Simple Routines for Optimization

Robert M. Freund

with assistance from Brian W. Anthony

February 12, 2004

1 Outline

- A Bisection Line-Search Algorithm for 1-Dimensional Optimization
- The Conditional-Gradient Method for Constrained Optimization (Frank-Wolfe Method)
- Subgradient Optimization
- Application of Subgradient Optimization to the Lagrange Dual Problem

2 A Bisection Line-Search Algorithm for 1-Dimensional Optimization

Consider the optimization problem:

$$P: \quad \text{minimize}_x \quad f(x)$$

$$\text{s.t.} \qquad x \in \Re^n \ .$$

Let us suppose that $f(x)$ is a differentiable convex function. In a typical algorithm for solving P we have a current iterate value $\bar{x}$ and we choose a direction $\bar{d}$ by some suitable means. The direction $\bar{d}$ is usually chosen to be a *descent direction*, defined by the following property:

$$f(\bar{x} + \epsilon \bar{d}) < f(\bar{x}) \text{ for all } \epsilon > 0 \text{ and sufficiently small} \ .$$

We then typically also perform the 1-dimensional line-search optimization:

$$\bar{\alpha} := \arg\min_{\alpha} f(\bar{x} + \alpha \bar{d}) \ .$$

Let

$$h(\alpha) := f(\bar{x} + \alpha \bar{d}),$$

whereby $h(\alpha)$ is a convex function in the scalar variable α, and our problem is to solve for

$$\bar{\alpha} := \arg\min_{\alpha} h(\alpha).$$

We therefore seek a value $\bar{\alpha}$ for which

$$h^{'}(\bar{\alpha}) = 0.$$

It is elementary to show that

$$h^{'}(\alpha) = \nabla f(\bar{x} + \alpha \bar{d})^T \bar{d}.$$

Property: If $\bar{d}$ is a descent direction at $\bar{x}$, then $h^{'}(0) < 0$.

Because $h(\alpha)$ is a convex function of α, we also have:

Property: $h^{'}(\alpha)$ is a monotone increasing function of α.

Figure 1 shows an example of convex function of two variables to be optimized. Figure 2 shows the function $h(\alpha)$ obtained by restricting the function of Figure 1 to the line shown in that figure. Note from Figure 2 that $h(\alpha)$ is convex. Therefore its first derivative $h^{'}(\alpha)$ will be a monotonically increasing function. This is shown in Figure 3.

Because $h^{'}(\alpha)$ is a monotonically increasing function, we can approximately compute $\bar{\alpha}$, the point that satisfies $h^{'}(\bar{\alpha}) = 0$, by a suitable bisection method. Suppose that we know a value $\hat{\alpha}$ that $h^{'}(\hat{\alpha}) > 0$. Since $h^{'}(0) < 0$ and $h^{'}(\hat{\alpha}) > 0$, the mid-value $\tilde{\alpha} = \frac{0+\hat{\alpha}}{2}$ is a suitable test-point. Note the following:

- If $h^{'}(\tilde{\alpha}) = 0$, we are done.
- If $h^{'}(\tilde{\alpha}) > 0$, we can now bracket $\bar{\alpha}$ in the interval $(0, \tilde{\alpha})$.
- If $h^{'}(\tilde{\alpha}) < 0$, we can now bracket $\bar{\alpha}$ in the interval $(\tilde{\alpha}, \hat{\alpha})$.

This leads to the following *bisection algorithm* for minimizing $h(\alpha) = f(\bar{x} + \alpha \bar{d})$ by solving the equation $h^{'}(\alpha) \approx 0$.

Step 0. Set $k = 0$. Set $\alpha_l := 0$ and $\alpha_u := \hat{\alpha}$.

Step k. Set $\tilde{\alpha} = \frac{\alpha_u + \alpha_l}{2}$ and compute $h^{'}(\tilde{\alpha})$.

- If $h^{'}(\tilde{\alpha}) > 0$, re-set $\alpha_u := \tilde{\alpha}$. Set $k \leftarrow k + 1$.
- If $h^{'}(\tilde{\alpha}) < 0$, re-set $\alpha_l := \tilde{\alpha}$. Set $k \leftarrow k + 1$.

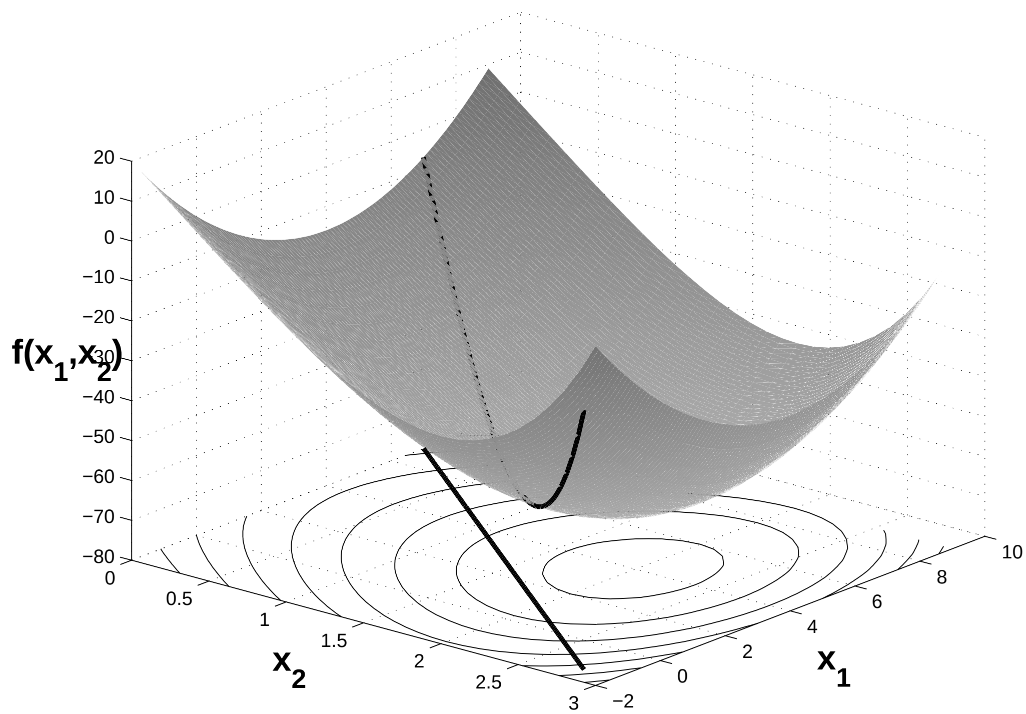

Figure 1: A convex function to be optimized.

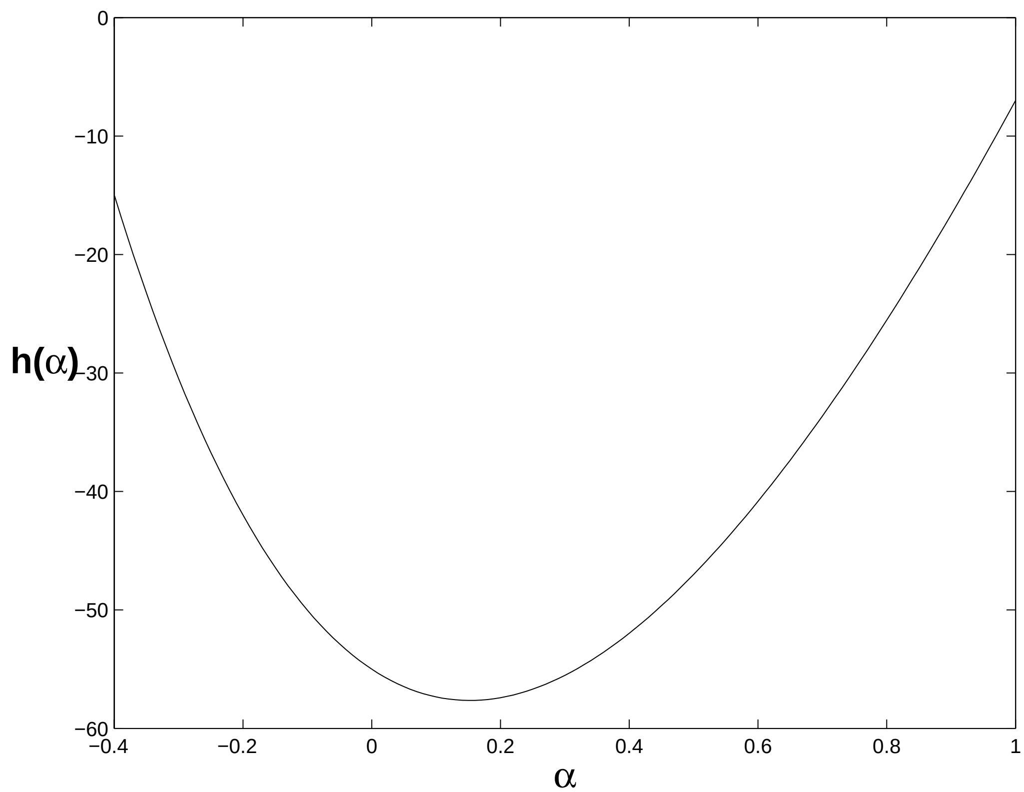

Figure 2: The 1-dimensional function $h(\alpha)$.

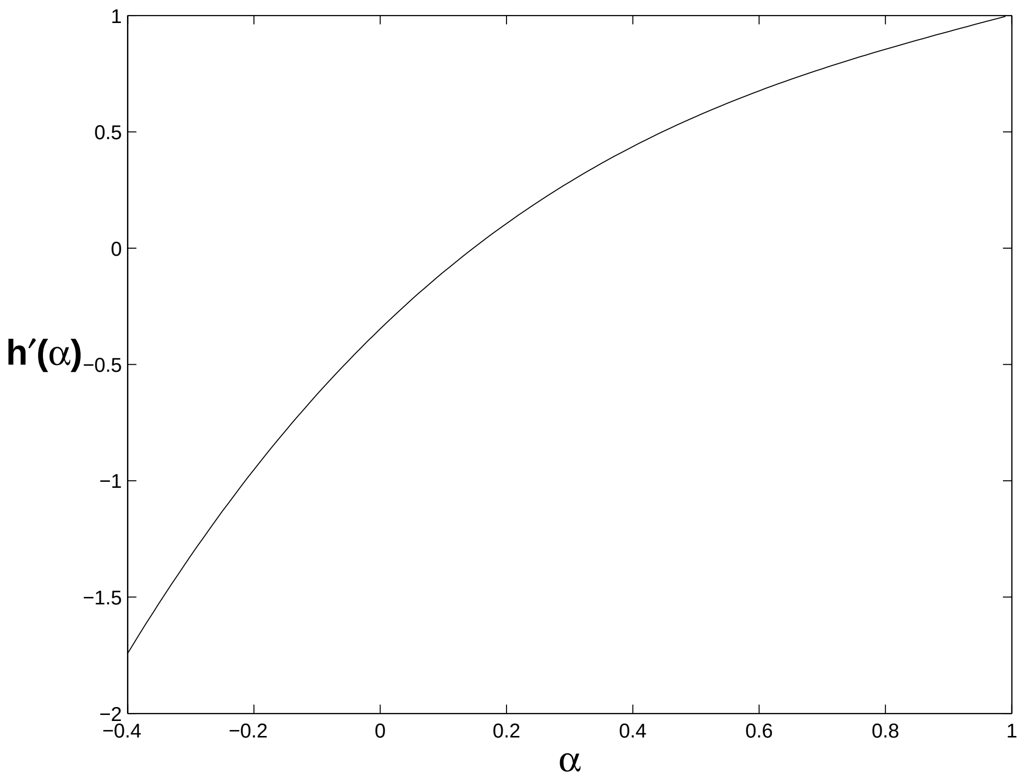

Figure 3: The function $h^{'}(\alpha)$ is monotonically increasing.

- If $h^{'}(\tilde{\alpha}) = 0$, stop.

Property: After every iteration of the bisection algorithm, the current interval $[\alpha_l, \alpha_u]$ must contain a point $\bar{\alpha}$ such that $h^{'}(\bar{\alpha}) = 0$.

Property: At the k^{th} iteration of the bisection algorithm, the length of the current interval $[\alpha_l, \alpha_u]$ is

$$L = \left(\frac{1}{2}\right)^k (\hat{\alpha}).$$

Property: A value of α such that $|\alpha - \bar{\alpha}| \leq \epsilon$ can be found in at most

$$\left\lceil \log_2 \left(\frac{\hat{\alpha}}{\epsilon}\right) \right\rceil$$

steps of the bisection algorithm.

2.1 Computing $\hat{\alpha}$ for which $h^{'}(\hat{\alpha}) > 0$

Suppose that we do not have available a convenient value $\hat{\alpha}$ for which $h^{'}(\hat{\alpha}) > 0$. One way to proceed is to pick an initial "guess" of $\hat{\alpha}$ and compute $h^{'}(\hat{\alpha})$. If $h^{'}(\hat{\alpha}) > 0$, then proceed to the bisection algorithm; if $h^{'}(\hat{\alpha}) \leq 0$, then re-set $\hat{\alpha} \leftarrow 2\hat{\alpha}$ and repeat the process.

2.2 Stopping Criteria for the Bisection Algorithm

In practice, we need to run the bisection algorithm with a stopping criterion. Some relevant stopping criteria are:

- Stop after a fixed number of iterations. That is, stop when $k = \bar{K}$, where $\bar{K}$ is specified by the user.
- Stop when the interval becomes small. That is, stop when $\alpha_u - \alpha_l \leq \epsilon$, where ϵ is specified by the user.
- Stop when $|h^{'}(\tilde{\alpha})|$ becomes small. That is, stop when $|h^{'}(\tilde{\alpha})| \leq \epsilon$, where ϵ is specified by the user.

This third stopping criterion typically yields the best results in practice.

2.3 Modification of the Bisection Algorithm when the Domain of $f(x)$ is Restricted

The discussion and analysis of the bisection algorithm has presumed that our optimization problem is

$$
\begin{array}{lll}
P: & \text{minimize}_x & f(x) \\
 & \text{s.t.} & x \in \Re^n .
\end{array}
$$

Given a point $\bar{x}$ and a direction $\bar{d}$, the line-search problem then is

$$
\begin{array}{lll}
LS: & \text{minimize}_\alpha & h(\alpha) := f(\bar{x} + \alpha \bar{d}) \\
 & \text{s.t.} & \alpha \in \Re .
\end{array}
$$

Suppose instead that the domain of definition of $f(x)$ is an open set $X \subset \Re^n$. Then our optimization problem is:

$$
\begin{array}{lll}
P: & \text{minimize}_x & f(x) \\
 & \text{s.t.} & x \in X ,
\end{array}
$$

and the line-search problem then is

$$
\begin{array}{lll}
LS: & \text{minimize}_\alpha & h(\alpha) := f(\bar{x} + \alpha \bar{d}) \\
 & \text{s.t.} & \bar{x} + \alpha \bar{d} \in X .
\end{array}
$$

In this case, we must ensure that all iterate values of α in the bisection algorithm satisfy $\bar{x} + \alpha \bar{d} \in X$. As an example, consider the following problem:

$$
\begin{array}{lll}
P: & \text{minimize}_x & f(x) := -\sum_{i=1}^{m} \ln(b_i - A_i x) \\
 & \text{s.t.} & b - Ax > 0 .
\end{array}
$$

Here the domain of $f(x)$ is $X = \{x \in \Re^n \mid b - Ax > 0\}$. Given a point $\bar{x} \in X$ and a direction $\bar{d}$, the line-search problem is:

$$\begin{array}{lll} LS: & \text{minimize}_\alpha & h(\alpha) := f(\bar{x} + \alpha \bar{d}) = -\sum_{i=1}^{m} \ln(b_i - A_i(\bar{x} + \alpha \bar{d})) \\ & \text{s.t.} & b - A(\bar{x} + \alpha \bar{d}) > 0. \end{array}$$

Standard arithmetic manipulation can be used to establish that

$$b - A(\bar{x} + \alpha \bar{d}) > 0 \ \text{ if and only if } \ \check{\alpha} < \alpha < \hat{\alpha}$$

where

$$\check{\alpha} := -\min_{A_i \bar{d} < 0} \left\{ \frac{b_i - A_i \bar{x}}{-A_i \bar{d}} \right\} \quad \text{and} \quad \hat{\alpha} := \min_{A_i \bar{d} > 0} \left\{ \frac{b_i - A_i \bar{x}}{A_i \bar{d}} \right\},$$

and the line-search problem then is:

$$\begin{array}{lll} LS: & \text{minimize}_\alpha & h(\alpha) := -\sum_{i=1}^{m} \ln(b_i - A_i(\bar{x} + \alpha \bar{d})) \\ & \text{s.t.} & \check{\alpha} < \alpha < \hat{\alpha}. \end{array}$$

3 The Conditional-Gradient Method for Constrained Optimization (Frank-Wolfe Method)

We now consider the following optimization problem:

$$\begin{array}{lll} P: & \text{minimize}_x & f(x) \\ & \text{s.t.} & x \in C\ . \end{array}$$

We assume that $f(x)$ is a convex function, and that C is a convex set. Herein we describe the conditional-gradient method for solving P, also called the Frank-Wolfe method. This method is one of the cornerstones of optimization, and was one of the first successful algorithms used to solve nonlinear optimization problems. It is based on the premise that the set C

is well-suited for linear optimization. That means that either C is itself a system of linear inequalities $C = \{x \mid Ax \leq b\}$, or more generally that the problem:

$$\begin{array}{lll} LO_c: & \text{minimize}_x & c^T x \\ & \text{s.t.} & x \in C \end{array}$$

is easy to solve for any given objective function vector c.

This being the case, suppose that we have a given iterate value $\bar{x} \in C$. Let us linearize the function $f(x)$ at $x = \bar{x}$. This linearization is:

$$z_1(x) := f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) \ ,$$

which is the first-order Taylor expansion of $f(\cdot)$ at $\bar{x}$. Since we can easily do linear optimization on C, let us solve:

$$\begin{array}{lll} LP: & \text{minimize}_x & z_1(x) = f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) \\ & \text{s.t.} & x \in C \ , \end{array}$$

which simplifies to:

$$\begin{array}{lll} LP: & \text{minimize}_x & \nabla f(\bar{x})^T x \\ & \text{s.t.} & x \in C \ . \end{array}$$

Let x^* denote the optimal solution to this problem. Then since C is a convex set, the line segment joining $\bar{x}$ and x^* is also in C, and we can perform a line-search of $f(x)$ over this segment. That is, we solve:

$$\begin{array}{lll} LS: & \text{minimize}_\alpha & f(\bar{x} + \alpha(x^* - \bar{x})) \\ & \text{s.t.} & 0 \leq \alpha \leq 1 \ . \end{array}$$

Let $\bar{\alpha}$ denote the solution to this line-search problem. We re-set $\bar{x}$:

$$\bar{x} \leftarrow \bar{x} + \bar{\alpha}(x^* - \bar{x})$$

and repeat this process.

The formal description of this method, called the conditional gradient method or the Frank-Wolfe method, is given below:

Step 0: Initialization. Start with a feasible solution $x^0 \in C$. Set $k = 0$. Set $LB \leftarrow -\infty$.

Step 1: Update upper bound. Set $UB \leftarrow f(x^k)$. Set $\bar{x} \leftarrow x^k$.

Step 2: Compute next iterate.

- Solve the problem

$$\begin{array}{rcl} \bar{z} & = & \min_x \quad f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) \\ & & \text{s.t.} \qquad\qquad x \in C \ , \end{array}$$

  and let x^* denote the solution.
- Solve the line-search problem:

$$\begin{array}{ll} \text{minimize}_\alpha & f(\bar{x} + \alpha(x^* - \bar{x})) \\ \text{s.t.} & 0 \leq \alpha \leq 1 \ , \end{array}$$

  and let $\bar{\alpha}$ denote the solution.
- Set $x^{k+1} \leftarrow \bar{x} + \bar{\alpha}(x^* - \bar{x})$

Step 3: Update Lower Bound. Set $LB \leftarrow \max\{LB, \bar{z}\}$.

Step 4: Check Stopping Criteria. If $|UB - LB| \leq \epsilon$, stop. Otherwise, set $k \leftarrow k + 1$ and go to **Step 1**.

3.1 Upper and Lower Bounds in the Frank-Wolfe Method, and Convergence

- The upper bound values UB are simply the objective function values of the iterates $f(x^k)$ for $k = 0, \ldots$. This is a monotonically decreasing sequence because the line-search guarantees that each iterate is an improvement over the previous iterate.

- The lower bound values LB result from the convexity of $f(x)$ and the gradient inequality for convex functions:

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) \quad \text{for any } x \in C \ .$$

  Therefore

$$\min_{x \in C} f(x) \geq \min_{x \in C} f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) \ = \bar{z} \ ,$$

  and so the optimal objective function value of P is bounded below by $\bar{z}$.

We also have the following convergence theorem for the Frank-Wolfe method:
Property: Suppose that C is a bounded set, and that there exists a constant L for which

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

for all $x, y \in C$. Then there exists a constant $\Omega > 0$ for which the following is true:

$$f(x^k) - \min_{x \in C} f(x) \leq \frac{\Omega}{k} \ .$$

3.2 Illustration of the Frank-Wolfe Method

Consider the following instance of P:

$$\begin{aligned} P: \quad \text{minimize} \quad & f(x) \\ \text{s.t.} \quad & x \in C \ , \end{aligned}$$

where

$$f(x) = f(x_1, x_2) = -32x_1 + x_1^4 - 8x_2 + x_2^2$$

and

$$C = \{(x_1, x_2) \mid x_1 - x_2 \leq 1,\ 2.2x_1 + x_2 \leq 7,\ x_1 \geq 0,\ x_2 \geq 0\} .$$

Notice that the gradient of $f(x_1, x_2)$ is given by the formula:

$$\nabla f(x_1, x_2) = \begin{pmatrix} 4x_1^3 - 32 \\ 2x_2 - 8 \end{pmatrix} .$$

Suppose that $x^k = \bar{x} = (0.5, 3.0)$ is the current iterate of the Frank-Wolfe method, and the current lower bound is $LB = -100.0$. We compute $f(\bar{x}) = f(0.5, 3.0) = -30.9375$ and we compute the gradient of $f(x)$ at $\bar{x}$:

$$\nabla f(0.5, 3.0) = \begin{pmatrix} 4x_1^3 - 32 \\ 2x_2 - 8 \end{pmatrix} = \begin{pmatrix} -31.5 \\ -2.0 \end{pmatrix} .$$

We then create and solve the following linear optimization problem:

$$\begin{array}{lll} LP: & \bar{z} = \min_{x_1, x_2} & -30.9375 - 31.5(x_1 - 0.5) - 2.0(x_2 - 3.0) \\ & & \\ & \text{s.t.} & x_1 - x_2 \leq 1 \\ & & 2.2x_1 + x_2 \leq 7 \\ & & x_1 \geq 0 \\ & & x_2 \geq 0 \ . \end{array}$$

The optimal solution of this problem is:

$$x^* = (x_1^*, x_2^*) = (2.5, 1.5) \ ,$$

and the optimal objective function value is:

$$\bar{z} = -50.6875 \ .$$

Now we perform a line-search of the 1-dimensional function

$$\begin{aligned} f(\bar{x} + \alpha(x^* - \bar{x})) \quad = \quad & -32(\bar{x}_1 + \alpha(x_1^* - \bar{x}_1)) + (\bar{x}_1 + \alpha(x_1^* - \bar{x}_1))^4 \\ & -8(\bar{x}_2 + \alpha(x_2^* - \bar{x}_2)) + (\bar{x}_2 + \alpha(x_2^* - \bar{x}_2))^2 \end{aligned}$$

over $\alpha \in [0, 1]$. This function attains its minimum at $\bar{\alpha} = 0.7165$ and we therefore update as follows:

$$x^{k+1} \leftarrow \bar{x} + \bar{\alpha}(x^* - \bar{x}) = (0.5, 3.0) + 0.7165((2.5, 1.5) - (0.5, 3.0)) = (1.9329, 1.9253)$$

and

$$LB \leftarrow \max\{LB, \bar{z}\} = \max\{-100, -50.6875\} = -50.6875 \ .$$

The new upper bound is

$$UB = f(x^{k+1}) = f(1.9329, 1.9253) = -59.5901 \ .$$

This is illustrated in Figure 4.

4 Subgradient Optimization

4.1 Definition

Suppose that $f(x)$ is a convex function. If $f(x)$ is differentiable, we have the gradient inequality:

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^T(x - \bar{x}) \ \text{ for any } x \in X \ ,$$

where typically we think of $X = \Re^n$. This inequality is illustrated in Figure 5.

There are many important convex functions that are not differentiable. The notion of the gradient generalizes to the concept of a *subgradient* of a convex function. A vector $g \in \Re^n$ is called subgradient of the convex function $f(x)$ at $x = \bar{x}$ if the following inequality is satisfied:

$$f(x) \geq f(\bar{x}) + g^T(x - \bar{x}) \quad \text{ for all } x \in X \ .$$

This definition is illustrated in Figure 6.

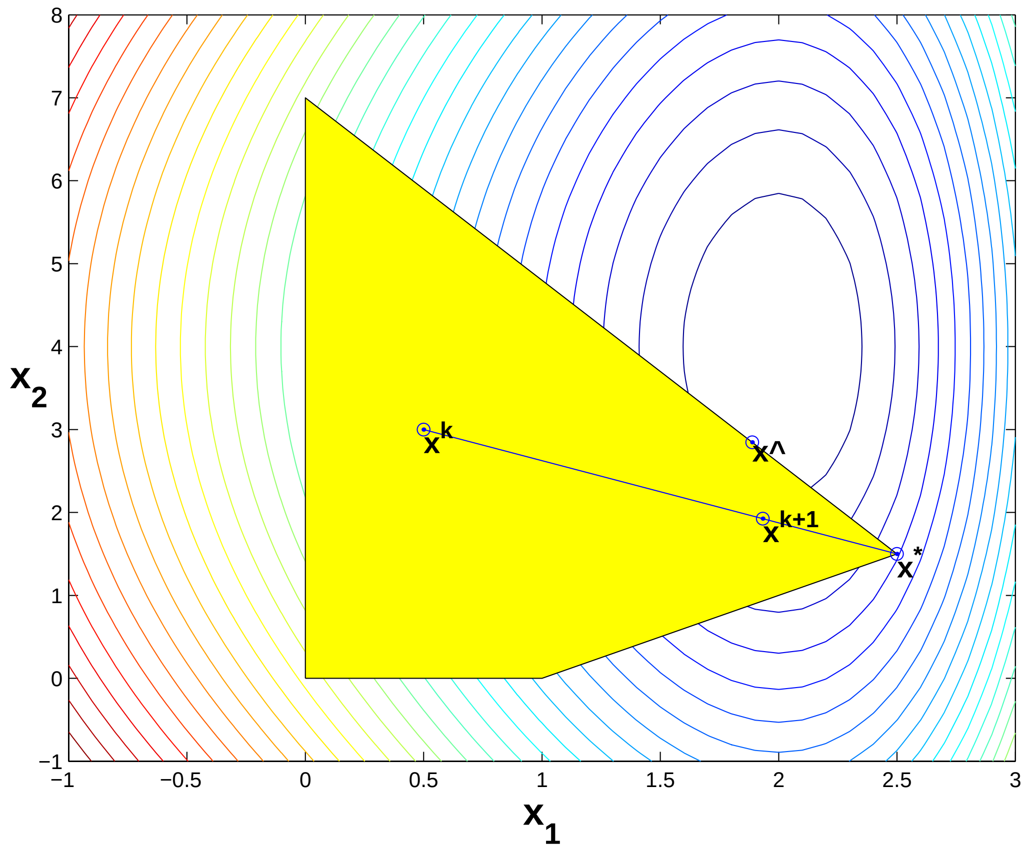

Figure 4: Illustration of an iteration of the Frank-Wolfe method.

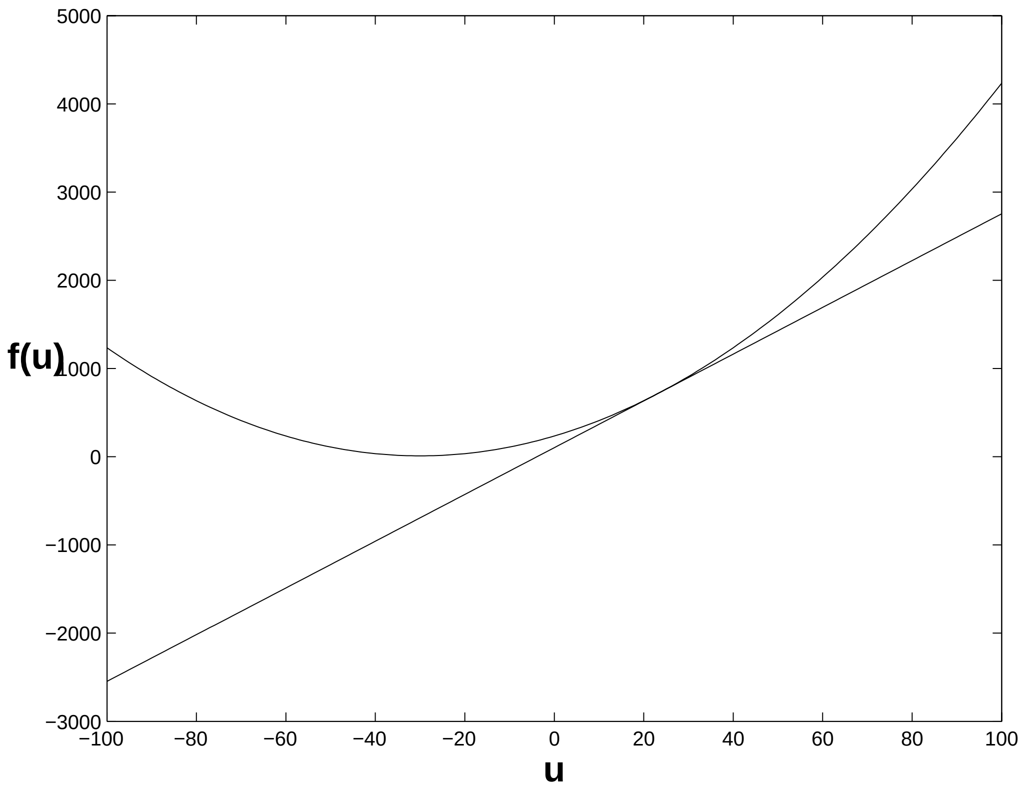

Figure 5: The gradient and the gradient inequality for a differentiable convex function.

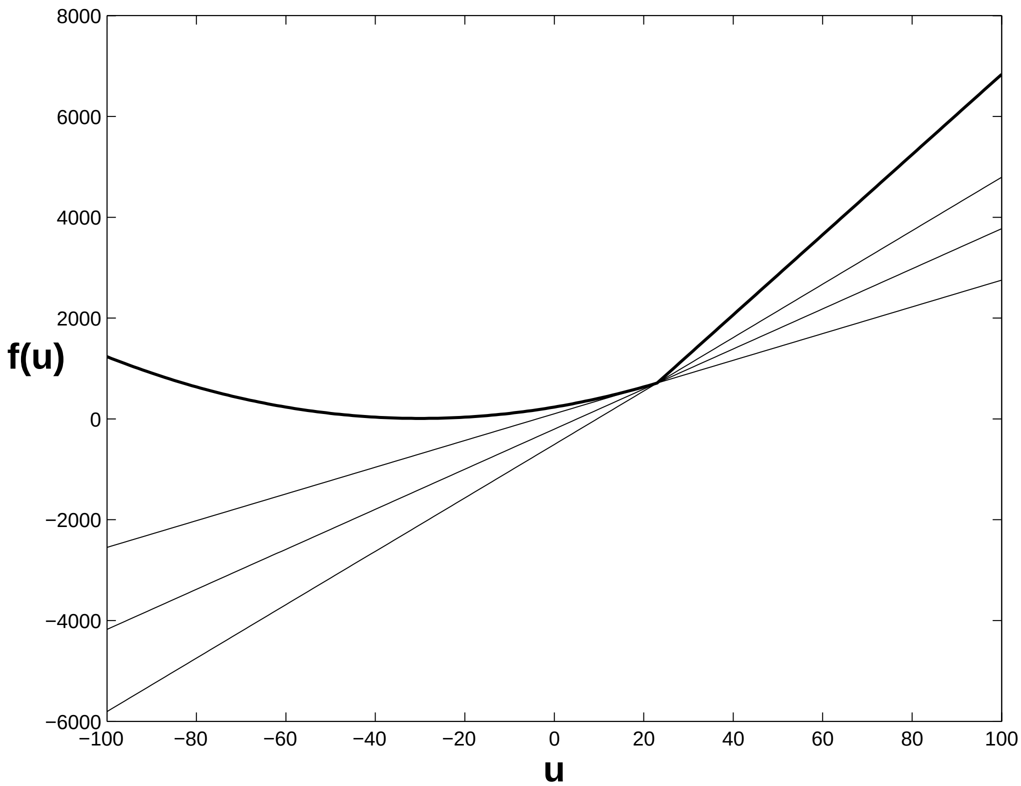

Figure 6: Subgradients and the subgradient inequality for a non-differentiable convex function.

4.2 Properties of Subgradients

Suppose that $f(x)$ is a convex function. For each x, let $\partial f(x)$ denote the set of all subgradients of $f(x)$ at x. We call $\partial f(x)$ the "subdifferential of $f(x)$."

- If $f(x)$ is convex, then $\partial f(x)$ is always a nonempty convex set.
- If $f(x)$ is differentiable, then $\partial f(x) = \{\nabla f(x)\}$.
- Subgradients plays the same role for convex functions as the gradient does for differentiable functions. Consider the following optimization problem:

$$\min_x f(x)$$

  - If $f(x)$ is convex and differentiable, then x is a global minimum if and only if $\nabla f(x) = 0$.
  - If $f(x)$ is convex and non-differentiable, then x is a global minimum if and only if $0 \in \partial f(x)$.

4.3 Subgradients for Concave Functions

If $f(x)$ is a concave function, then g is a subgradient of $f(x)$ at $x = \bar{x}$ if:

$$f(x) \leq f(\bar{x}) + g^T(x - \bar{x}) \quad \text{for all } x \in X \ .$$

This is illustrated in Figure 7. Figure 8 shows a piecewise-linear concave function. Figure 9 illustrates the subdifferential for a concave function.

4.4 Computing Subgradients

Subgradients play a very important role in non-differentiable optimization. In most algorithms, we assume that we have a subroutine that receives as input a value x, and has output g where g is a subgradient of $f(x)$.

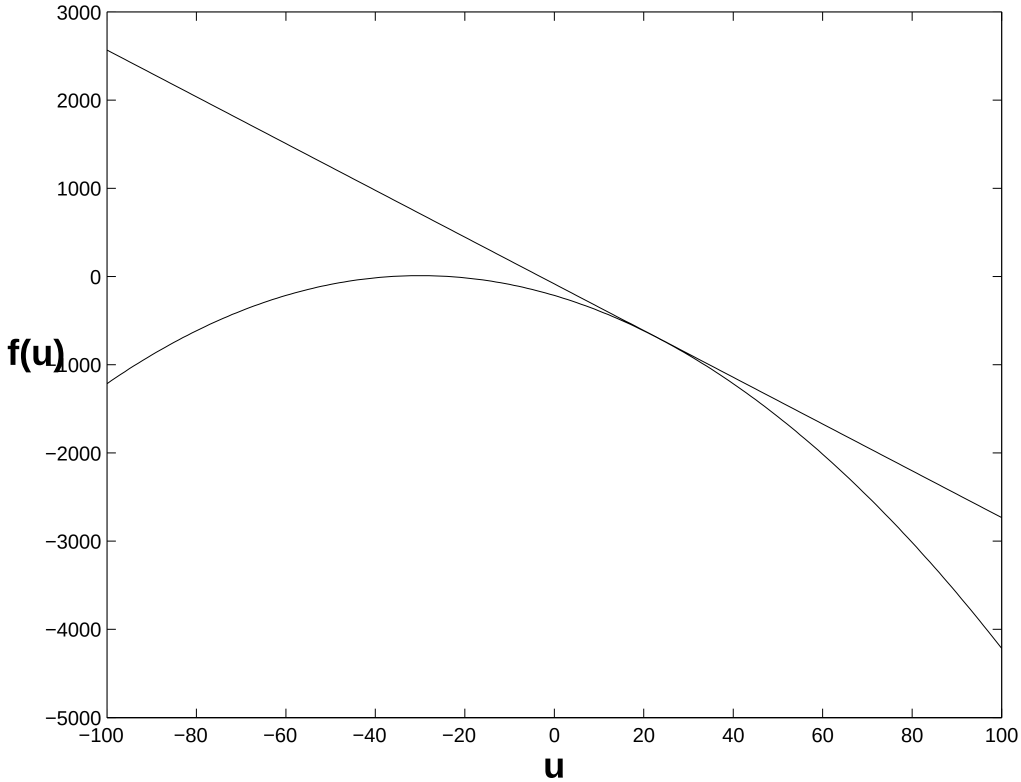

Figure 7: The subgradient of a concave function.

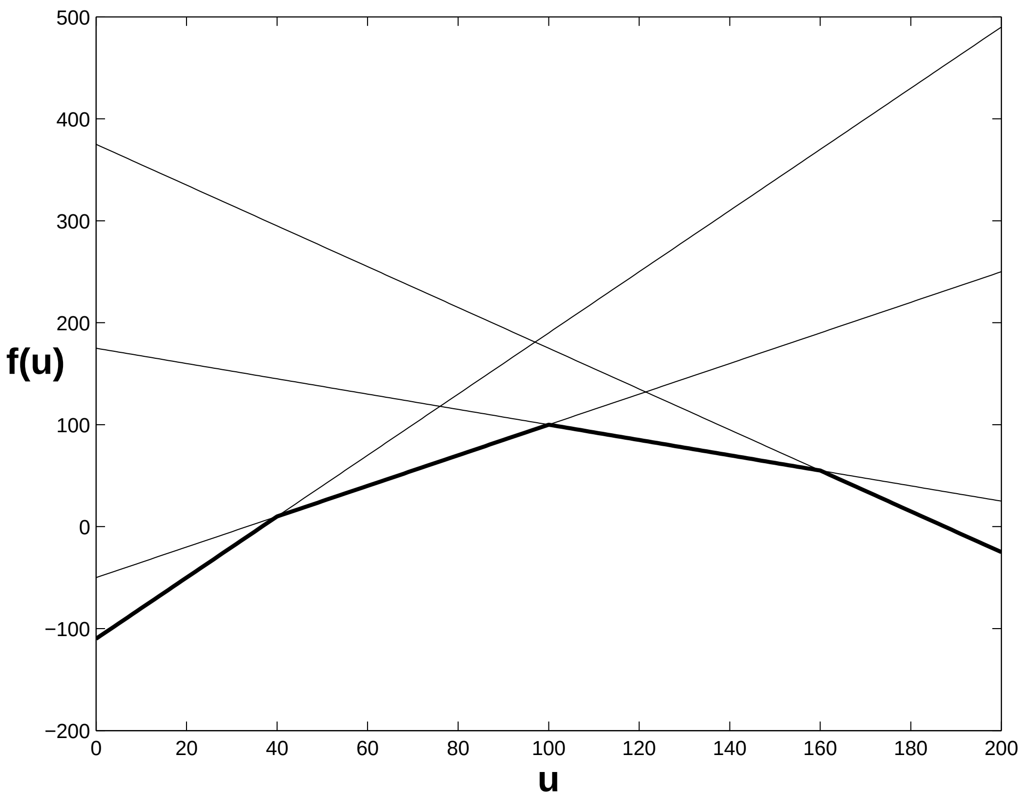

Figure 8: A piecewise linear concave function.

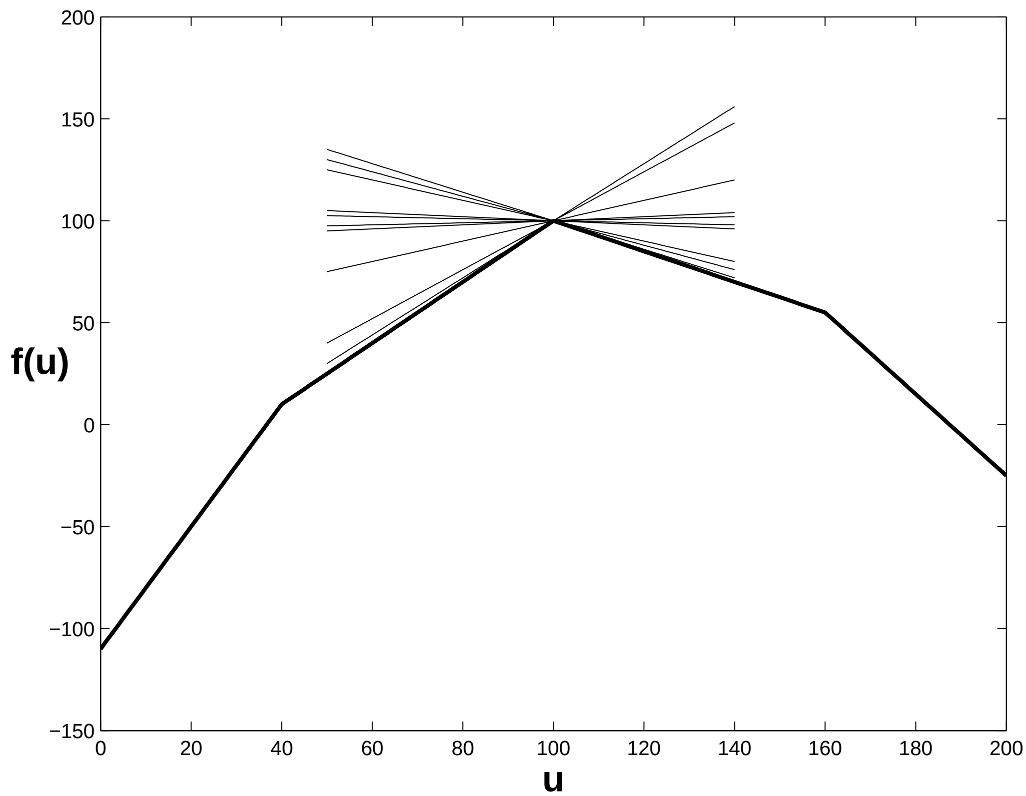

Figure 9: The subdifferential of a concave function.

5 The Subgradient Method for Maximizing a Concave Function

Suppose that $Z(u)$ is a concave function, and that we seek to solve:

$$P: \quad \text{maximize}_u \quad Z(u)$$

$$\text{s.t.} \qquad u \in \Re^n \ .$$

If $Z(u)$ is differentiable and $d := \nabla Z(\bar{u})$ satisfies $d \neq 0$, then d is an *ascent direction* at $\bar{u}$, namely

$$Z(\bar{u} + \epsilon d) > Z(\bar{u}) \text{ for all } \epsilon > 0 \text{ and sufficiently small .}$$

This is illustrated in Figure 10. However, if $Z(u)$ is not differentiable and g is a subgradient of $Z(u)$ at $u = \bar{u}$, then g is not necessarily an ascent direction. This is illustrated in Figure 11.

The following algorithm generalizes the steepest descent algorithm and can be used to maximize a nondifferentiable concave function $Z(u)$.

Step 0: Initialization. Start with any point $u^1 \in \Re^n$. Choose an infinite sequence of positive stepsize values $\{\alpha_k\}_{k=1}^{\infty}$. Set $k = 1$.

Step 1: Compute a subgradient. Compute $g \in \partial Z(u^k)$.

Step 2: Compute stepsize. Compute stepsize α_k from stepsize series.

Step 3: Update Iterate. Set $u^{k+1} \leftarrow u^k + \alpha_k \frac{g}{\|g\|}$. Set $k \leftarrow k+1$ and go to **Step 1**.

As it turns out, the viability of the subgradient algorithm depends critically on the sequence of stepsizes:

Property: Suppose that $\{\alpha_k\}_{k=1}^{\infty}$ satisfies:

$$\lim_{k \to \infty} \alpha_k = 0 \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k = \infty \ .$$

Then under very mild additional assumptions,

$$\sup_k Z(u^k) = \max_{u \in \Re^n} Z(u) \ .$$

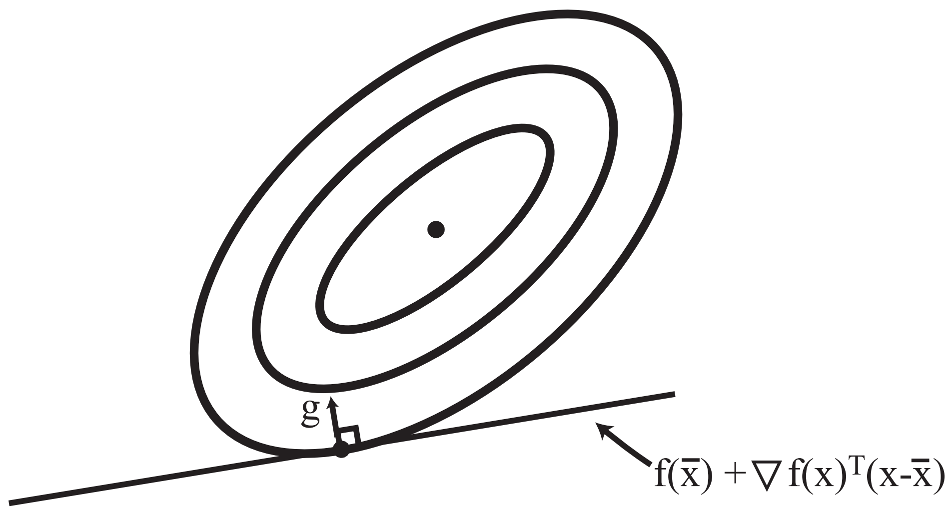

Figure 10: The gradient is an ascent direction.

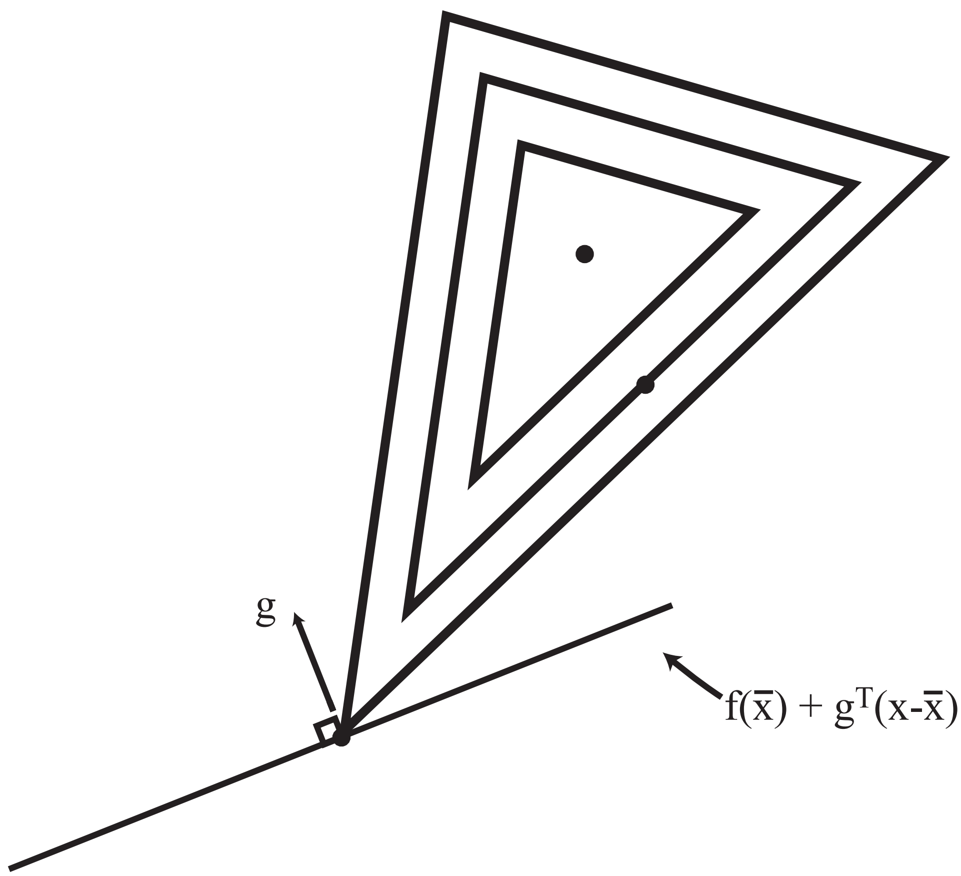

Figure 11: A subgradient is not necessarily an ascent direction.

5.1 Example of Subgradient Algorithm in One Variable

Consider the following concave optimization problem:

$$P: \quad \text{maximize}_u \quad Z(u) = \min\{0.5u + 2, -1u + 20\}$$

$$\text{s.t.} \qquad u \in \Re \ .$$

We illustrate various implementations of the subgradient method on this simple problem.

- Choose $u^1 = 0$ and $\alpha_k = \frac{0.14}{k}$. Figure 12 illustrates the performance of the subgradient algorithm for this stepsize sequence.
- Choose $u^1 = 0$ and $\alpha_k = 0.02$. Figure 13 illustrates the performance of the subgradient algorithm for this stepsize sequence.
- Choose $u^1 = 0$ and $\alpha_k = \frac{0.01}{k}$. Figure 14 illustrates the performance of the subgradient algorithm for this stepsize sequence.
- Choose $u^1 = 0$ and $\alpha_k = 0.01 \times (0.9)^k$. Figure 15 illustrates the performance of the subgradient algorithm for this stepsize sequence.

5.2 Example of Subgradient Algorithm in Two Variables

Consider the following concave optimization problem:

$$\begin{array}{llll} P: & \text{maximize}_u & Z(u) = \min\{ & 2.8571u_1 - 0.2857u_2 - 5.7143, \\ & & & -u_1 + u_2 + 2, \\ & & & -0.1290u_1 - 1.0323u_2 + 21.1613\} \\ & & & \\ & \text{s.t.} & u \in \Re^2 \ . & \end{array}$$

We illustrate the implementation of the subgradient method on this problem with $u^1 = (0, 0)$ and $\alpha_k = \frac{1}{\sqrt{k}}$. Figure 16 shows the function level sets and the path of iterations. Figure 17 shows the objective function values, and Figure 18 shows values of the variables $u = (u_1, u_2)$.

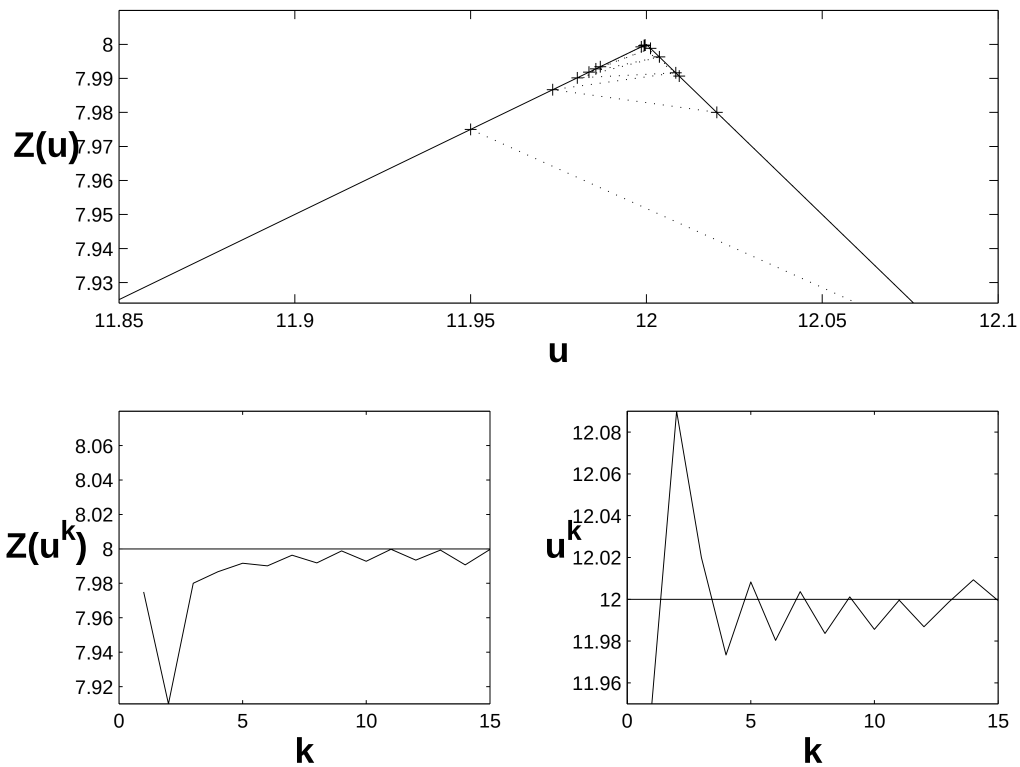

Figure 12: Illustration of subgradient algorithm, $\alpha_k = \frac{0.14}{k}$.

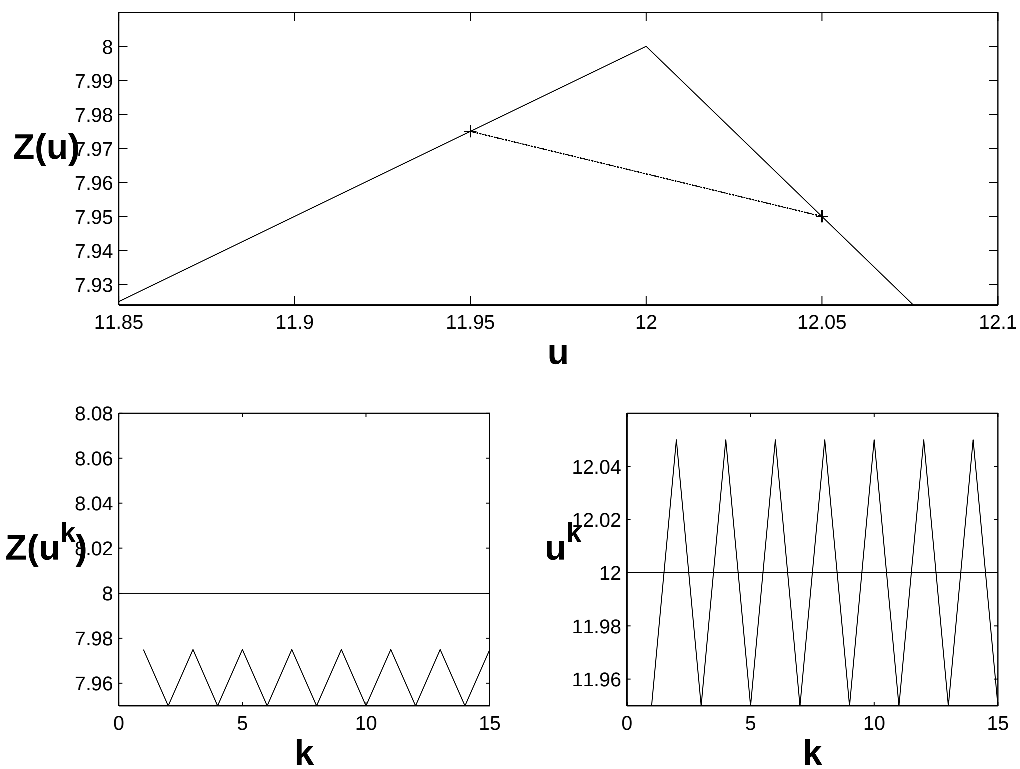

Figure 13: Illustration of subgradient algorithm, $\alpha_k = 0.02$.

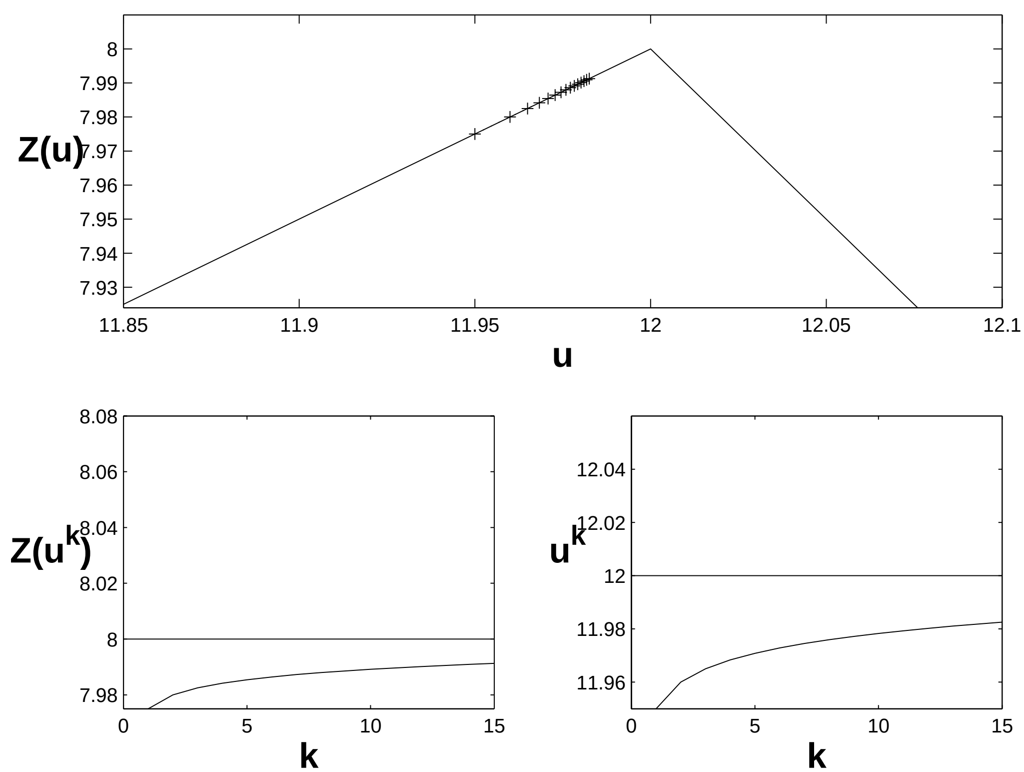

Figure 14: Illustration of subgradient algorithm, $\alpha_k = \frac{0.01}{k}$.

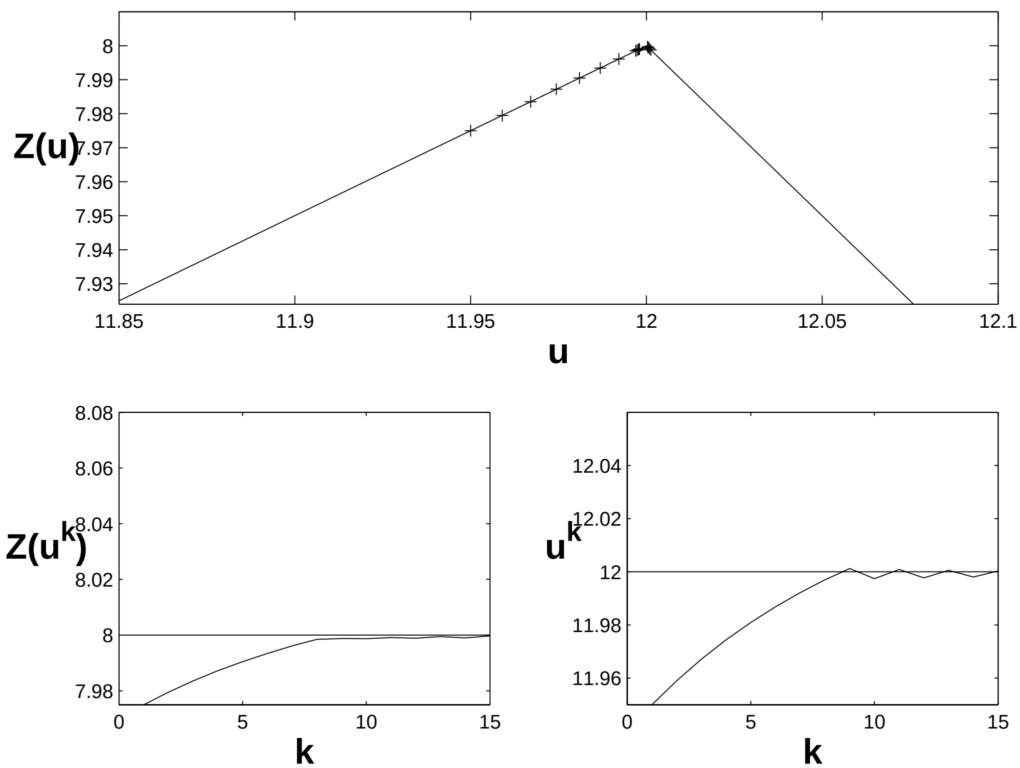

Figure 15: Illustration of subgradient algorithm, $\alpha_k = 0.01 \times (0.9)^k$.

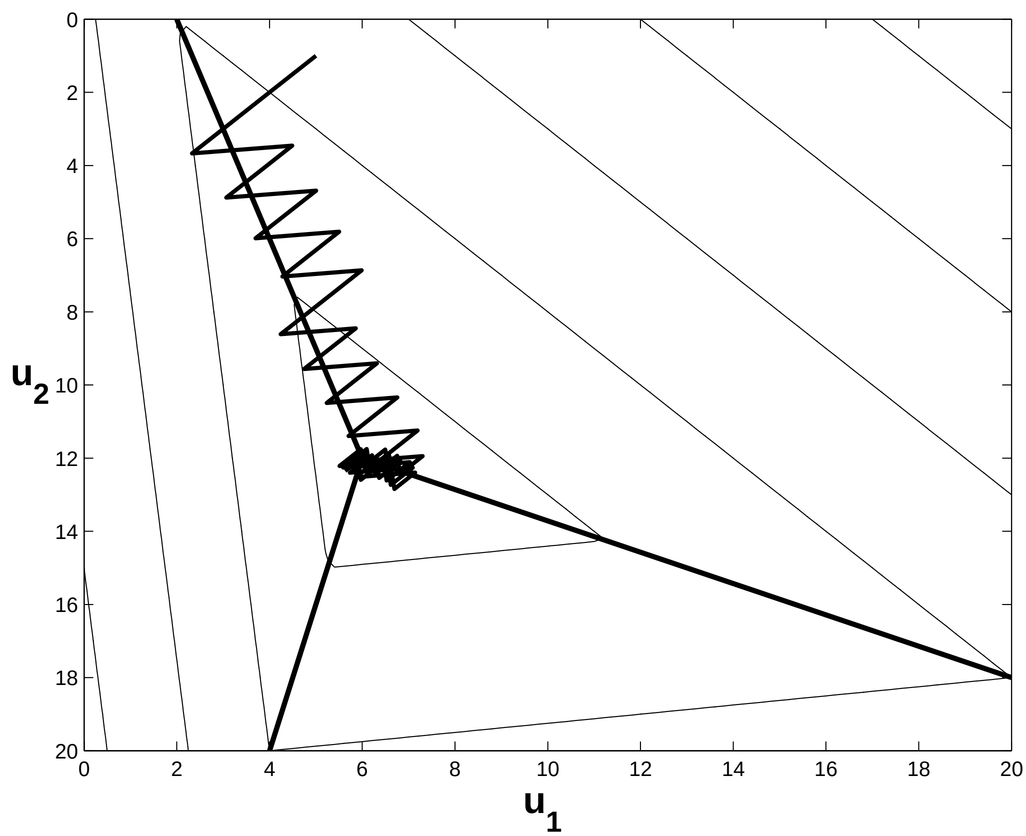

Figure 16: Illustration of the subgradient method in two variables: level sets and path of iterations.

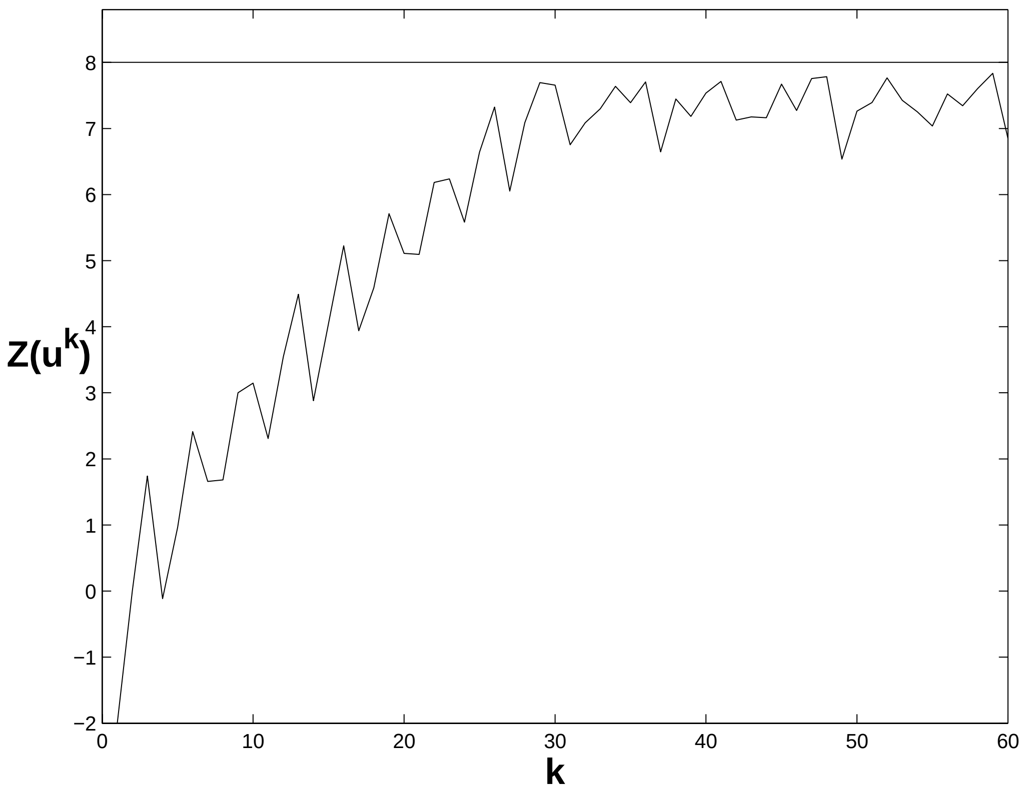

Figure 17: Illustration of the subgradient method in two variables: objective function values.

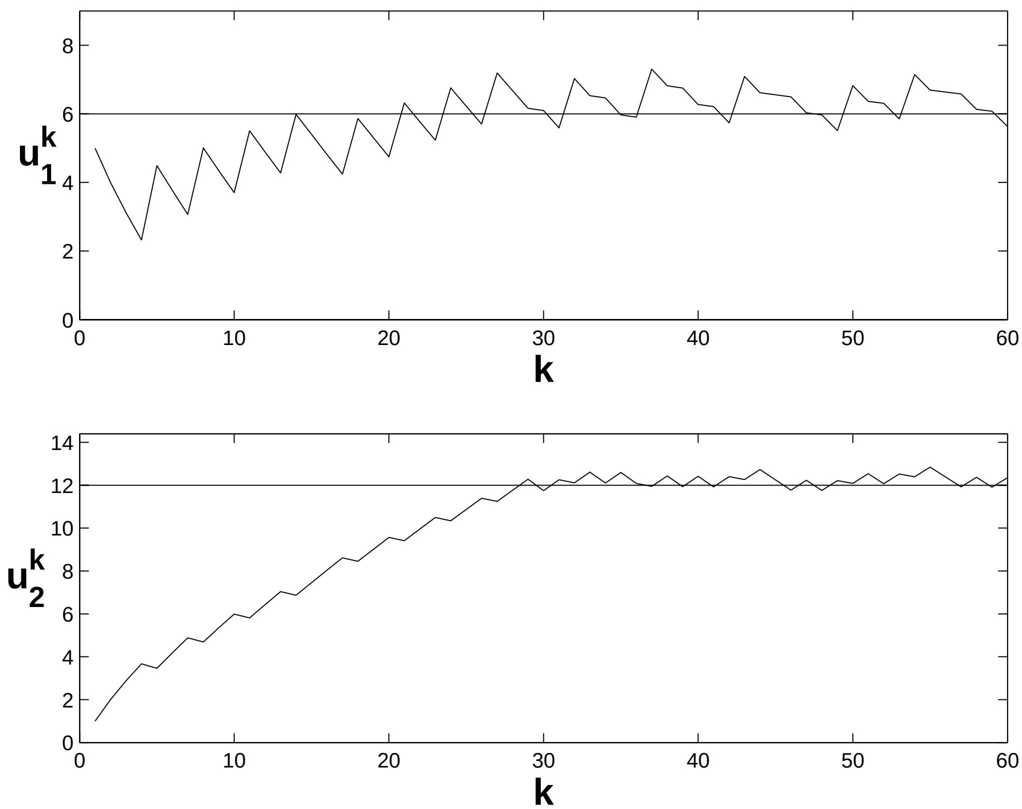

Figure 18: Illustration of the subgradient method in two variables: values of variables $u = (u_1, u_2)$.

6 Solution of the Lagrangian Dual via Subgradient Optimization

We start with the primal problem:

$$
\begin{array}{llrcll}
\mathrm{OP}: & \mathrm{minimum}_x & f(x) & & & \\
& \mathrm{s.t.} & g_i(x) & \leq & 0, & i = 1, \ldots, m \\
& & x \in P, & & &
\end{array}
$$

We create the Lagrangian:

$$L(x, u) := f(x) + u^T g(x)$$

and the dual function:

$$L^*(u) := \quad \mathrm{minimum}_{x \in P} \quad f(x) + u^T g(x)$$

The dual problem then is:

$$
\begin{array}{lll}
\mathrm{D}: & \mathrm{maximum}_u & L^*(u) \\
& \mathrm{s.t.} & u \geq 0
\end{array}
$$

Recall that $L^*(u)$ is a concave function. The premise of Lagrangian duality is that it is "easy" to compute $L^*(\bar{u})$ for any given $\bar{u}$. That is, it is easy to compute an optimal solution $\bar{x}$ of

$$L^*(\bar{u}) := \quad \mathrm{minimum}_{x \in P} \quad f(x) + \bar{u}^T g(x) \quad = \quad f(\bar{x}) + \bar{u}^T g(\bar{x})$$

for any given $\bar{u}$, where $\bar{x} \in P$. It turns out that computing subgradients of $L^*(u)$ is then also easy. We have:

Property: Suppose that $\bar{u}$ is given and that $\bar{x} \in P$ is an optimal solution of $L^*(\bar{u}) = \min_{x \in P} f(x) + \bar{u}^T g(x)$. Then $g := g(\bar{x})$ is a subgradient of $L^*(u)$ at $u = \bar{u}$.

Proof: For any $u \geq 0$ we have

$$\begin{aligned} L^*(u) = & \quad \min_{x \in P} f(x) + u^T g(x) \\ \leq & \quad f(\bar{x}) + u^T g(\bar{x}) \\ = & \quad f(\bar{x}) + \bar{u}^T g(\bar{x}) + (u - \bar{u})^T g(\bar{x}) \\ = & \quad \min_{x \in P} f(x) + \bar{u}^T g(x) + g(\bar{x})^T (u - \bar{u}) \\ = & \quad L^*(\bar{u}) + g^T (u - \bar{u}) \ . \end{aligned}$$

Therefore g is a subgradient of $L^*(u)$ at $\bar{u}$.
q.e.d.

The subgradient method for solving the Lagrangian dual can now be stated:

> **Step 0: Initialization.** Start with any point $u^1 \in \Re^n$, $u^1 \geq 0$. Choose an infinite sequence of positive stepsize values $\{\alpha_k\}_{k=1}^{\infty}$. Set $k = 1$.
>
> **Step 1: Compute a subgradient.** Solve for an optimal solution $\bar{x}$ of $L^*(u^k) = \min_{x \in P} f(x) + (u^k)^T g(x)$. Set $g := g(\bar{x})$.
>
> **Step 2: Compute stepsize.** Compute stepsize α_k from stepsize series.
>
> **Step 3: Update Iterate.** Set $u^{k+1} \leftarrow u^k + \alpha_k \frac{g}{\|g\|}$. If $u^{k+1} \not\geq 0$, re-set $u_i^{k+1} \leftarrow \max\{u_i^{k+1}, 0\}$, $i = 1, \ldots, m$. Set $k \leftarrow k + 1$ and go to **Step 1**.

Note that we have modified Step 3 slightly in order to ensure that the values of u^k remain nonnegative.

6.1 Illustration and Exercise using the Subgradient Method for solving the Lagrangian Dual

Consider the primal problem:

$$
\begin{aligned}
\text{OP}: \quad \text{minimum}_x \quad & c^T x \\
\text{s.t.} \quad & Ax - b \leq 0 \\
& x \in \{0,1\}^n \ .
\end{aligned}
$$

Here $g(x) = Ax - b$ and $P = \{0,1\}^n = \{x \mid x_j = 0 \text{ or } 1, j = 1, \ldots, n\}$.

We create the Lagrangian:

$$L(x,u) := c^T x + u^T (Ax - b)$$

and the dual function:

$$L^*(u) := \quad \text{minimum}_{x \in \{0,1\}^n} \quad c^T x + u^T (Ax - b)$$

The dual problem then is:

$$
\begin{aligned}
\text{D}: \quad \text{maximum}_u \quad & L^*(u) \\
\text{s.t.} \quad & u \geq 0
\end{aligned}
$$

Now let us choose $\bar{u} \geq 0$. Notice that an optimal solution $\bar{x}$ of $L^*(\bar{u})$ is:

$$\bar{x}_j = \begin{cases} 0 & \text{if } (c - A^T \bar{u})_j \geq 0 \\ 1 & \text{if } (c - A^T \bar{u})_j \leq 0 \end{cases}$$

for $j = 1, \ldots, n$. Also,

$$L^*(\bar{u}) = c^T \bar{x} + \bar{u}^T (A\bar{x} - b) = -\bar{u}^T b - \sum_{j=1}^{n} \left[(c - A^T \bar{u})_j \right]^- \ .$$

Also

$$g := g(\bar{x}) = A\bar{x} - b$$

is a subgradient of $L^*(\bar{u})$.

Now consider the following data instance of this problem:

$$A = \begin{pmatrix} 7 & -8 \\ -2 & -2 \\ 6 & 5 \\ -5 & 6 \\ 3 & 12 \end{pmatrix}, \; b = \begin{pmatrix} 12 \\ -1 \\ 45 \\ 20 \\ 42 \end{pmatrix}$$

and

$$c^T = \begin{pmatrix} -4 & 1 \end{pmatrix} .$$

Solve the Lagrange dual problem of this instance using the subgradient algorithm starting at $u^1 = (1, 1, 1, 1, 1)^T$, with the following step-size choices:

- $\alpha_k = \frac{1}{k}$ for $k = 1, \ldots$.
- $\alpha_k = \frac{1}{\sqrt{k}}$ for $k = 1, \ldots$.
- $\alpha_k = 0.2 \times (0.75)^k$ for $k = 1, \ldots$.
- a stepsize rule of your own.