLPQL (Fortran)
- MATLAB
- SuanShu (Java)

References and further readings

1 A. R. Conn, N. I. M. Gould and Ph. L. Toint, Trust-Region Methods, SIAM 2000.

2. J. Dennis and R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear equations, (republished by) SIAM, 1996.

3. R. Fletcher, Practical Methods of Optimization, 2nd edition Wiley, 1987 (republished in 2000).

4. P. Gill, W. Murray and M. H. Wright, Practical Optimization, Academic Press, 1981.

5. N. I. M. Gould, An Introduction to Algorithms for Continuous Optimization, 2006. Available for download at http://www.numerical.rl.ac.uk/nimg/course/lectures/paper/paper.pdf.

6. J. Nocedal and S. J. Wright, Numerical Optimization, Springer Verlag, 1999 (1st edition) or 2006 (2nd edition).

7. Mathematical Modeling and Computation of Real-Time Problems: An Interdisciplinary Approach (Mathematical Engineering, Manufacturing, and Management Sciences) 1st Edition by Rakhee Kulshrestha, Chandra Shekhar, Madhu Jain, Srinivas R. Chakravarthy