Subgradient Methods

Stephen Boyd, Lin Xiao, and Almir Mutapcic
Notes for EE392o, Stanford University, Autumn, 2003

October 1, 2003

The *subgradient method* is a simple algorithm for minimizing a nondifferentiable convex function. The method looks very much like the ordinary gradient method for differentiable functions, but with several notable exceptions. For example, the subgradient method uses step lengths that are fixed ahead of time, instead of an exact or approximate line search as in the gradient method. Unlike the ordinary gradient method, the subgradient method is *not* a descent method; the function value can (and often does) increase.

The subgradient method is far slower than Newton's method, but is much simpler and can be applied to a far wider variety of problems. By combining the subgradient method with primal or dual decomposition techniques, it is sometimes possible to develop a simple distributed algorithm for a problem.

The subgradient method was originally developed by Shor in the Soviet Union in the 1970s. The basic reference on subgradient methods is his book [Sho85]. Another book on the topic is Akgul [Akg84]. Bertsekas [Ber99] is a good reference on the subgradient method, combined with primal or dual decomposition.

1 The subgradient method

Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is convex. To minimize f, the subgradient method uses the iteration

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}.$$

Here $x^{(k)}$ is the kth iterate, $g^{(k)}$ is *any* subgradient of f at $x^{(k)}$, and $\alpha_k > 0$ is the kth step size. Thus, at each iteration of the subgradient method, we take a step in the direction of a negative subgradient. Recall that a subgradient of f at x is any vector g that satisfies the inequality $f(y) \geq f(x) + g^T(y - x)$ for all y. When f is differentiable, the only possible choice for $g^{(k)}$ is $\nabla f(x^{(k)})$, and the subgradient method then reduces to the gradient method (except, as we'll see below, for the choice of step size).

Since the subgradient method is not a descent method, it is common to keep track of the best point found so far, *i.e.*, the one with smallest function value. At each step, we set

$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f(x^{(k)})\},$$

and set $i_{\text{best}}^{(k)} = k$ if $f(x^{(k)}) = f_{\text{best}}^{(k)}$, *i.e.*, if $x^{(k)}$ is the best point found so far. (In a descent method there is no need to do this — the current point is always the best one so far.) Then we have

$$f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), \dots, f(x^{(k)})\},$$

i.e., $f_{\text{best}}^{(k)}$ is the best objective value found in k iterations. Since $f_{\text{best}}^{(k)}$ is decreasing, it has a limit (which can be $-\infty$). (For convenience of later notations, we label the initial point with 1 instead of 0.)

1.1 Step size rules

Several different types of step size rules are used.

- *Constant step size.* $\alpha_k = h$ is a constant, independent of k.
- *Constant step length.* $\alpha_k = h/\|g^{(k)}\|_2$. This means that $\|x^{(k+1)} - x^{(k)}\|_2 = h$.
- *Square summable but not summable.* The step sizes satisfy

  $$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

  One typical example is $\alpha_k = a/(b+k)$, where $a > 0$ and $b \geq 0$.
- *Nonsummable diminishing.* The step sizes satisfy

  $$\lim_{k\to\infty} \alpha_k = 0, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

  Step sizes that satisfy this condition are called *diminishing step size rules.* A typical example is $\alpha_k = a/\sqrt{k}$, where $a > 0$.

1.2 Convergence results

There are many results on convergence of the subgradient method. For constant step size and constant step length, the subgradient algorithm is guaranteed to converge to within some range of the optimal value, *i.e.*, we have

$$\lim_{k\to\infty} f_{\text{best}}^{(k)} - f^\star < \epsilon,$$

where $f^\star$ denotes the optimal value of the problem, *i.e.*, $f^\star = \inf_x f(x)$. (This implies that the subgradient method finds an ϵ-suboptimal point within a finite number of steps.) The number ϵ is a function of the step size parameter h, and decreases with it.

For the diminishing step size rule (and therefore also the square summable but not summable step size rule), the algorithm is guaranteed to converge to the optimal value, *i.e.*, we have $\lim_{k\to\infty} f(x^{(k)}) = f^\star$.

When the function f is differentiable, we can say a bit more about the convergence. In this case, the subgradient method with constant step size yields convergence to the optimal value, provided the parameter h is small enough.

2 Convergence proof

Here we give a proof of some typical convergence results for the subgradient method. We assume that there is a minimizer of f, say $x^\star$. We also make one other assumption on f: We will assume that the norm of the subgradients is bounded, *i.e.*, there is a G such that $\|g^{(k)}\|_2 \leq G$ for all k. This will be the case if, for example, f satisfies the Lipschitz condition

$$|f(u) - f(v)| \leq G\|u - v\|_2,$$

for all u, v, because then $\|g\|_2 \leq G$ for any $g \in \partial f(x)$, and any x. In fact, some variants of the subgradient method work when this assumption doesn't hold; see [Sho85].

Recall that for the standard gradient descent method, the convergence proof is based on the function value decreasing at each step. In the subgradient method, the key quantity is not the function value (which often increases); it is the *Euclidean distance to the optimal set.*

Recall that $x^\star$ is a point that minimizes f, *i.e.*, it is an arbitrary optimal point. We have

$$\begin{aligned} \|x^{(k+1)} - x^\star\|_2^2 &= \|x^{(k)} - \alpha_k g^{(k)} - x^\star\|_2^2 \\ &= \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k g^{(k)T}(x^{(k)} - x^\star) + \alpha_k^2\|g^{(k)}\|_2^2 \\ &\leq \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k(f(x^{(k)}) - f^\star) + \alpha_k^2\|g^{(k)}\|_2^2, \end{aligned}$$

where $f^\star = f(x^\star)$. The last line follows from the definition of subgradient, which gives

$$f(x^\star) \geq f(x^{(k)}) + g^{(k)T}(x^\star - x^{(k)}).$$

Applying the inequality above recursively, we have

$$\|x^{(k+1)} - x^\star\|_2^2 \leq \|x^{(1)} - x^\star\|_2^2 - 2\sum_{i=1}^k \alpha_i(f(x^{(i)}) - f^\star) + \sum_{i=1}^k \alpha_i^2\|g^{(i)}\|_2^2.$$

Using $\|x^{(k+1)} - x^\star\|_2^2 \geq 0$ we have

$$2\sum_{i=1}^k \alpha_i(f(x^{(i)}) - f^\star) \leq \|x^{(1)} - x^\star\|_2^2 + \sum_{i=1}^k \alpha_i^2\|g^{(i)}\|_2^2.$$

Combining this with

$$\sum_{i=1}^k \alpha_i(f(x^{(i)}) - f^\star) \geq \left(\sum_{i=1}^k \alpha_i\right) \min_{i=1,\ldots,k}(f(x^{(i)}) - f^\star),$$

we have the inequality

$$f_{\rm best}^{(k)} - f^\star = \min_{i=1,\ldots,k} f(x^{(i)}) - f^\star \leq \frac{\|x^{(1)} - x^\star\|_2^2 + \sum_{i=1}^k \alpha_i^2\|g^{(i)}\|_2^2}{2\sum_{i=1}^k \alpha_i}. \qquad (1)$$

Finally, using the assumption $\|g^{(k)}\|_2 \leq G$, we obtain the basic inequality

$$f_{\rm best}^{(k)} - f^\star = \min_{i=1,\ldots,k} f(x^{(i)}) - f^\star \leq \frac{\|x^{(1)} - x^\star\|_2^2 + G^2\sum_{i=1}^k \alpha_i^2}{2\sum_{i=1}^k \alpha_i}. \qquad (2)$$

From this inequality we can read off various convergence results.

Since $x^\star$ is any minimizer of f, we can state that

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{\mathbf{dist}(x^{(1)}, X^\star)^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2\sum_{i=1}^k \alpha_i},$$

where $X^\star$ denotes the optimal set, and $\mathbf{dist}(x^{(1)}, X^\star)$ is the (Euclidean) distance of $x^{(1)}$ to the optimal set.

Constant step size. If $\alpha_k = h$, we have

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{\mathbf{dist}(x^{(1)}, X^\star)^2 + G^2 h^2 k}{2hk}.$$

The righthand side converges to $G^2h/2$ as $k \to \infty$. Thus, for the subgradient method with fixed step size h, $f_{\text{best}}^{(k)}$ that converges to within $G^2h/2$ of optimal.

We can also say, for example, that we have $f(x^{(k)}) - f^\star \leq G^2 h$ within a finite number of steps. (Indeed, we can easily give a specific upper bound on the number of steps needed.)

Constant step length. If $\alpha_k = h/\|g^{(k)}\|_2$, then inequality (1) becomes

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{\mathbf{dist}(x^{(1)}, X^\star)^2 + h^2 k}{2\sum_{i=1}^k \alpha_i}.$$

By assmpution, we have $\alpha_k \geq h/G$. Applying this to the denominator of the above inequality gives

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{\mathbf{dist}(x^{(1)}, X^\star)^2 + h^2 k}{2hk/G}.$$

The righthand side converges to $Gh/2$ as $k \to \infty$, so in this case the subgradient method converges to within $Gh/2$ of optimal.

Square summable but not summable. Now suppose

$$\|\alpha\|_2^2 = \sum_{k=1}^\infty \alpha_k^2 < \infty, \qquad \sum_{k=1}^\infty \alpha_k = \infty.$$

Then we have

$$f_{\text{best}}^{(k)} - f^\star \leq \frac{\mathbf{dist}(x^{(1)}, X^\star)^2 + G^2 \|\alpha\|_2^2}{2\sum_{i=1}^k \alpha_i},$$

which converges to zero as $k \to \infty$. In other words, the subgradient method converges (in the sense $f_{\text{best}}^{(k)} \to f^\star$).

Diminishing step size rule. If the sequence α_k converges to zero and is nonsummable, then the righthand side of the inequality (2) converges to zero, which implies the subgradient method converges. To show this, let $\epsilon > 0$. Then there exists an integer N_1 such that $\alpha_i \le \epsilon/G^2$, for all $i > N_1$. There also exists an integer N_2 such that

$$\sum_{i=1}^{k} \alpha_i \ge \frac{1}{\epsilon}\left(\|x^{(1)} - x^\star\|_2^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2 \right) \quad \text{for all } k > N_2$$

(because $\sum_{i=1}^{\infty} \alpha_i = \infty$). Let $N = \max\{N_1, N_2\}$. Then for all $k > N$, we have

$$\begin{aligned} \min_{i=1,\ldots,k} f(x^{(i)}) - f^\star &\le \frac{\|x^{(1)} - x^\star\|_2^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2}{2\sum_{i=1}^{k} \alpha_i} + \frac{G^2 \sum_{i=N_1+1}^{k} \alpha_i^2}{2\sum_{i=1}^{N_1} \alpha_i + 2\sum_{i=N_1+1}^{k} \alpha_i} \\ &\le \frac{\|x^{(1)} - x^\star\|_2^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2}{\frac{2}{\epsilon}\left(\|x^{(1)} - x^\star\|_2^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2 \right)} + \frac{G^2 \sum_{i=N_1+1}^{k} \frac{\epsilon}{G^2} \alpha_i}{2\sum_{i=N_1+1}^{k} \alpha_i} \\ &= \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon. \end{aligned}$$

3 Projected subgradient method

One extension of the subgradient method is the *projected subgradient method*, which solves the constrained convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C}, \end{array}$$

where $\mathcal{C}$ is a convex set. The projected subgradient method is given by

$$x^{(k+1)} = P\left(x^{(k)} - \alpha_k g^{(k)} \right),$$

where P is (Euclidean) projection on $\mathcal{C}$, and $g^{(k)}$ is any subgradient of f at $x^{(k)}$. The step size rules described before can be used here, with similar convergence results.

The convergence proofs for the subgradient method are readily extended to handle the projected subgradient method. Let $z^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$, *i.e.*, a standard subgradient update, before the projection back onto $\mathcal{C}$. As in the subgradient method, we have

$$\begin{aligned} \|z^{(k+1)} - x^\star\|_2^2 &= \|x^{(k)} - \alpha_k g^{(k)} - x^\star\|_2^2 \\ &= \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k g^{(k)T}(x^{(k)} - x^\star) + \alpha_k^2 \|g^{(k)}\|_2^2 \\ &\le \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^\star) + \alpha_k^2 \|g^{(k)}\|_2^2. \end{aligned}$$

Now we observe that

$$\|x^{(k+1)} - x^\star\|_2 = \|P(z^{(k+1)}) - x^\star\|_2 \le \|z^{(k+1)} - x^\star\|_2,$$

i.e., when we project a point onto $\mathcal{C}$, we move closer to every point in $\mathcal{C}$, and in particular, any optimal point. Combining this with the inequality above we get

$$\|x^{(k+1)} - x^\star\|_2^2 \le \|x^{(k)} - x^\star\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^\star) + \alpha_k^2 \|g^{(k)}\|_2^2,$$

and the proof proceeds exactly as in the ordinary subgradient method.

3.1 Approximate projected subgradient method

This proof reveals that the method still works even when P is not the Euclidean projection operator. All that matters is that P should satisfy $P(u) \in \mathcal{C}$, and the inequality

$$\|P(u) - z\|_2 \leq \|u - z\|_2,$$

for any $z \in \mathcal{C}$. In other words, P must return a point in $\mathcal{C}$ that is no farther from z than u is. In other words, the operator P can be an *approximate projection operator*; it only needs to not increase the distance to the set.

One practical example comes up when $\mathcal{C}$ can be expressed as

$$\mathcal{C}_1 \supseteq \mathcal{C}_2 \supseteq \cdots \supseteq \mathcal{C}_k = \mathcal{C},$$

where $\mathcal{C}_i$ are convex sets, and we have a simple method for computing $P_i(z)$, the Euclidean projection on $\mathcal{C}_i$, provided $z \in \mathcal{C}_{i-1}$. Then we can construct an approximate projection operator on $\mathcal{C}$ as

$$P(u) = P_k \circ P_{k-1} \circ \cdots \circ P_1(u),$$

i.e., by first projecting u onto $\mathcal{C}_1$, then projecting the result onto $\mathcal{C}_2$, and so on. To see that this satisfies the required condition, let $z \in \mathcal{C}$. Since $z \in \mathcal{C}_1$, we have

$$\|P_1(u) - z\|_2 \leq \|u - z\|_2.$$

Since $z \in \mathcal{C}_2$, we have

$$\|P_2(P_1(u)) - z\|_2 \leq \|P_1(u) - z\|_2,$$

which, combined with the inequality above yields

$$\|P_2(P_1(u)) - z\|_2 \leq \|u - z\|_2.$$

Continuing in this way we find that $\|P(u) - z\|_2 \leq \|u - z\|_2$.

4 Piecewise linear minimization

Our first example is minimizing a piecewise linear convex function:

$$\text{minimize} \quad f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i).$$

Of course this problem is readily (and efficiently) solved via linear programming. Finding a subgradient of f is easy: given x, we first find an index j for which

$$a_j^T x + b_j = \max_{i=1,\ldots,m}(a_i^T x + b_i).$$

Then we can take as subgradient $g = a_j$. The subgradient method update has the form

$$x^{(k+1)} = x^{(k)} - \alpha_k a_j,$$

where j is chosen so that $a_j^T x^{(k)} + b_j = \max_{i=1,\ldots,m}(a_i^T x^{(k)} + b_i)$.

Note that to apply the subgradient method, all we need is a way to evaluate $\max_i(a_i^T x + b_i)$ (along with an index corresponding to one of the maximizers) and the ability to carry out the simple update above. If the problem is dense and very large (and so, beyond the abilities of standard linear programming), but we have some efficient method for evaluating f, the subgradient method might be a reasonable choice of algorithm.

We illustrate the subgradient method with a specific problem instance with $n = 10$ variables and $m = 100$ terms. The problem data a_i and b_i were generated from a unit normal distribution.

We first consider the constant step length rule $\alpha_k = h/\|g^{(k)}\|_2$. Figure 1 shows convergence for several different values of step length h. The figure reveals a trade-off: larger h gives faster convergence, but larger final suboptimality. Figure 2 shows the bound given by the basic inequality (2). In this case for constant step length $h = 0.02$, the inequality is

$$f_{\text{best}}^{(k)} - f^\star = \min_{i=0,\ldots,k} f(x^{(i)}) - f^\star \leq \frac{\|x^{(0)} - x^\star\|_2^2 + h^2 k}{2hk/G},$$

where $\alpha_k = 0.02/\|g^{(k)}\|_2$ and we have $\alpha_k \geq h/G$. The constant G is given by $G = \max_i \|a_i\|_2$ for $i = 1, \ldots, m$. The plot of the bound and $f_{\text{best}}^{(k)} - f^\star$ given at each step k reveals that the bound is not very tight.

Figure 3 shows the convergence of the subgradient method with constant step size rule $\alpha_k = h$, for several values of h.

To illustrate the subgradient method with some diminishing step size rules, we consider the nonsummable diminishing step size rule $\alpha = 0.1/\sqrt{k}$, and the square summable but not summable step rule $\alpha = 0.1/k$. The convergence for these step size rules is plotted in figure 4. Figure 5 shows $f(x^{(k)}) - f^\star$ and its upper bound based on the inequality (2) (which we see is not very tight).

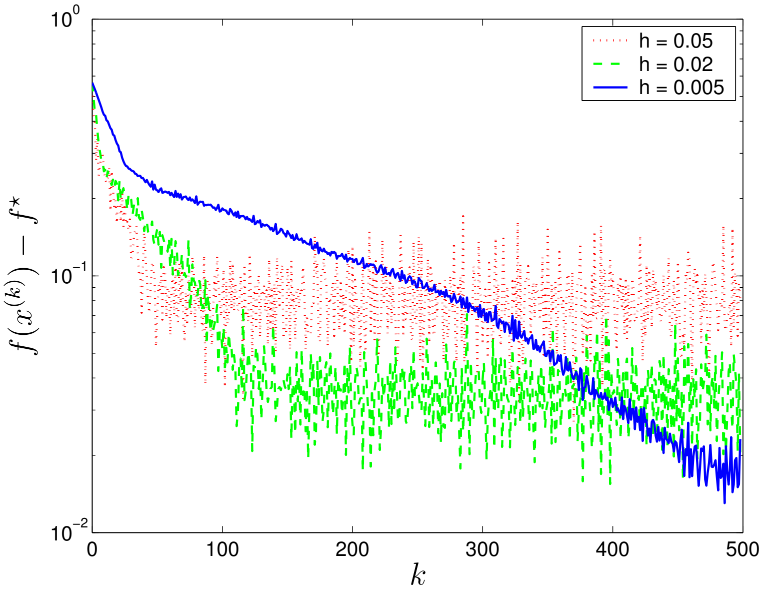

Figure 1: Function value versus iteration number k, for the subgradient method with constant step length h.

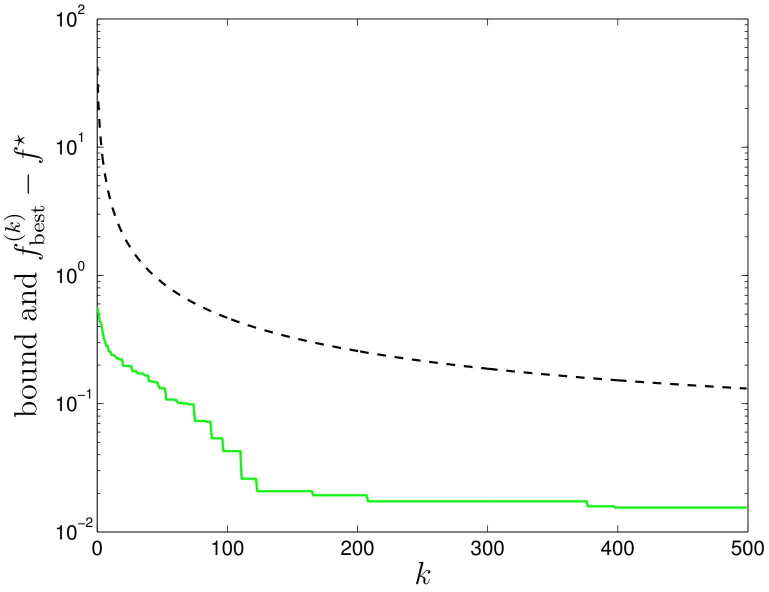

Figure 2: Upper bound (dashed line) and $f_{\mathrm{best}}^{(k)} - f^\star$ (solid line) versus iteration number k for subgradient method with constant step length $h = 0.02$.

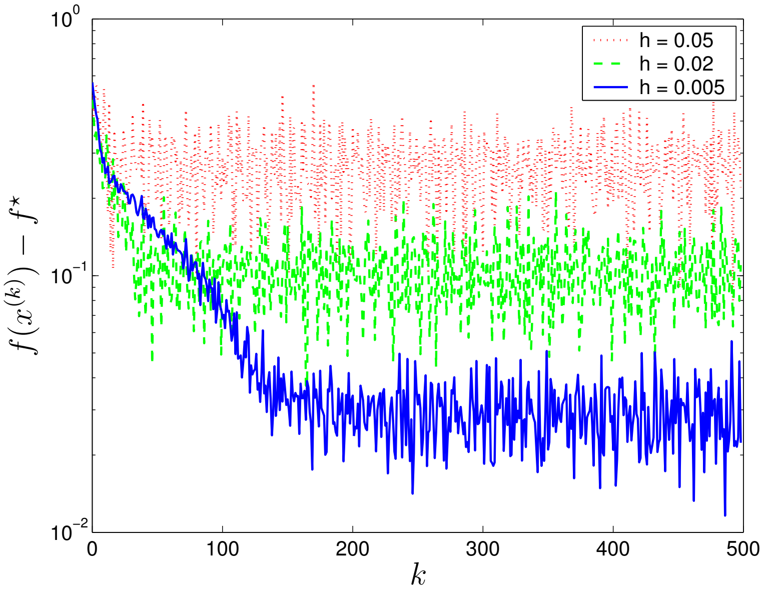

Figure 3: Function value versus iteration number k, for the subgradient method with constant step size h.

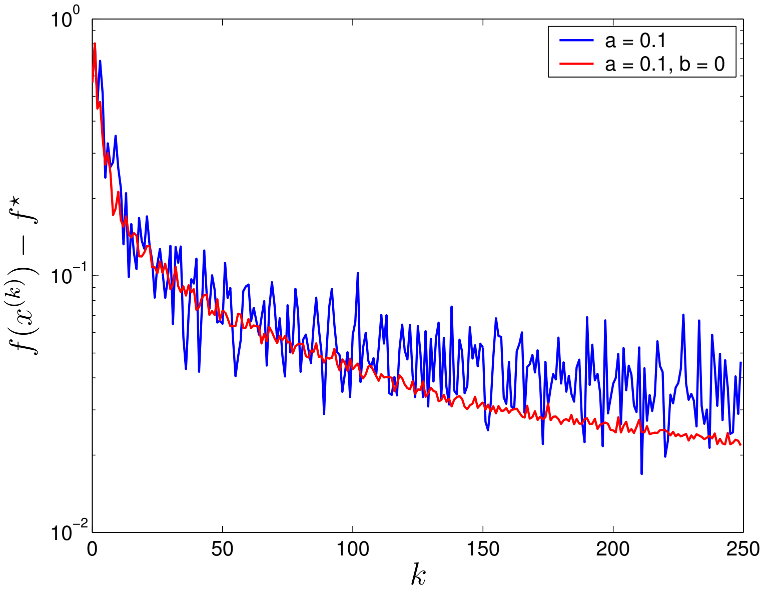

Figure 4: Function value versus iteration number k, for the subgradient method with diminishing step rule $\alpha = 0.1/\sqrt{k}$, and square summable step size rule $\alpha = 0.1/k$.

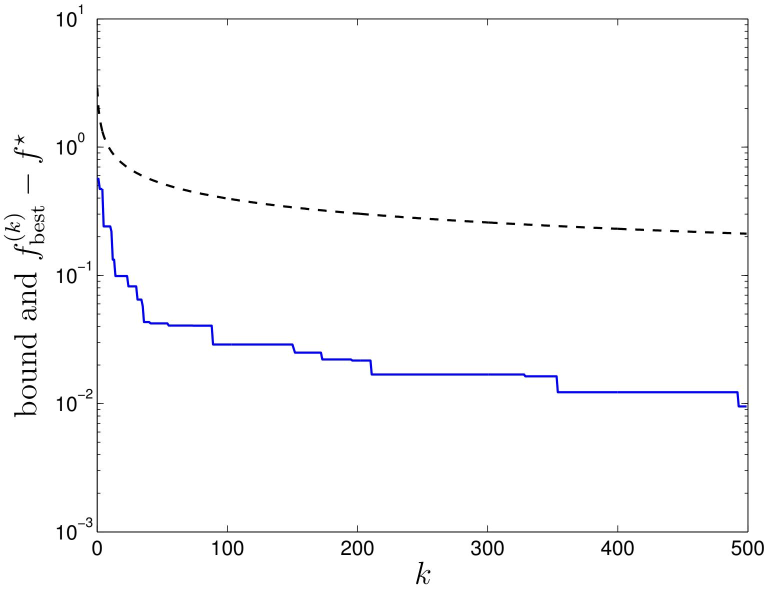

Figure 5: Upper bound (dashed line) and $f_{\rm best}^{(k)} - f^\star$ (solid line) versus iteration number k for linear piecewise minimization with diminishing step rule, $\alpha = 0.1/\sqrt{k}$.

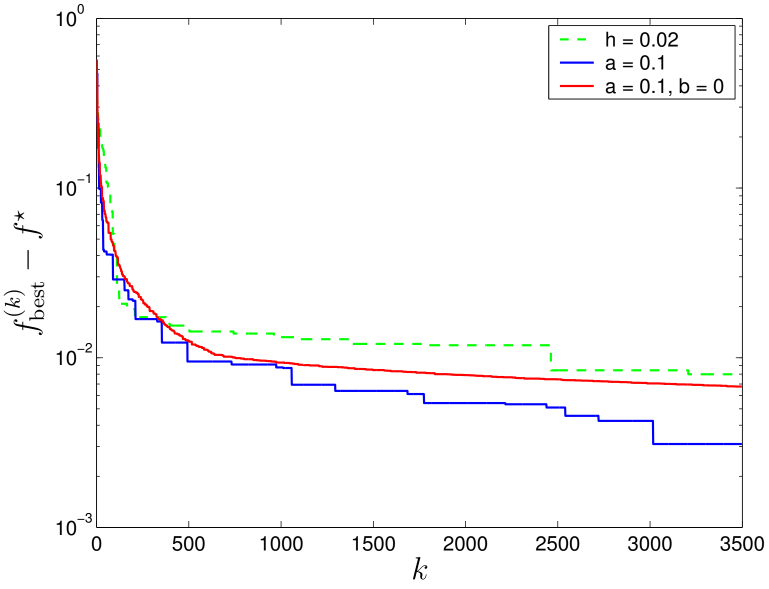

Figure 6: The best function value $f_{\rm best}^{(k)} - f^\star$ versus iteration number k, in the subgradient method with constant step length $h = 0.02$, diminishing step rule $\alpha = 0.1/\sqrt{k}$, and square summable step rule $\alpha = 0.1/k$.

5 Optimal network flow

In this section we consider a longer example that illustrates a typical use for the subgradient method. One very interesting feature of the solution method we derive is that it is completely distributed. For background and more on network flow see Boyd and Vandeberghe [BV03].

5.1 Optimal network flow problem

We consider a connected directed graph or network with n edges and p nodes. We let x_j denote the flow or traffic on arc j, with $x_j > 0$ meaning flow in the direction of the arc, and $x_j < 0$ meaning flow in the direction opposite the arc. There is also a given external source (or sink) flow s_i that enters (if $s_i > 0$) or leaves (if $s_i < 0$) node i. The flow must satisfy a conservation equation, which states that at each node, the total flow entering the node, including the external sources and sinks, is zero. This conservation equation can be expressed as $Ax = s$ where $A \in \mathbf{R}^{p \times n}$ is the *node incidence matrix* of the graph,

$$A_{ij} = \begin{cases} 1 & \text{arc } j \text{ leaves node } i \\ -1 & \text{arc } j \text{ enters node } i \\ 0 & \text{otherwise.} \end{cases}$$

Thus, each column of A describes a link; it has exactly two nonzero entries (one equal to 1 and the other equal to -1) indicating the start and end nodes of the link respectively. Each row of A describes all links incident to a node: the $+1$ entries indicate outgoing links while the -1 entries indicate incoming links.

The flow conservation equation $Ax = s$ is inconsistent unless $\mathbf{1}^T s = 0$, which we assume is the case. (In other words, the total of the source flows must equal the total of the sink flows.) The flow conservation equations $Ax = s$ are also redundant, since $\mathbf{1}^T A = 0$. To obtain an independent set of equations we can delete any one equation, to obtain $\tilde{A}x = b$, where $\tilde{A} \in \mathbf{R}^{(p-1)\times n}$ is the *reduced node incidence matrix* of the graph (*i.e.*, the node incidence matrix with one row removed) and $b \in \mathbf{R}^{p-1}$ is reduced source vector (*i.e.*, s with the associated entry removed).

We will take traffic flows x as the variables, and the sources as given. We introduce the separable objective function

$$f(x) = \sum_{j=1}^{n} \phi_j(x_j),$$

where $\phi_j : \mathbf{R} \to \mathbf{R}$ is the flow cost function for arc j. We assume that the flow cost functions are convex.

The problem of choosing the best flow that satisfies the flow conservation requirement is formulated as

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{n} \phi_j(x_j) \\ \text{subject to} & Ax = s. \end{array} \qquad (3)$$

There are many ways to solve this problem, but here we explore a method based on *dual decomposition*, using a subgradient algorithm to solve the dual.

5.2 The dual problem

The Langrangian is

$$\begin{aligned} L(x,\nu) &= \sum_{j=1}^{n} \phi_j(x_j) + \nu^T(s - Ax) \\ &= \sum_{j=1}^{n} \Big(\phi_j(x_j) - (a_j^T\nu)x_j\Big) + \nu^T s, \end{aligned}$$

where a_j is the jth column of A. We use the notation $\Delta\nu_j$ to denote $a_j^T\nu$, since it is the difference of the dual variable between the starting node and ending node of arc j. We will see that the dual variables ν_i can be interpreted as *potentials* on the network; $\Delta\nu_j$ is the potential difference appearing across arc j.

The dual function is

$$\begin{aligned} q(\nu) &= \inf_x L(x,\nu) \\ &= \sum_{j=1}^{n} \inf_{x_j} \left(\phi_j(x_j) - (\Delta\nu_j)x_j\right) + \nu^T s \\ &= -\sum_{j=1}^{n} \phi_j^*(\Delta\nu_j) + \nu^T s, \end{aligned}$$

where ϕ_j^* is the conjugate function of ϕ_j, *i.e.*,

$$\phi_j^*(y) = \sup_x \left(y x_j - \phi_j(x_j)\right).$$

The dual problem is the unconstrained convex problem

$$\text{maximize} \quad q(\nu).$$

There is no duality gap; the optimal values of the primal and dual problems are the same.

5.3 Optimal nework flow via the dual

We will assume that the flow cost functions ϕ_j are strictly convex, which means for each y there is a unique maximizer of $yx_j - \phi_j(x_j)$. We will denote this maximizer as $x_j^*(y)$. If ϕ_j is differentiable, then $x_j^* = (\phi_j')^{-1}$, the inverse of the derivative.

We can solve the network flow problem via the dual, as follows. We first solve the dual by maximizing $q(\nu)$ over ν to obtain the optimal dual variable (or potentials) $\nu^\star$. Then the optimal solution of the network flow problem is given by

$$x_j^\star = x_j^*(\Delta\nu_j^\star).$$

Thus, the optimal flow on link j is a function of the optimal potential difference across it. In particular, the optimal flow can be determined locally; we only need to know the optimal potential values at the two adjacent nodes to find the optimal flow on the arc.

5.4 Subgradient method for the dual problem

We will use the subgradient method to solve the dual problem. We first need to find a subgradient for the negative dual function $-q$ (since we are to maximize q, *i.e.*, minimize $-q$). We start by working out a subgradient for the function

$$\phi_j^*(y) = \sup_x (yx - \phi_j(x)).$$

By the supremum rule for subgradients, we can find a subgradient as any subgradient of $yx_j^* - \phi_j(x_j^*)$, where $x_j^* = x_j^*(y)$. We get the very simple expression x_j^*:

$$x_j^*(y) \in \partial\phi_j^*(y).$$

Using this simple expression, we obtain a subgradient for the negative dual function $-q$ as

$$g(\nu) = Ax^*(\Delta\nu) - s.$$

Written out in components, we have

$$g_i(\nu) = a_i^T x - s_i,$$

which is precisely the residual for the ith flow conservation law constraint. (The quantity $-g_i$ is sometimes called the *flow surplus* at node i.) As a consequence, we have

$$\|Ax - s\|_2 = \left(\sum_{i=1}^p g_i^2\right)^{1/2},$$

i.e., the norm of the equality constraint residual is exactly the same as the norm of the subgradient given above.

The subgradient method applied to the dual can be expressed as:

$$\begin{aligned} x_j &:= x_j^*(\Delta\nu_j) \\ g_i &:= a_i^T x - s_i \\ \nu_i &:= \nu_i - \alpha g_i, \end{aligned}$$

where α is the step length in the subgradient method (which may vary with iteration number).

The method proceeds as follows. Given the current value of the potentials, a flow is calculated. This is local; to find x_j we only need to know the two potential values at the ends of arc j. We then compute the flow surplus at each node. Again, this is local; to find the flow surplus at node i, we only need to know the flows on the arcs that enter or leave node i. Finally, we update the potentials based on the current flow surpluses. The update is very simple: we increase the potential at a node with a positive flow surplus (recall that flow surplus is $-g_i$ at node i), which (we hope) will result in reduced flow into the node. Provided the step length α can be computed locally, the algorithm is distributed; the arcs and nodes only need information relating to their adjacent flows and potentials. There is no need to know the global topology of the network, or any other nonlocal information, such as what the flow cost functions are. (Note that constant step length ruins the distributed nature of the algorithm, but constant or diminishing step size rules do not.)

The flows x_j only depend on the potential differences $\Delta\nu_j$. If $\nu^\star$ is optimal, so is $\nu^\star + c\mathbf{1}$ for any constant $c \in \mathbf{R}$. (This is consistent with our interpretation of ν as a potential.) The nonuniqueness of the dual solution comes from the fact that the constraints $Ax = s$ in the primal problem are redundant. We can, without loss of generality, fix the potential at an arbitrary node and then update only the remaining $p - 1$ nodes. (This corresponds to removing an equation in $Ax = s$ and working with the reduced incidence matrix.)

5.5 Analogy with electrical networks

We can give a nice analogy between the optimal flow problem and electrical networks. We consider an electrical network with topology determined by A. The variable x_j is the current flow in branch j (with positive indicating flow in the reference direction, negative indicating current flow in the opposite direction). The source s_i is an external current injected at node i. Naturally, the sum of the external currents must be zero. The flow conservation equation $Ax = s$ is Khirkov's current law (KCL).

The dual variables correspond to the node potentials in the circuit. We can arbitrarily choose one node as the ground or datum node, and measure potentials with respect to that node. The potential difference $\Delta\nu_j$ is precisely the voltage appearing across the j branch of the circuit. Each branch in the circuit contains a nonlinear resistor, with current-voltage characteristic $I_j = x_j^*(V_j)$.

It follows that the optimal flow is given by the current in the branches, with the topology determined by A, external current s, and current-voltage characteristics related to the flow cost functions. The node potentials correspond to the optimal dual variables.

The subgradient algorithm gives us an iterative way to find the currents and voltages in such a circuit. The method updates the node potentials in the circuit. For a given set of node potentials we calculate the branch currents from the branch current-voltage characteristics. Then we calculate the KCL residual, *i.e.*, the excess current at each node, and update the potentials based on these mismatches. In particular, we increase the node potential at each node which has too much current flowing into it, and decrease the potential at each node which has too little current flowing into it. (For constant step size, the subgradient method corresponds roughly to putting a capacitor to ground at each node.)

5.6 Example: minimal queueing delays

We now consider a more specific example, with flow cost function

$$\phi_j(x_j) = \frac{|x_j|}{c_j - |x_j|},$$

where $c_j > 0$ are given. (We define $\phi_j(x_j) = \infty$ for $|x_j| \geq c_j$.) For $x_j \geq 0$, this function gives the expected queuing delay in an M/M/1 queue, with exponential arrival times with rate x_j and exponential service time with rate c_j. The constant c_j is called the link *capacity*. We assume that traffic can flow either way down the link.

The conjugate of this function is

$$\phi_j^*(y) = \begin{cases} 0, & |y| \leq 1/c_j \\ \left(\sqrt{|c_j y|} - 1\right)^2, & |y| > 1/c_j. \end{cases}$$

The function and its conjugate are plotted in figure 7, for $c = 1$.

From the conjugate we can work out the function $x_j^*(\Delta\nu_j)$:

$$x_j = \arg \min_{z \in \mathbf{dom}\, \phi_j} (z\Delta\nu_j - \phi_j(z)) = \begin{cases} c_j - \sqrt{c_j}/\sqrt{\Delta\nu_j}, & \Delta\nu_j > 1/c_j \\ 0, & |\Delta\nu_j| \leq 1/c_j \\ \sqrt{c_j}/\sqrt{-\Delta\nu_j} - c_j, & \Delta\nu_j < -1/c_j. \end{cases} \tag{4}$$

This function (which corresponds to the current-voltage characteristic of a nonlinear resistor in the analogous electrical network) is shown in figure 8. Note that it has a symmetric Zener type characteristic: no current flows unless the voltage exceeds one.

Now we will consider a specific problem instance with $p = 5$ nodes and $n = 7$ arcs, with the network shown in figure 9. The incidence matrix is

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}.$$

Each link has capacity $c_j = 1$, and the source vector is

$$s = (0.2, 0.6, 0, 0, -0.8).$$

We use the subgradient method to solve the dual problem, with initial dual variables $\nu_i = 0$ for $i = 1, \ldots, p$. At each step of the subgradient method, we fix the value of ν_p and only update the dual variables at the remaining $p - 1$ nodes. We use a constant step size rule. The optimal solutions are plotted in figure 10.

Since the primal objective function is strictly convex, the dual function is actually differentiable, so the subgradient we obtain at each step is in fact the gradient. The subgradient method with constant step size is guaranteed to converge to the optimal solution, if the step size is small enough (and in any case, it converges to within $G^2h/2$ of optimal).

Second, while the constant stepsize rule can be implemented in a distributed manner, the constant step length rule cannot, because it requires computation of the norm of the subgradient in order to normalize the subgradient.

Figure 11 shows the value of the dual function at each iteration, for four different constant stepsize rules, and figure 12 shows the corresponding primal residual $\|Ax - s\|_2$ (which is precisely the norm of the subgradient).

For $\alpha = 2$, the primal residual reduces to 4.28×10^{-5} after 100 iterations. The plot suggests that for $\alpha = 3$, the algorithm does not converge to the optimal point (but does, of course, converge to within $G^2\alpha/2$ of optimal). Figures 13 and 14 show the cases with four different nonsummable diminishing stepsize rules.

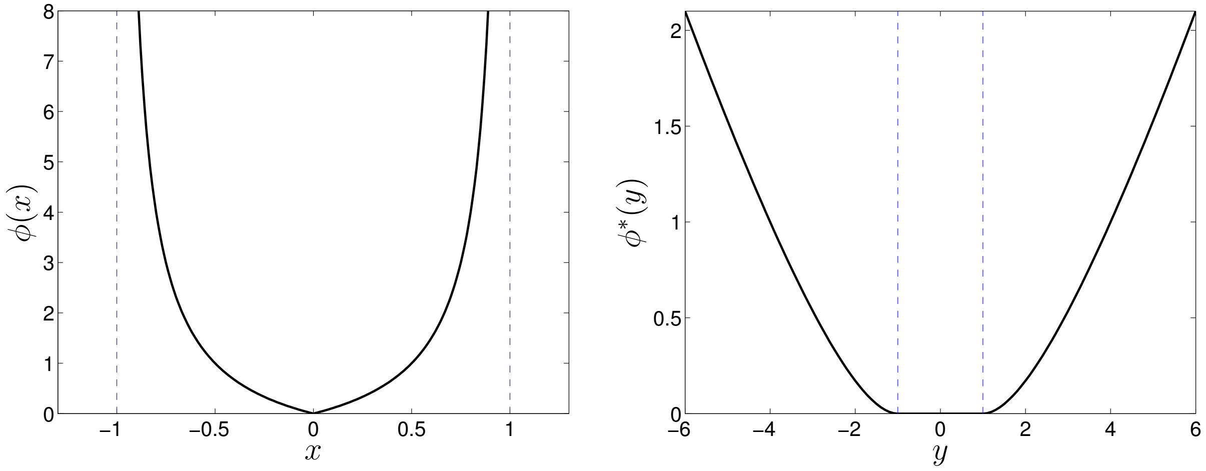

Figure 7: The queueing delay cost function $\phi(x)$ (left) and its conjugate function $\phi^*(y)$ (right).

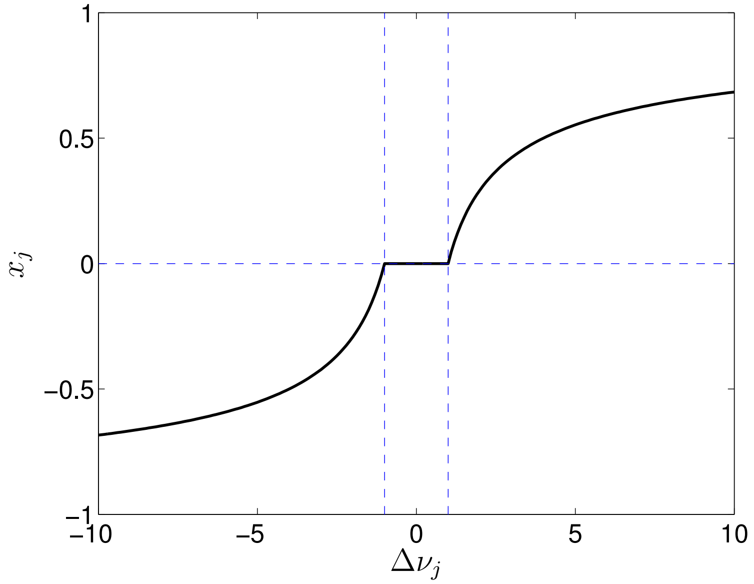

Figure 8: The function $x_j^*(\Delta\nu_j)$, for $c_j = 1$. This can be interpreted as the current-voltage characteristic of a nonlinear resistor.

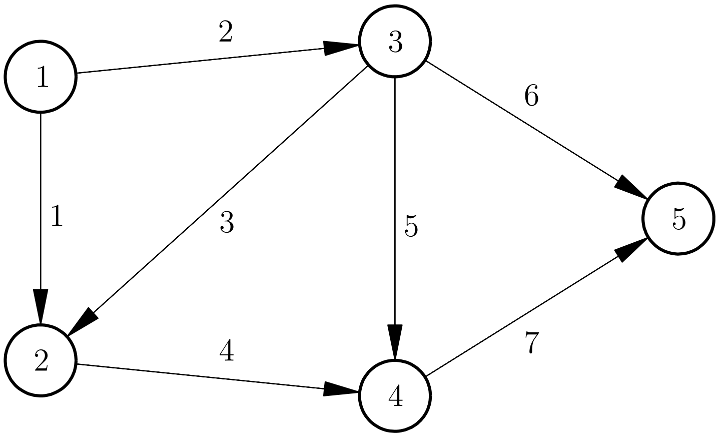

Figure 9: A network with 5 nodes and 7 arcs. Each arc j is given a reference direction. The flow x_j is positive in the reference direction and negative in the opposite direction.

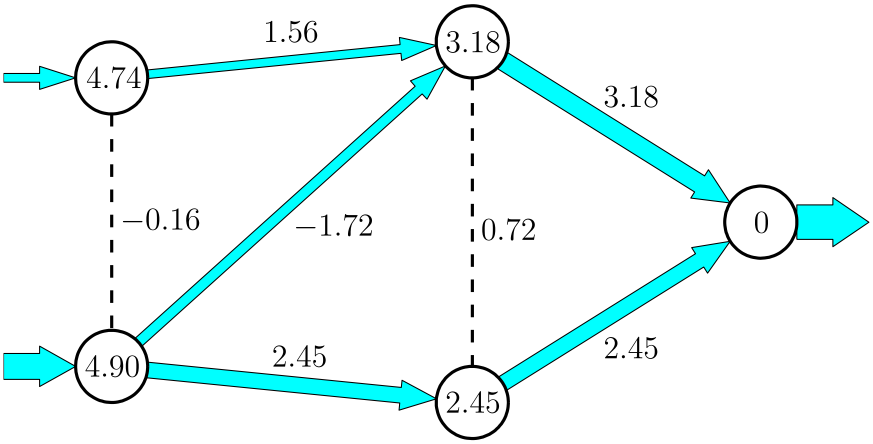

Figure 10: Optimal network flows plotted as pipes with arrows, and the widths of the pipes are proportional to the flows x_j. The numbers at each node i are the optimal dual variables ν_i, and the numbers at each link j are $\Delta\nu_j$, *i.e.*, the differences between the dual variables at their two incident nodes (along the reference direction given in figure 9). Link 1 and 5 have zero flow because their corresponding $\Delta\nu_j$ are smaller than the threshold 1, see figure 8.

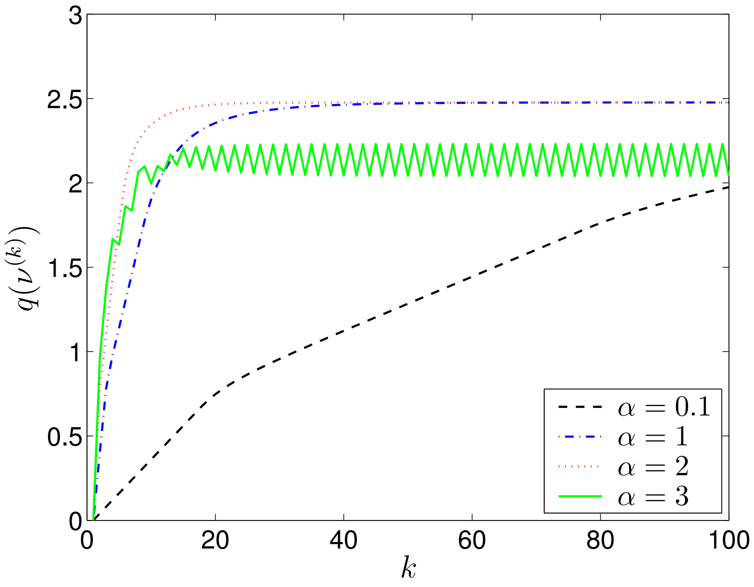

Figure 11: Dual function value versus iteration number k, for the subgradient method with the constant stepsize rule. For $\alpha = 1$ and 2, the dual function comes very close to the optimal value 2.48 after about 40 iterations.

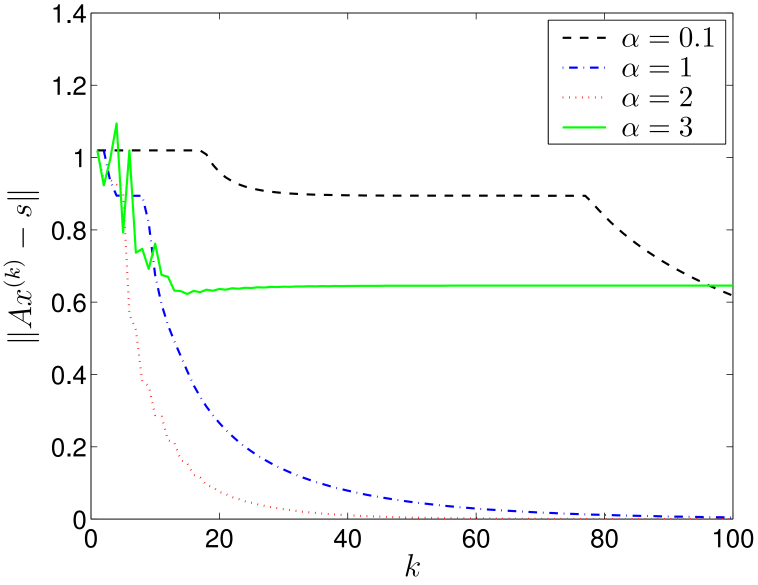

Figure 12: The primal residual $\|Ax - s\|_2$ versus iteration number k, in the subgradient method with the constant stepsize rule. For $\alpha = 2$, the primal residual reduces to 4.28×10^{-5} after 100 iterations.

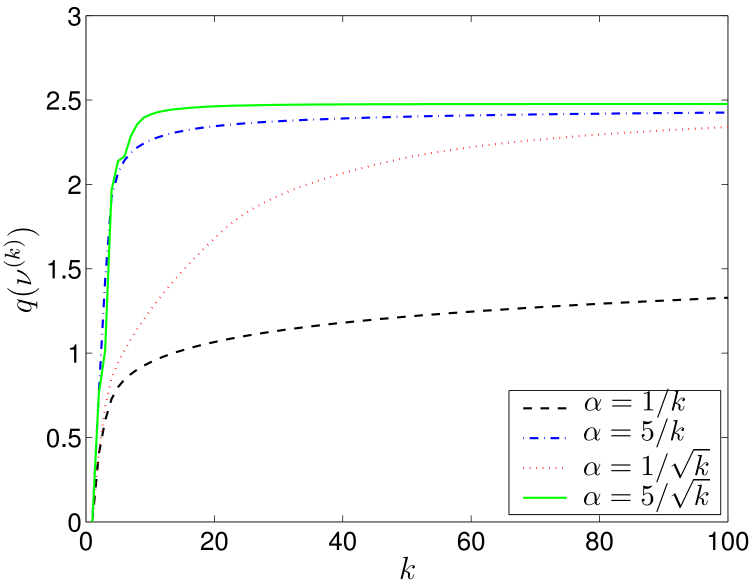

Figure 13: Dual function value versus iteration number k, for the subgradient method with nonsummable diminishing stepsize rules.

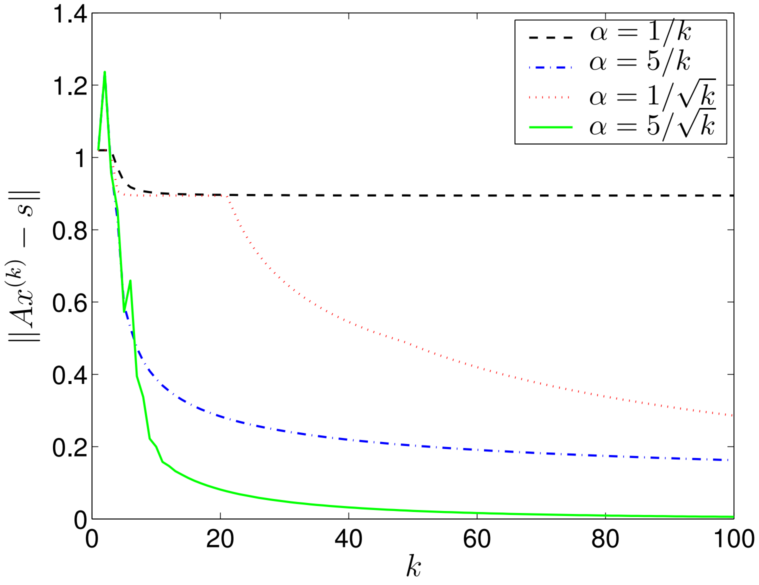

Figure 14: The primal residual $\|Ax - s\|_2$ versus iteration number k, in the subgradient method with nonsummable diminishing stepsize rules.

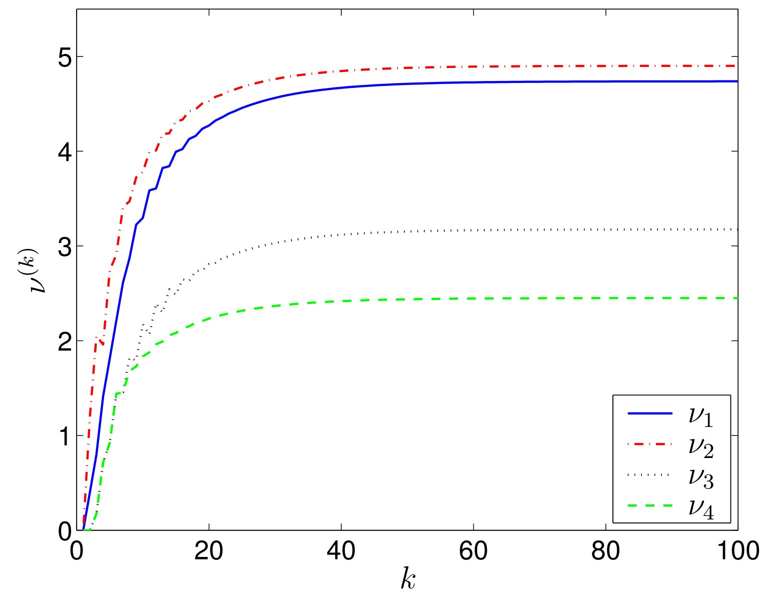

Figure 15: Dual variables $\nu^{(k)}$ versus iteration number k, with constant stepsize rule $\alpha = 2$. Note that ν_5 is fixed to zero.

References

[Akg84] M. Akgül. *Topics in Relaxation and Ellipsoidal Methods*, volume 97 of *Research Notes in Mathematics*. Pitman, 1984.

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[BV03] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[Sho85] N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics. Springer, 1985.