明治大学大学院理工学研究科

２０１９年度

博士学位請求論文

Fixed Point Subgradient Methods for Constrained Nonsmooth Optimization

（制約付き非平滑最適化に対する不動点劣勾配法）

学位請求者　情報科学専攻

菱沼　和弘

数理最適化研究室 博士論文アーカイブ 2020, pp. 1–120.

Doctoral Thesis

Fixed Point Subgradient Methods for Constrained Nonsmooth Optimization

Research Fellow of Japan Society for the Promotion of Science;
Computer Science Course, Graduate School of Science and Technology, Meiji University
Kazuhiro HISHINUMA

This thesis considers constrained nonsmooth optimization problems, especially focusing on the situation in which the constraints are expressed as fixed points of some mapping. First, we propose incremental and parallel line-search subgradient algorithms for solving a constrained, nonsmooth, convex optimization problem. Next, we propose the fixed point quasiconvex subgradient method for solving a constrained, nonsmooth, quasiconvex optimization problem. After that, we discuss the rate of convergence of the fixed point quasiconvex subgradient method. For each method, we performed numerical experiments to compare it with existing methods and to evaluate its advantages and disadvantages. Through the provision of these novel methods and the discussion of their convergence and efficiency, we attempt to resolve issues such as the efficiency of the algorithm, difficulties in applying the algorithm, and limitations on solvable problems.

This thesis is part of the JSPS KAKENHI Research Project “Efficient methods for nonsmooth quasiconvex optimization with complicated constraints based on fixed point theory (Grant Number: JP17J09220).” Please see `https://kaken.nii.ac.jp/en/grant/KAKENHI-PROJECT-17J09220/` for details.

Contents

Chapter 1

Introduction

Section 1.1. Main problem and motivations behind it

This thesis considers constrained, nonsmooth optimization problems. A *constrained optimization problem* is composed of two factors: an objective functional and a constraint. The objective functional represents profit, time, potential energy, or some combination of these quantities, and the constraint represents what the solution must satisfy, such as budgetary constraints in an economic problem or shape constraints in a design problem [68, Chapter 1]. For given objective functional and constraints, the goal of a constrained optimization problem is to find a solution that minimizes or maximizes the objective functional and that satisfies all the constraints. We call a constrained optimization problem whose objective functional is nondifferentiable a constrained, nonsmooth optimization problem. This thesis especially focuses on the situation in which the constraints are expressed as fixed points of some mapping.

This thesis mainly deals with the following two kinds of the constrained, nonsmooth optimization problem:

- in which the objective functional is the sum of convex functionals;
- in which the objective functional is a quasiconvex functional.

Let us see the motivations behind each of them in order. There are many instances of optimization problems whose objective functional can be expressed as the sum of convex functionals [13, 79, 83]. The optimization task appearing in learning with a support vector machine is a typical instance [83]. The goal of the learning is to make a classifier capable of correctly predicting the label for each given data. To reach this goal, the task minimizes a objective functional that expresses the degree of misclassification for each training data. To obtain a classifier that can correctly predict all of the given data, the task minimizes the sum of these objective functionals. Similarly to learning with a support vector machine, the task of multilayer neural networks also forms an objective functional summing a number of functionals [30]. Signal recovery [10], bandwidth allocation [39], and beamforming [85] are also instances of optimization problems minimizing the sum of nonsmooth, convex functionals with some constraints. Hence, providing a method that can solve them

efficiently is important.

Let us examine the motivations behind each of them. There are many instances of optimization problems whose objective functional can be expressed as the sum of convex functionals [13, 79, 83]. The optimization task appearing in learning with a support vector machine is a typical instance [83]. The goal of the learning is to make a classifier capable of correctly predicting the label for each given data. To reach this goal, the task minimizes an objective functional that expresses the degree of misclassification for each training data. To obtain a classifier that can correctly predicts all of the given data, the task minimizes the sum of these objective functionals. Similarly to learning with a support vector machine, the task of multilayer neural networks also forms an objective functional summing a number of functionals [30]. Signal recovery [10], bandwidth allocation [39], and beamforming [85] are instances of optimization problems that involve minimizing the sum of nonsmooth, convex functionals with certain constraints. Hence, finding methods that can solve them efficiently is important.

There are many cases in which the objective functional cannot be expressed as a convex functional [33, 35, 37, 38, 50, 86]. Some of them belong to the class of quasiconvex optimization problems whose objective functional is not convex but that inherits several properties of convex functionals [33, 35, 86]. A typical instance of a quasiconvex functional is a fractional functional. This functional is expressed as a fractional of two functionals and is used for modeling ratio indicators, such as the debt/equity in financial and corporate planning, inventory/sales and output/employee in production planning, and cost/patient and nurse/patient ratios in healthcare and hospital planning [86]. We cannot use all the properties of convex functionals for solving these problems, but some of them give us clues on how to tackle them. Furthermore, studies on quasiconvex functionals shed light on convex optimization problems, since any convex functional is also a quasiconvex functional. Hence, this thesis studies the development of a method for solving constrained, quasiconvex optimization problems.

Section 1.2. Review of the existing studies

Let us review the existing studies for clarifying the contributions of the later discussion. Throughout this thesis, we tackle the construction and analysis of *iterative methods* [68, Chapter 3]. An iterative method is one that generates a sequence converging to some solution of a given problem by an iterative procedure. Here, we study their fundamental instances and variants.

1.2.1 Iterative methods for unconstrained optimization

Steepest descent method. First, let us consider how to solve an unconstrained optimization problem wherein the task is to minimize a given objective functional over the whole space. Consider the case in which the minimizer of the objective functional cannot be computed explicitly, such as a complicated objective functional. If we can

easily find a minimizer of the objective functional, there is no problem. Hence, let us consider the case where we cannot. If the objective functional is differentiable, the most basic approach to solving the problem is the *steepest descent method* [68, Section 3.3].

The steepest descent method iterates the current approximation to decrease the value of the objective functional. First, we give an initial point to the method. If we have some information on the problem setting, we can use it for deciding the initial point. If we don't have any information, we can pick the initial point in some way, such as by generating it randomly. After that, the method improves it by using the information on the gradient. The behavior of the steepest descent method is illustrated in Figure 1.1. Suppose that the rightmost point on the horizontal axis

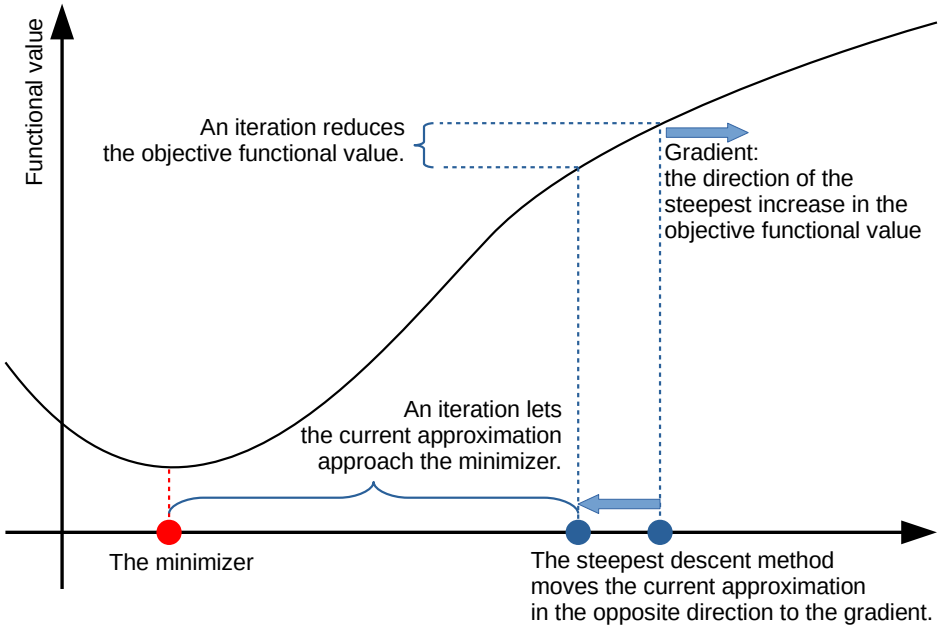


Fig. 1.1: Illustration of the behavior of the steepest descent method

is the current approximation and the curved graph shows the objective functional. As shown by the blue arrow pointing to the right in Figure 1.1, the gradient at this approximation shows the direction of the steepest increase in the objective functional value. This implies that the the functional value should decrease by moving the current approximation in the opposite direction to the gradient. Indeed, Figure 1.1 shows that an iteration which moves the approximation in the opposite direction to the gradient actually reduces the objective functional value. The opposite direction to the gradient like this is called the *(steepest) descent direction* [68, Section 2.2]. The principal idea of the steepest descent method is to minimize the objective functional value by iterating this improvement.

There is a question of whether the sequence generated by the steepest descent method always converges to the minimizer of the objective functional. The answer to this question is no. Figure 1.2 shows an example in which the sequence does not

converge. In the first several iterations, the steepest descent method reduces the

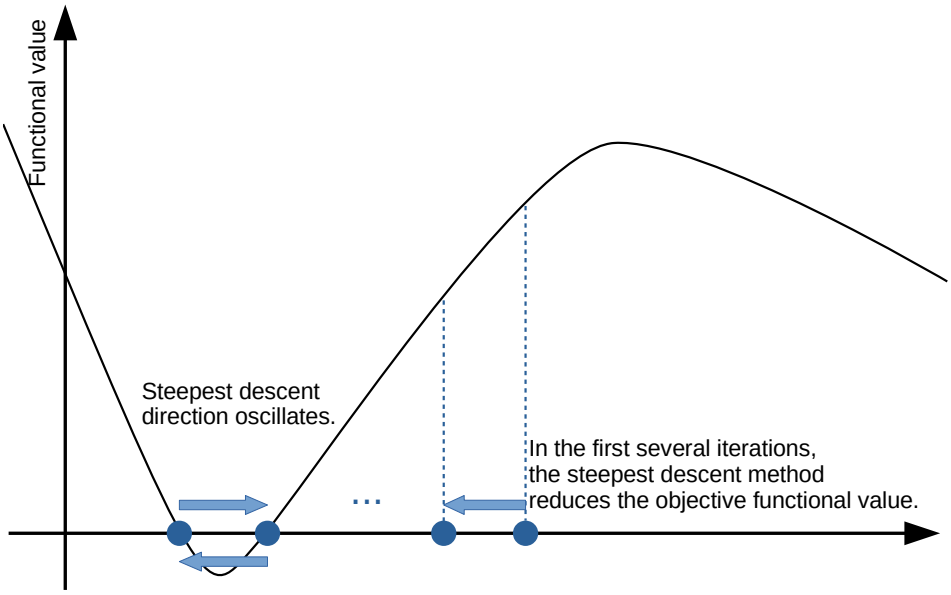
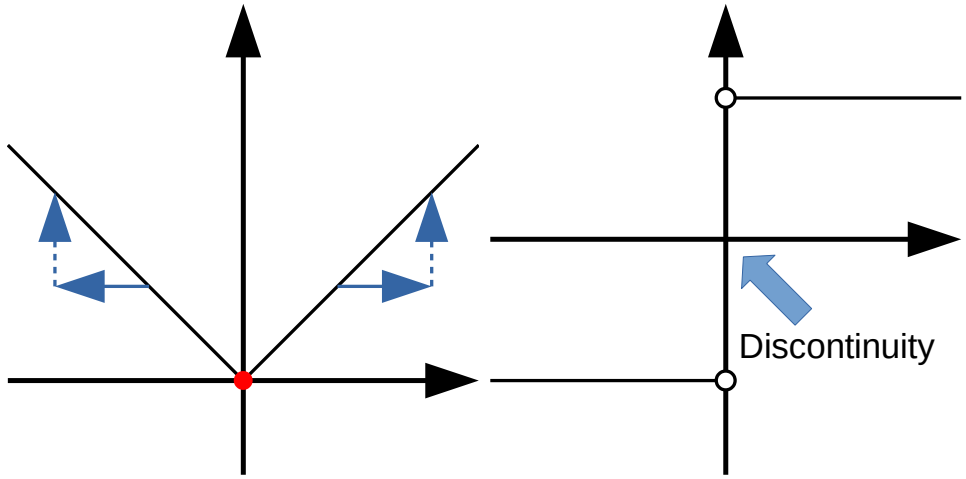


Fig. 1.2: A case in which the generated sequence repeats two points and does not converge to the minimizer

value of the objective functional. However, after that, the steepest descent method eternally generates a nonconvergent sequence so as to move approximations to their symmetric point with respect to the minimizer. This example shows that we cannot ensure convergence of the generated sequence to the minimizer only by using the direction of the gradient.

To guarantee convergence of the generated sequence to the minimizer, we can use other information in the gradient besides its direction, i.e., its norm. The norm of the gradient shows by how much the functional value changes at the point. If the norm of the gradient is large, we can consider the objective functional value to be dramatically changing in the neighborhood of the current approximation. After making assumptions about the continuity of the objective functional, this implies that the current approximation is still far from the minimizer. On the other hand, we can consider the current approximation to be possibly close to the minimizer when the norm of the gradient is small. Hence, we can overcome the issue shown in Figure 1.2 by moving the current approximation in a direction determined from the gradient by the distance proportional to the norm of the gradient. Specifically, if the objective functional is Lipschitz continuous, which is a kind of smoothness condition, and if it is convex, i.e., its epigraph (the area above the graph of the functional) is convex, the generated sequence converges to the minimizer of the objective functional by using the direction determined from an appropriately scaled gradient [36, Proposition 2.3], [44, Definition (1.16)].

Subgradient method. Next, let us consider the case in which the objective functional may not be differentiable at some point. The simplest example of this case is the absolute value functional. As shown in Figure 1.3, the absolute value functional is not differentiable at the vertex. The absolute value functional is differentiable if we

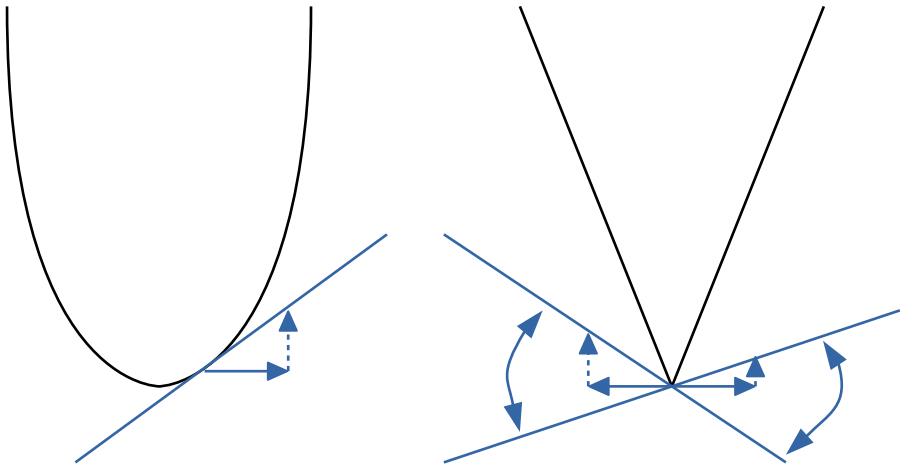


(a) Functional values (b) Gradients

Fig. 1.3: The gradients on the left side of the vertex are vectors whose norms are constant and directions are towards the left, and the gradients on the right side of the vertex are vectors whose norms are constant and directions are to the right. Since these two gradients are not equal to each other at the vertex, the absolute value functional cannot be differentiated at the vertex.

exclude its value at the vertex. On the left side of the vertex, the norm of the gradient is constant and the direction of the gradient is to the left. On the right side of the vertex, the norm of the gradient is also constant and the direction of the gradient is to the right. However, the gradient at the vertex cannot be defined since the values of the right and left gradients are not equal at the vertex, as shown in Figure 1.3b. Hence, we cannot use the steepest descent method, which uses the gradient of the objective functional for generating the sequence, to minimize a nondifferentiable functional such as this example.

There is an optimization method that overcomes this issue, called the *subgradient method* [5, Section 8.2]. Instead of the gradient, which is defined for differentiable functionals, the subgradient method uses subgradients, which can be defined even if the functional is not differentiable. The subgradient is an extension of the gradient, and it is defined from the viewpoint of the relationship between the gradient and the tangent hyperplane. Here, let us consider the gradient of a convex functional. As shown in Figure 1.4a, the gradient at a given point draws a hyperplane which touches the functional at that point, whose gradient coincides with the gradient of the functional at that point, and which lies below the functional [81, Theorem 2.14]. It is defined as shown in Figure 1.4b so that the subgradient inherit these properties



(a) A gradient (b) Subgradients

Fig. 1.4: Extending the gradient concept to that of subgradients

from the gradient: a gradient of the hyperplane that is a tangent hyperplane of the functional and lies below the functional is called the subgradient [2, Definition 8.3 and Proposition 8.12]. Numerous subgradients may exist at the same point. Indeed, an infinite number of subgradients exist at the vertex in Figure 1.4b. Therefore, we usually consider the set of all subgradients at the given point and call it the *subdifferential* [87, Section 7.3]. Notice that subgradients at a given point may have a variety of norm values since numerous subgradients may exist at the same point, but it still describes the degree of the gradient of the functional. In fact, even at the vertex in Figure 1.4b, a vector with too large a norm cannot be a subgradient of the absolute functional because it violates the condition that the tangent hyperplane drawn by the subgradient must lie under the functional. This property will be important when we discuss the nonconvex case later.

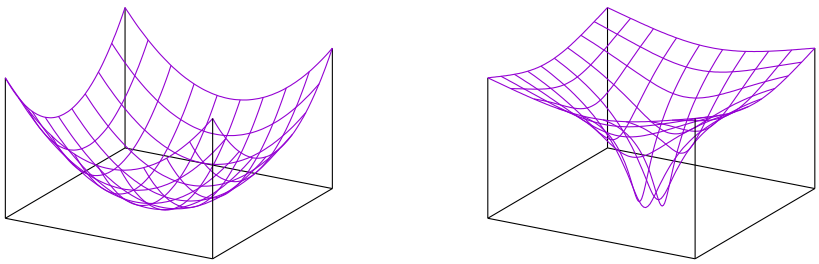
In a similar way to the steepest descent method, we can generate a sequence converging to the minimizer of a nondifferentiable convex objective functional by using the suitably scaled descent direction calculated from an arbitrarily chosen subgradient from the subdifferential at each approximation [5, Proposition 8.2.6]. We call this way of generating this sequence the subgradient method. Figure 1.5 summarizes the steepest descent method and the subgradient method.

Quasiconvex subgradient method. At the end of this subsection, let us consider the problem of applying the subgradient method to a nonconvex functional. As an instance of nonconvex functionals, we will discuss quasiconvex functionals in particular. In contrast to convex functionals, the epigraph of a quasiconvex functional does not have to be convex. Here, we will put off giving the general definition of quasiconvexity till later and limit ourselves here to a typical instance, as shown in Figure 1.6. As described above, the epigraph of a convex functional is convex (Figure 1.6a). By

For minimizing a differentiable objective functional:	*For minimizing a nondifferentiable convex objective functional:*
Step 1: Choose an initial point.	Step 1: Choose an initial point.
Step 2: Compute the gradient.	Step 2: Compute the subdifferential.
	Step 3: Choose a subgradient from the subdifferential.
Step 3: Improve the current approximation with the descent direction obtained from the gradient.	Step 4: Improve the current approximation with the descent direction obtained from the chosen subgradient.
Step 4: Go to Step 2.	Step 5: Go to Step 2.

(a) Steepest descent method (b) Subgradient method

Fig. 1.5: Steepest descent method and subgradient method



(a) Convex functional (b) Quasiconvex functional

Fig. 1.6: Typical examples of convex and quasiconvex functionals

contrast, a quasiconvex functional like the one shown in Figure 1.6b may have a line segment that violates the convexity of the epigraph (see Figure 1.7). However, the quasiconvexity of a functional is an extension of convexity from some viewpoint. Let us consider the contour graph (Figure 1.8) of the functionals we saw in Figure 1.6. Figure 1.8a is the contour graph of the convex functional in Figure 1.6a, and Figure 1.8b is the contour graph of the quasiconvex functional in Figure 1.6b. Both graphs draw similar concentric circles whose center is the origin, and the only difference is in the number density of the circles. One of the most important properties of quasiconvex functionals is the convexity of its slices, as we saw in Figure 1.8b [19, Proposition 4.8]. In other words, we can say that the quasiconvexity of a functional is an extension of the concept of convexity from the viewpoint of the contour graph.

Let us consider the difficulty of minimizing a quasiconvex objective functional by directly using the idea of steepest descent and/or subgradient. Here, the optimization requires computation of the gradient or the subgradient. However, the usual

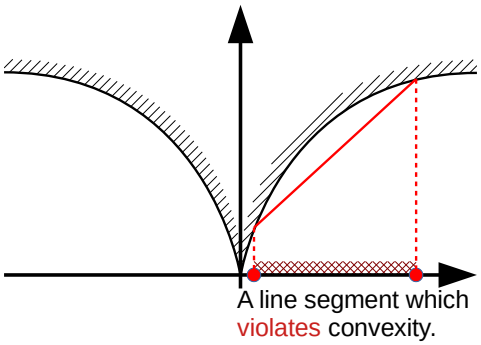
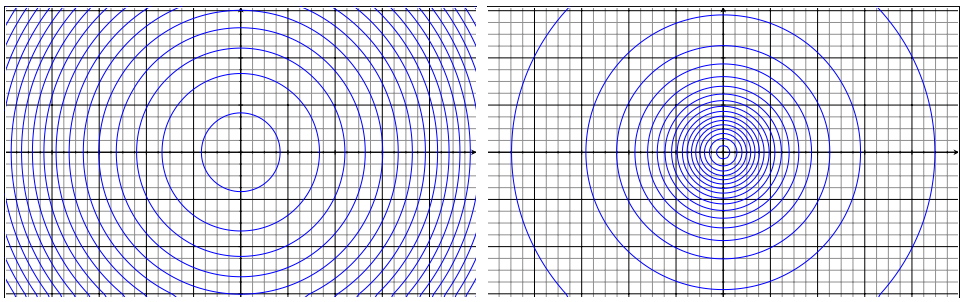


Fig. 1.7: Quasiconvex functional may violate the convexity of its epigraph



(a) Convex functional (b) Quasiconvex functional

Fig. 1.8: Contour graphs of the examples shown in Figure 1.6

subdifferential for the convex functional may be empty, i.e., it may not be available, for the quasiconvex functional, since it might be the case that no subgradient exists, as is shown in Figure 1.11. In this case, we cannot draw any tangent hyperplane

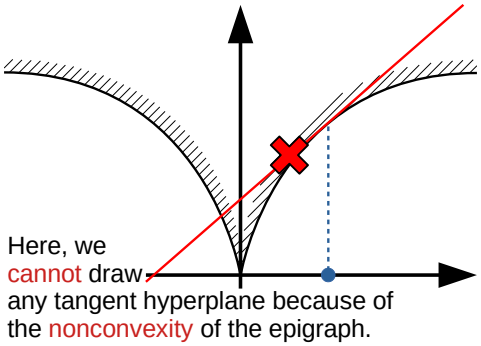


Fig. 1.9: Counterexample showing a quasiconvex functional that is not subdifferentiable

running below the quasiconvex objective functional at the given point because the nonconvexity of the functional does not guarantee the convexity of its epigraph.

Of course, there are various, extended subdifferentials for nonconvex functionals

[74, Section 4], [81, Definition 8.3]. Some of them can be defined for quasiconvex functionals. A subdifferential constructed from directional derivatives [74, Section 4] is one. Let us consider a piecewise-linear quasiconvex functional such as the one shown in Figure 1.10 and the subdifferential at a point at which the functinal is not differentiable. From the viewpoint of the blue point, the functional value sharply

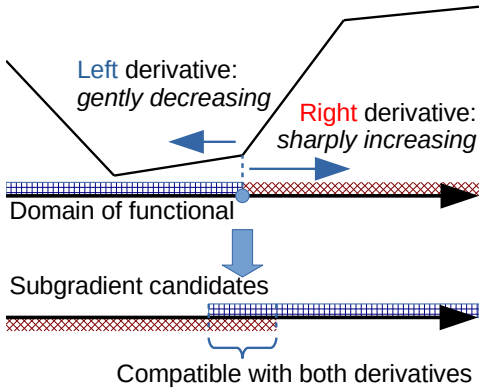


Fig. 1.10: Subdifferential constructed from the directional derivatives

increases on the right side of the blue point while it is gently decreasing on the left side. Similarly to the subdifferential for convex functionals, let us construct a subdifferential by using the notion of the tangent hyperplane. If we consider only the right side, the gradient of the tangent hyperplane is bounded from above by the directional derivative on that side. If we consider only the left side, the gradient of the tangent hyperplane is bounded from below by the directional derivative on that side. Hence, we can obtain gradients that are compatible with both cases; we call the set of them the subdifferential constructed by the directional derivatives [74, Section 4].

When we can obtain a subdifferential for quasiconvex functionals, can we always run the subgradient method for quasiconvex optimization? Unfortunately, knowing the subdifferential is not enough, and Figure 1.11 shows why. This figure illustrates the behavior of the subgradient method when it is applied to a piecewise-linear quasiconvex minimization. Unlike a convex functional, a quasiconvex functional may have a flat part even if it is not a (set of) minimizer; in other words, a local minimizer might not coincide with the global minimizer in quasiconvex optimization. When the approximation reaches the flat part, the subdifferential reduces to a set containing only the zero vector. Since the zero vector has no information on the direction, the method cannot use it to improve the approximation. Hence, the method terminates even if the approximation is far from the minimizer.

As a way of overcoming this issue, Konnov [54] introduced a variant of the subgradient method for minimizing the quasiconvex objective functional. This variant uses the normal vector to the slice at the current approximation as a subgradient. Let us examine the behavior of this method (Figure 1.12) when it is used to minimize the functional in Figure 1.11. The slice at the current approximation is defined as

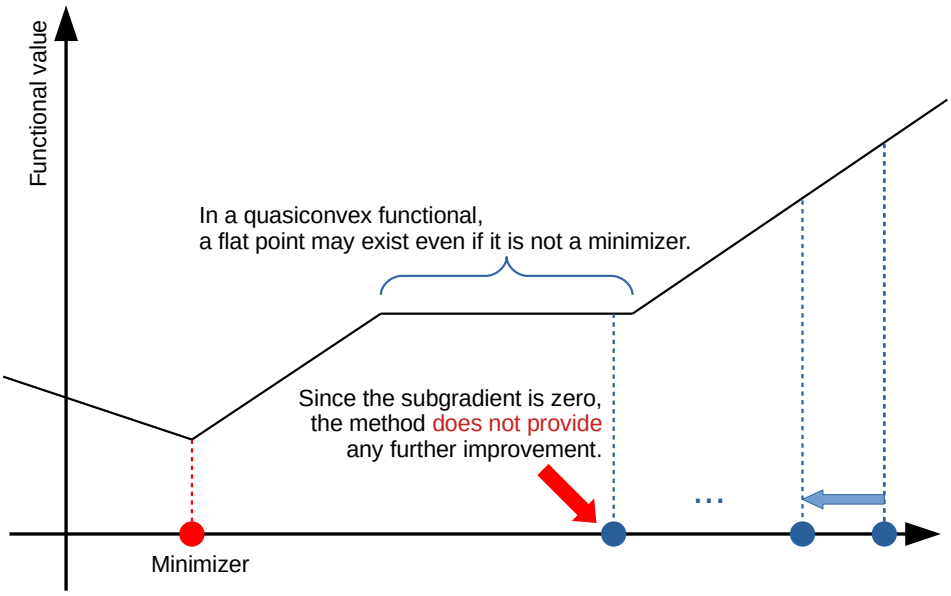


Fig. 1.11: Termination problem of the subgradient method in quasiconvex optimization

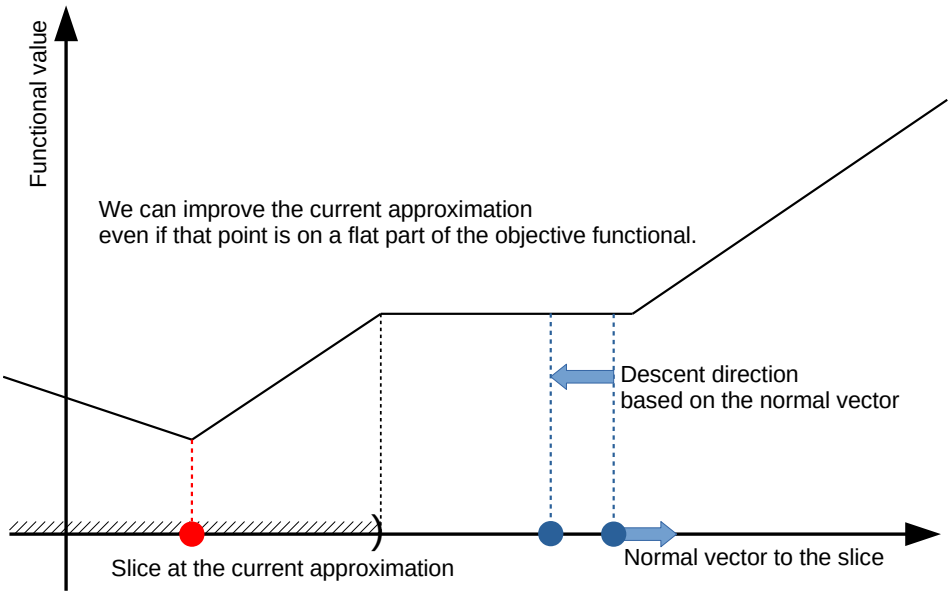


Fig. 1.12: Termination problem occurring when the subgradient method is applied to quasiconvex optimization

the area in which the functional value is less than the current one. As we saw in Figure 1.8, every slice (which is expressed as a contour line in the contour graph) of a quasiconvex functional is convex. Here, the slice does not contain the current approximation. Therefore, from the separation theorem [87, Corollary 5.3.3], there exists a half-space that includes the slice and does not contain the current approximation, such as illustrated in Figure 1.13. This implies that there exists at least one nonzero

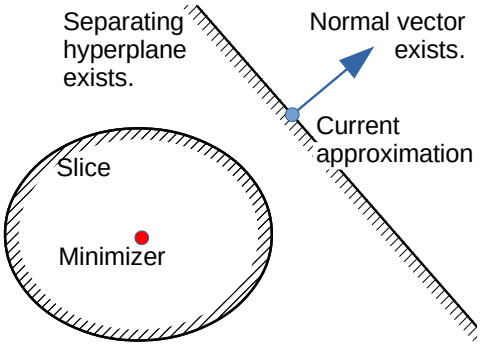


Fig. 1.13: Existence of a normal vector to the slice, from the separation theorem

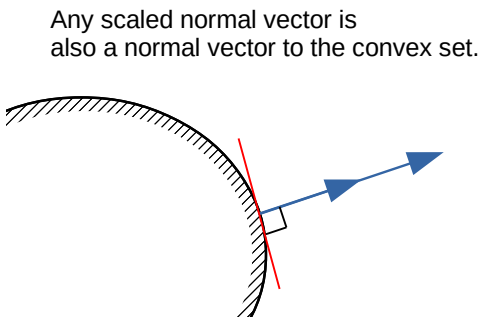


Fig. 1.14: Any scaled normal vector to the convex set is also a normal vector to it

subgradient if the functional is continuous [53, Lemma 3]. This in turn means that the method can continue to improve the approximation even when it is on the flat part of the objective functional. Furthermore, the minimizer is contained in each slice whenever the current approximation is not the minimizer. Since the descent direction based on the normal vector always points to the minimizer, the sequence generated by this method with an appropriate scaling of the descent direction converges to the minimizer under a certain assumption on the continuity of the objective functional [54, Corollary 2.3].

Here, recall that we use the norm of the gradient to ensure the sequence generated by the steepest descent method converges to the minimizer. However, any scaled normal vector to the slice is also a normal vector to it (as shown in Figure 1.14). This implies that the norm of the subgradient which is defined as the normal vector to the slice has no longer has any meaning. Hence, for the generated sequence to converge, we have to make an additional assumption or incorporate some other mechanism into the subgradient method.

1.2.2 Fixed point theory and constrained optimization

Metric projection. In this subsection, let us consider a way to solve constrained optimization problems. Before discussing concrete methods for solving them, let us see what tasks they can model. Suppose that as the manager of a factory, you are charged with finding production factors such as how many of each item should be produced [33, Section 6]. Here, it would be good plan if you could minimize the cost

of production and maximize the total profit. However, we usually have restrictions on the feasible plan, such as the duties assigned to each production project and limitations on resources. Hence, to solve such a problem, we need to optimize the objective functional subject to constraints. This sort of problem is called *constrained optimization* [68, Chapter 1].

The relationship between the objective functional and the constraint set is shown in Figure 1.15. The constraints of the optimization problem can be expressed as a subset

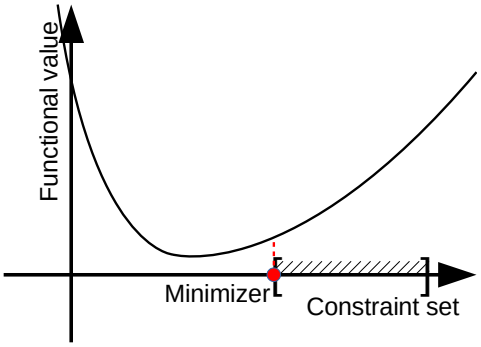


Fig. 1.15: Constrained optimization

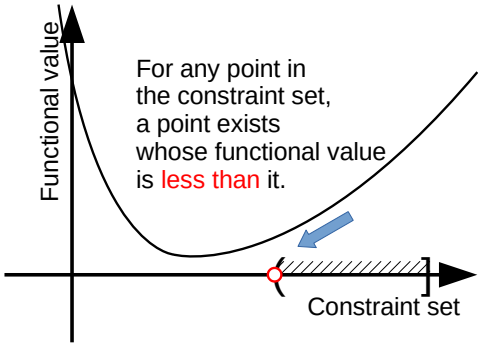


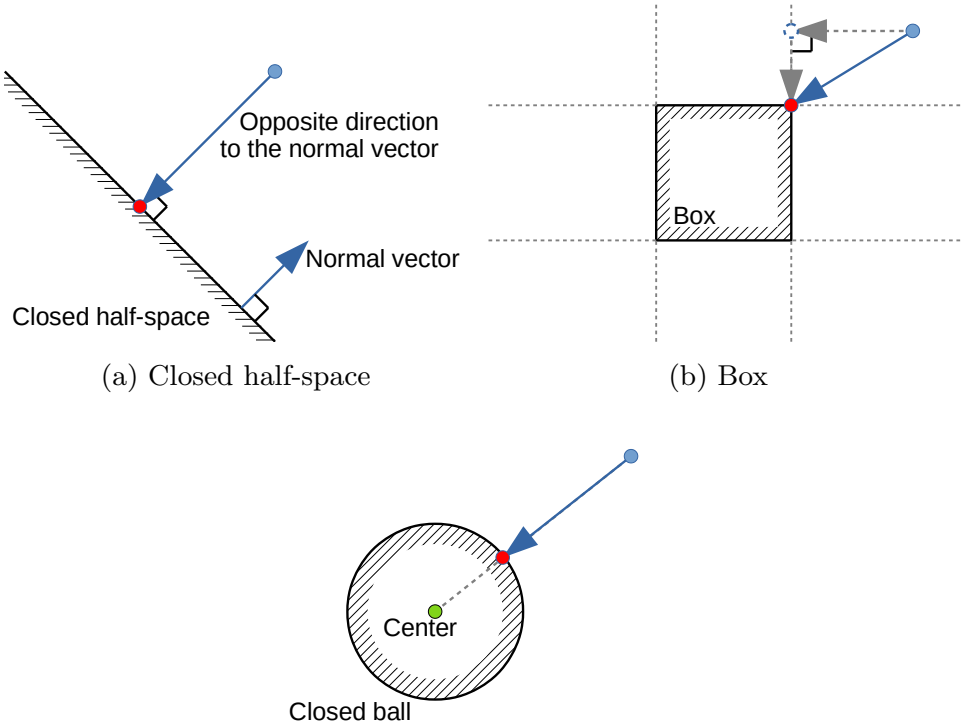
Fig. 1.16: Case of a non-existent minimizer

of the domain of the objective functional. This subset is called the *constraint set*. The task of the constrained optimization problem is to find a point whose functional value is the infimum on the constraint set. Obviously from the figure, the minimizer of the constrained optimization problem may not coincide with the vertex of the objective functional.

Throughout this thesis, we will assume that the constraint set is a nonempty, closed, convex set if it is given. The reason why we assume the constraint set is closed is to avoid corner cases where the minimizer does not exist. Let us refer to Figure 1.16. If the constraint set is not closed, the boundary of the constraint set may not be contained in it. In this case, the minimizer may not exist, because for any point in the constraint set, there is another point whose objective functional value is strictly less than that one. To ensure the existence of the minimizer of the constrained optimization problem, we need another assumption such as coerciveness of the objective functional [87, Theorem 7.2.2] or boundedness of the constraint set [87, Theorem 7.2.3]. However, closedness and convexity are required in both cases. Hence, we will assume that the constraint set is closed and convex.

Here, let us consider how to solve a constrained optimization problem with the methods described above. The simplest way is to use the *metric projection* onto the constraint set. The metric projection is a mapping which sends the given point to the nearest point belonging to the constraint set [87, Section 5.2]. For any nonempty, closed, convex set and for any point, the existence of such a point is guaranteed, as is its uniquess [87, Theorem 5.2.1]. Here, Figure 1.17 shows some examples of metric projections onto simple, closed convex sets, i.e., a closed half-space, a box, and a

closed ball. If the given point is contained in the set, the metric projection does



(a) Closed half-space

(b) Box

(c) Closed ball

Fig. 1.17: Metric projections onto the simple, closed, convex sets

not move it. Let us consider the case where the given point is out of the set. The direction of moving the given point to the closed half-space over the shortest distance is the opposite one to the normal vector to the set. Hence, the metric projection onto the closed half-space moves the given point in the opposite direction to the normal vector by an appropriate distance (as shown in Figure 1.17a; [2, Example 29.20]). To compute the metric projection onto a box, we can use the properties of orthogonality (as shown in Figure 1.17b [2, Proposition 29.6]). The metric projection of a given point onto the closed ball is on the line segment connecting it and the center of the ball. Hence, as shown in Figure 1.17c, we can compute it explicitly [2, Example 3.18 and Proposition 3.19].

The metric projection onto the constraint set sends the given point to the nearest point, i.e., the best approximate point, in the constraint set. Hence, by projecting the sequence generated by the steepest descent method onto the constraint set, we can obtain a sequence which is contained in the constraint set and which approximates the generated sequence. This method is called the *projected gradient method* [2, Corollary 28.10]. The behavior of the sequence generated by the projected gradient method is shown in Figure 1.18. If the current approximation is far enough from the boundary of the constraint set, an iteration of the projected gradient method

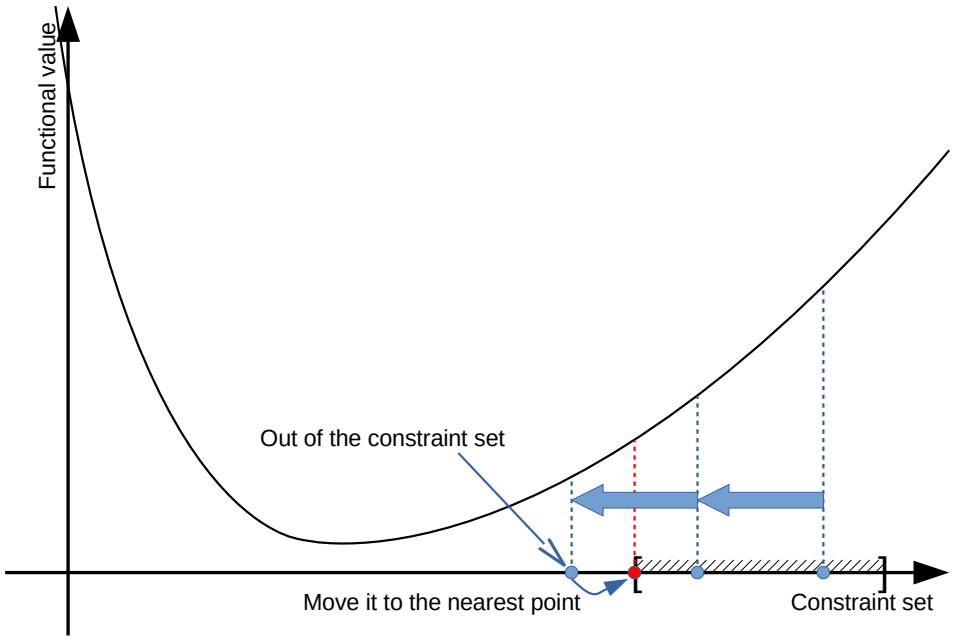


Fig. 1.18: Illustration of the behavior of the projected gradient method

coincides with the steepest descent method. Indeed, the metric projection does not move a given point when it is in the constraint set. When the point generated by an iteration of the steepest descent method is out of the constraint set, the metric projection moves it to the nearest point in the constraint set.

The sequence generated by the projected gradient method converges to the minimizer when the objective functional is smooth enough [2, Corollary 28.10]. Furthermore, to minimize a nonsmooth convex functional or a quasiconvex functional, the subgradient method [5, Section 8.2] and quasiconvex subgradient method [53] with the metric projection can be constructed in a similar way.

Sometimes (as shown in Figure 1.18), the projected point coincides with or is close to the minimizer of the optimization problem. However, this fortunate case is uncommon. Here, let us consider the contour graph (Figure 1.19) of a constrained optimization problem whose objective functional is linear; that is, the gradients at any point are the same, and the constraint set is closed, convex, and bounded. Even for this simple problem, the projected gradient method may suffer from slow convergence. One of the reasons is that the zig-zagging phenomenon occurs near the boundary of the constraint set. To avoid this phenomenon, one can select a specific descent direction considering the effect of the metric projection and the constraint set [35]. However, this requires additional information about the problem setting and an assumption that the constraint set is simple. Hence, there have been many studies on how to efficiently solve constrained optimization problems, and they illustrate the difficulty of constrained optimization.

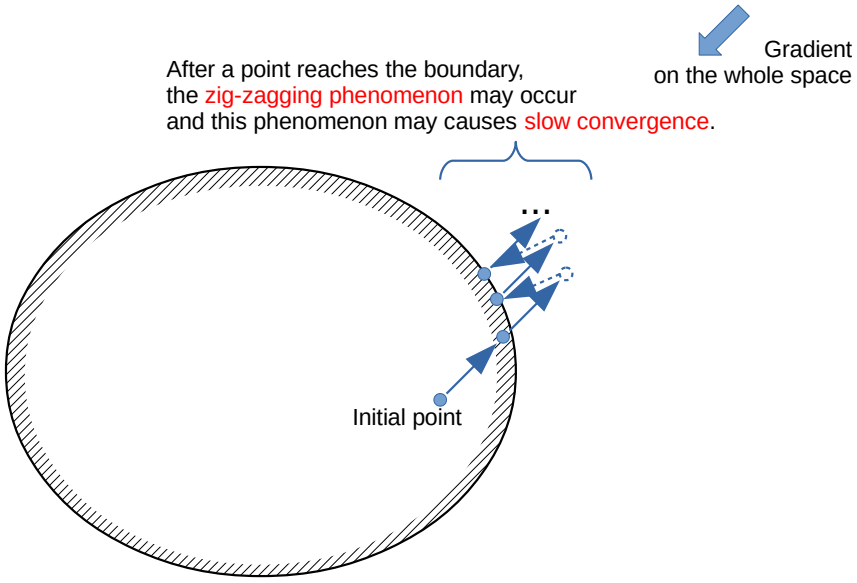


Fig. 1.19: Zig-zagging phenomenon occurring on the boundary of the constraint set

Nonexpansive mappings and their fixed points. Here, let us consider a case where we would like to let the optimal solution satisfy two or more constraints. In the previous example of managing a factory, we may need a plan which considers multiple items and production projects. In such a case, we can't solve the problem by imposing only one simple constraint; we must solve it under multiple constraints.

Let us use the metric projection onto the intersection of the constraint sets, in other words, onto the set whose points satisfy all the constraints. As we saw in Figure 1.17b (metric projection onto a box), the composite mapping of the metric projections onto the constraint sets may coincide with the metric projection onto their intersection in some special cases. However, as Figure 1.20 shows, it does not do so in general. This

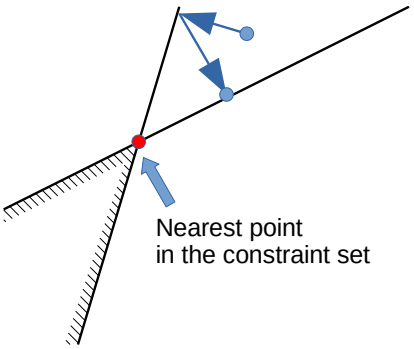


Fig. 1.20: Composite metric projection that does not project a given point onto the intersection

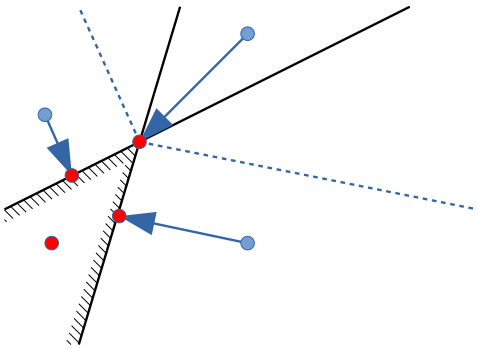


Fig. 1.21: Computing the metric projection by dividing up its domain

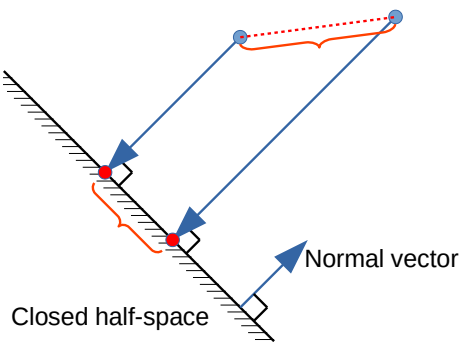
figure illustrates the behavior of a composite mapping of the metric projections onto two closed half-spaces; the red point is the nearest point to the given (upper blue) point in the intersection of the two closed half-spaces. Even in this simple case, the composite mapping cannot project the given point onto the constraint set. Although the metric projection onto two closed half-spaces can be computed by dividing the whole space into four parts (Figure 1.21) [2, Proposition 29.23], this might be difficult in more complicated cases.

How should we express the intersection of two or more constraint sets when solving the constrained optimization problem? As an answer to this question, we can use a notion called a *fixed point*. In the previous discussion, we considered the metric projection from the viewpoint of the constraint set. Here, let us consider it from the opposite perspective, i.e., considering the constraint set from the viewpoint of the metric projection. If the given point is in the set, the metric projection onto it does not move that point since the nearest point is itself. Similarly, the composite mapping of the metric projections onto two or more sets does not move any point in the intersection of these sets, since each metric projection does not move such a point. In general, this property holds as a necessary and sufficient condition; that is, any point in the intersection of nonempty, closed, convex sets does not move even if it is mapped by a composite mapping of the metric projections onto these sets, and any point outside of the intersection moves under the composite mapping [2, Proposition 4.49]. A point that does not move when a mapping is applied is called a fixed point. Hence, we can alternatively call the set onto which the metric projection projects the *fixed point set* [87, Chapter 6].

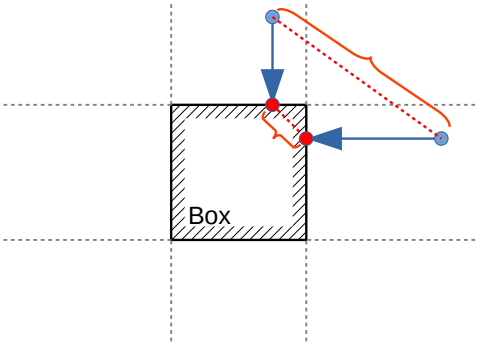
Unfortunately, the composition mapping is not a metric projection since it does not move a given point to the nearest fixed point. Hence, we need to extend the notion of the metric projection in order to discuss the properties of mappings such as composition mappings. Nonexpansivity is an extension of the notion of the metric projection, and composition mappings satisfy it. It is defined as follows: a mapping is called *nonexpansive* if it shrinks or does not change the distance between two arbitrary points [2, Definition 4.1.(ii)]. Any metric projection onto a nonempty, closed, convex set is a nonexpansive mapping [2, Proposition 4.16]. An illustration is given in Figure 1.22. The metric projections onto a closed half-space, a box, and a closed ball shrink the distance between two points outside of the fixed point set. Since these metric projections do not move any fixed point, they do not change distance between two fixed points. This proves that a metric projection is a nonexpansive mapping.

A mapping that rotates a given point around the origin is a typical example of a nonexpansive mapping, and it is not a metric projection. Figure 1.23 shows the behavior of this mapping. The rotation mapping does not change the relative positions of any two points. This implies that it does not change the distance between any two points, and thus, it is a nonexpansive mapping. The fixed point set of this mapping consists of only the center of the rotation, i.e., the origin.

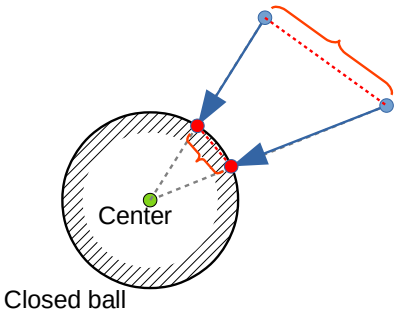
Now let us consider the *generalized convex feasible set* as an example of a set that can be expressed as the fixed point set of a nonexpansive mapping [45, Definition (10)], [93, Subsection 4.B]. Here, we will consider several closed convex constraint sets and



(a) Closed half-space



(b) Box



(c) Closed ball

Fig. 1.22: Nonexpansivities of the metric projections in Figure 1.17

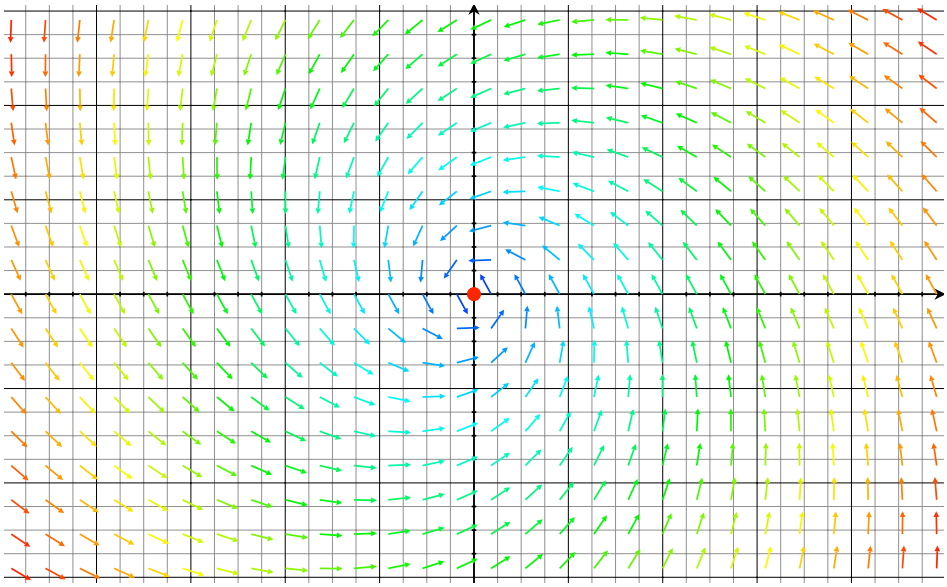


Fig. 1.23: Example of a nonexpansive mapping: rotation

suppose that the metric projections onto them can be easily calculated. In general, the intersection of arbitrarily chosen closed convex sets may be empty. Hence, we would like to use the metric projections to express the set of points whose (squared) distances to each constraint set are at a minimum (as illustrated in Figure 1.24); this set is called the generalized convex feasible set. We can construct a nonexpansive mapping whose

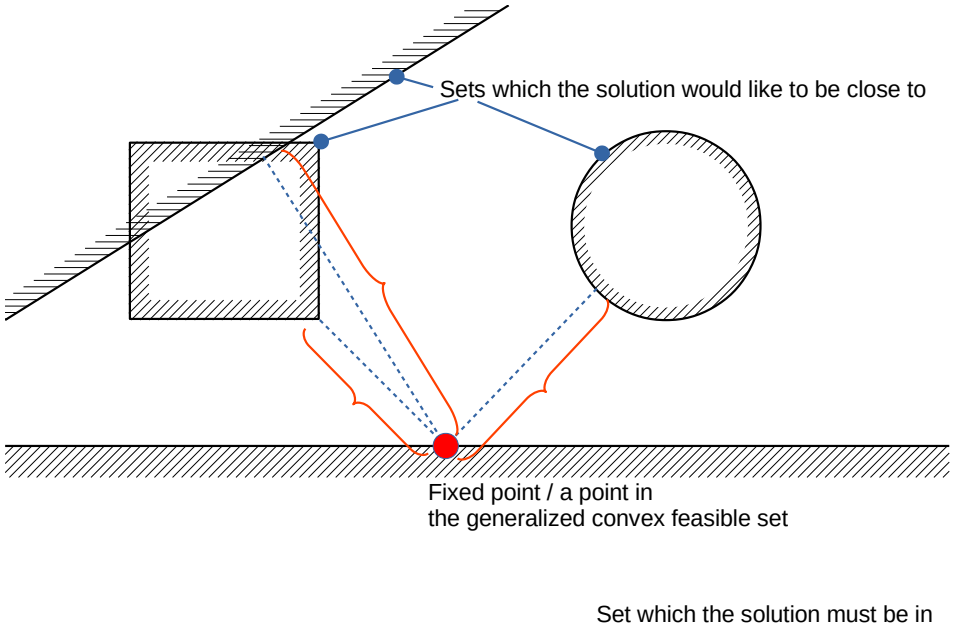
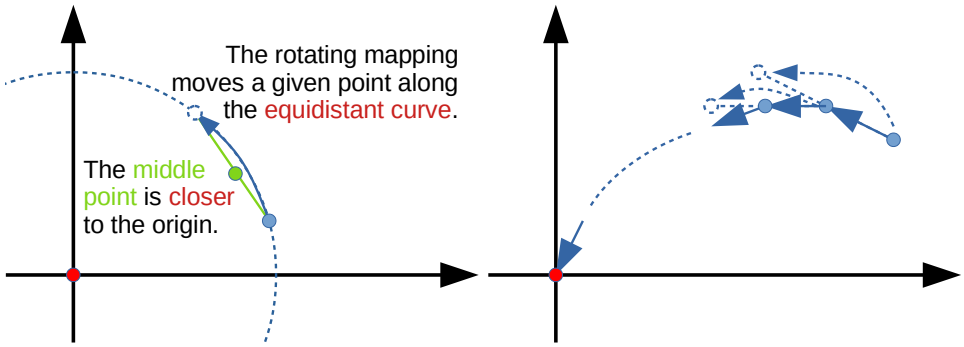


Fig. 1.24: Example of a nonexpansive mapping: generalized convex feasible set

fixed point set expresses this set in the way described in [45, Definition (9)], [93, Definition (50)], and it can be used instead of the intersection of the given constraint sets even when the intersection may be empty.

Fixed point algorithms. We viewed nonexpansive mappings and their fixed point set as extensions of the metric projections and the constraint sets expressed by them. Here, let us review algorithms to find a fixed point of a given nonexpansive mapping and the methods for solving constrained optimization problems whose constraint set is a fixed point set of nonexpansive mappings.

The *Krasnosel'skiĭ-Mann algorithm* [55, 63] is useful for finding fixed points of a nonexpansive mapping. Let us examine the behavior of this algorithm (Figure 1.25) when it is applied to the rotation mapping in Figure 1.23. The mapping rotates the given point around the origin. Hence, the distance between that point and the origin does not change after it is moved by the mapping. However, this implies that the given point and moved one are on an equidistant curve from the origin. Since the inner area of the equidistant curve, that is, the ball whose center is the origin, is strictly convex [2, Corollary 2.16], the midpoint between the given point and the moved one approaches the origin when the given point is not a fixed point of the mapping. On the basis of this idea, the Krasnosel'skiĭ-Mann algorithm [55, 63] generates a sequence



(a) Idea behind the Krasnosel'skiĭ-Mann algorithm

(b) The generated sequence weakly converges to the origin

Fig. 1.25: Behavior of the sequence generated by the Krasnosel'skiĭ-Mann algorithm

approaching a fixed point. The algorithm always generates a sequence that converges weakly to some fixed point of a given nonexpansive mapping on a real Hilbert space [87, Theorem 6.2.3].

Let us consider a way to solve constrained optimization problems whose constraint set is expressed as a fixed point set of some nonexpansive mapping. When the objective functional is differentiable, the *hybrid steepest descent method* [94] can be used to solve the problem. This method is constructed by incorporating the nonexpansive mapping into the steepest descent method. When the gradient of the objective functional is convex and smooth enough, the generated sequence converges to a solution of the problem under certain conditions [94, Theorem 3.2].

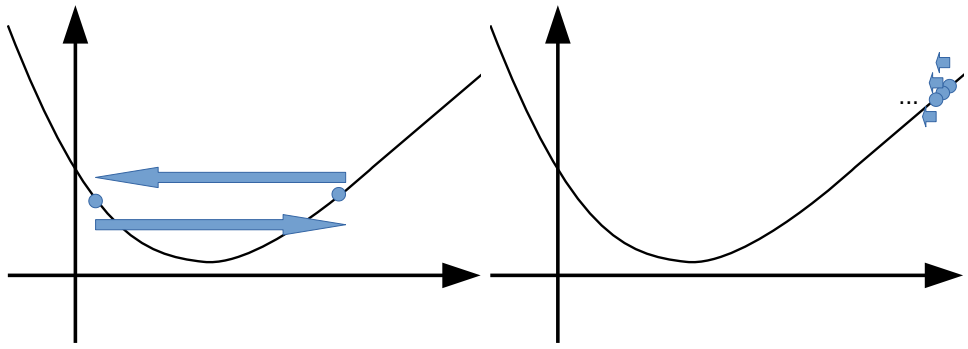
Even when the objective functional is not differentiable, we can construct an algorithm for solving the problem by combining the subgradient method and the Krasnosel'skiĭ-Mann algorithm. By replacing the metric projection appearing in the subgradient method with the computation of the Krasnosel'skiĭ-Mann iterator, we can obtain an algorithm that generates a sequence converging to a solution of a nondifferentiable convex optimization problem with a fixed point constraint [41, Theorem 3.2]. In the next subsection, we describe applications that can be framed as optimization problems whose constraint set is expressed as the fixed point set of a nonexpansive mapping and develop further means of solving them.

1.2.3 Details and variants of the subgradient method

Step size rules. We have discussed three kinds of objective functional: differentiable and nondifferentiable convex functionals and quasiconvex functionals (typical nonconvex functionals). We also described three kinds of constraint: unconstrained (optimization on the whole space), those for which the metric projection can be computed easily, and those that can be expressed as fixed points of some nonexpansive mapping. For each kind of objective functional, there exists a method for minimizing

it, i.e., the steepest descent method, subgradient method, or quasiconvex subgradient method. Similarly, we have tools for dealing with the constraints, i.e., the metric projection onto the constraint set and the Krasnosel'skiĭ-Mann algorithm for finding a fixed point of given nonexpansive mapping. In this subsection, we will investigate these ideas further with the aim of solving more complicated problems.

First, let us consider how to decide on the step size, which is how far the the current approximation moves along the descent direction in an iteration of the (sub-)gradient method. As we saw in Subsection 1.2.1, too large a step causes oscillation of the generated sequence. This phenomenon may, at worst, cause the generated sequence to repeat the same points eternally (it fails to converge; Figure 1.26a). On the contrary,



(a) Too large a step size causes oscillation of the generated sequence, which does not converge to a solution

(b) Too small a step size suffers from slow convergence

Fig. 1.26: Examples of unsuitable step sizes

too small a step size suffers from slow convergence, wherein each iteration does not improve the current approximation enough (Figure 1.26b). Thus, we should always give suitable step sizes to the optimization methods for them to converge quickly.

Constant and diminishing step size rules are often used [5, Section 8.2] [24, 33, 35, 41, 53, 67]. A constant step size rule gives an appropriately small positive constant step; it is useful for acquiring close approximations to the solution [5, Proposition 8.2.2], [41, Theorem 3.1], [33, Theorem 3.1]. Its implementation is easy, since the only thing that needs to be done is to give a constant value to the method. Furthermore, the situation in Figure 1.26b can be avoided since the step size does not decrease. In contrast, the diminishing step size rule gives a sequence of decreasing real numbers; it can guarantee convergence of the generated sequence to the optimal solution [5, Proposition 8.2.6], [41, Theorem 3.2], [33, Theorem 3.2]. After the enough iterations, the step size becomes small enough to determine that the solution has converged. Hence, by adopting a diminishing step size rule, we can avoid the situation in Figure 1.26a.

Line search. As discussed above, choosing a suitable step size is important for efficient convergence. Here, we will describe the technique used in unconstrained opti-

mization for obtaining a suitable step size at run-time, which is called the *line search*. Let us again consider the contour graph and the behavior of the steepest descent method on it (Figure 1.27). The steepest descent method improves the current ap-

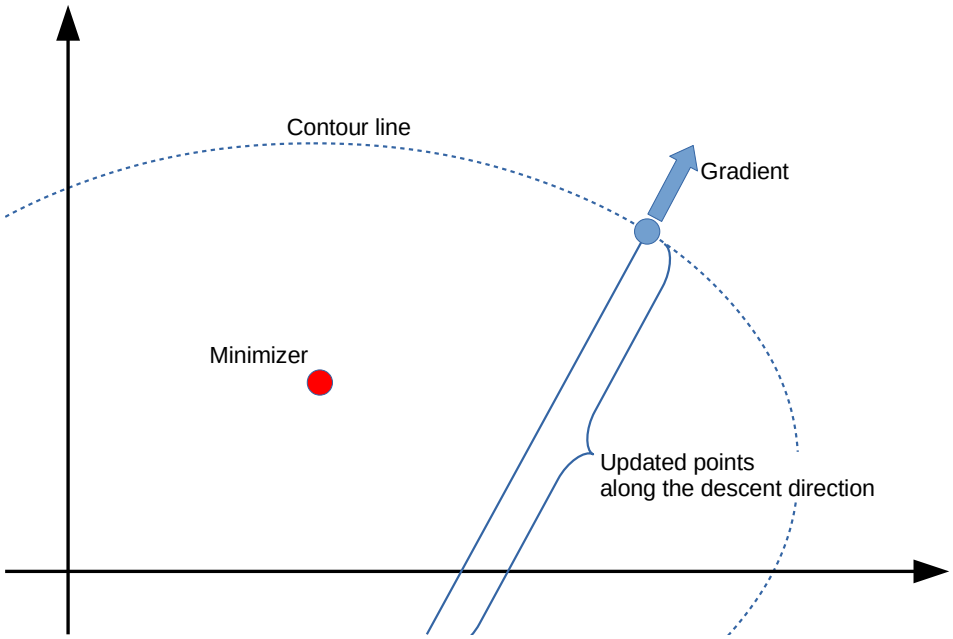


Fig. 1.27: Updated points along the descent direction

proximation by moving it in the descent direction, i.e., the opposite direction to the gradient. Depending on the choice of step size, the updated point can be far from or close to the current approximation. In other words, there are many candidate update points along the descent direction, and this implies that we need to find an appropriate one, i.e., to choose a suitable step size. Figure 1.28 is a graph of the objective functional on this candidate line; the horizontal axis is the step size, and the vertical axis is the value of the objective functional. One of the simplest criteria for choosing a suitable step size is to find the minimizer of the graph. This is called the *exact line search* [68, Definition (3.1)]. As shown in Figure 1.27, the minimizer of the objective functional may not be on the candidate line. In fact, the exact line search may require too many computations, and it may slow convergence even if it can be done. Hence, more practical strategies, called inexact line searches, are used to find a suitable step size at a minimal cost [68, Section 3.1].

The *Armijo condition* is that the step size is acceptable only if its functional value is below a linear functional for the criterion with a negative slope. The linear functional is plotted as the red line in Figure 1.28, and the acceptable step sizes are in the red hatched area. This condition ensures that each iteration updates the current approximation to a better one.

However, although the algorithm will make progress for any sufficiently small step size, as we saw in Figure 1.26b, too small a step size suffers from slow convergence. To deal with this issue, we can impose a *curvature condition* that accepts only large

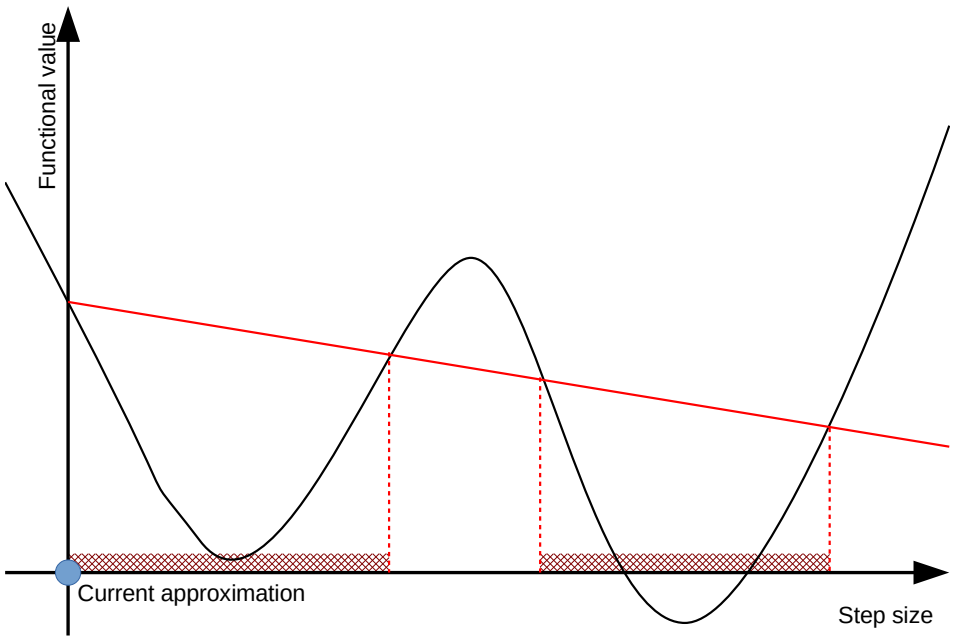


Fig. 1.28: Acceptable step sizes for the Armijo condition [68, Figure 3.3]

enough step sizes by taking the derivatives of the objective functional. The combination of the curvature condition and the Armijo condition is called the *Wolfe condition* [68, Chapter 3], [92].

A line search is often used in unconstrained optimization [22, 95]. However, it cannot be used directly in constrained optimization. Let us consider the candidate line (illustrated in Figure 1.29) when the projected gradient method is used. When

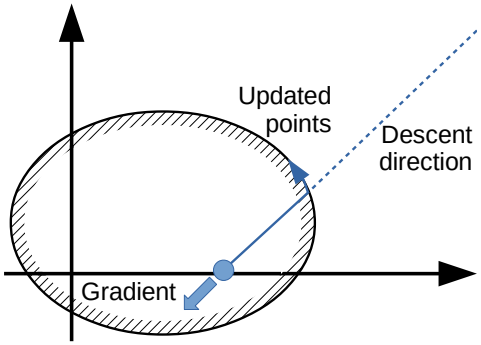


Fig. 1.29: Updated points in the constrained case

the point moving along the descent direction is outside the constraint set, the method projects that point onto the constraint set. This operation distorts the candidate line. Indeed, Figure 1.29 shows that the candidate line curves along the boundary of the constraint set. We have to deal with such distortion when we use a line search to find a suitable step size in constrained optimization.

Distributed optimization. Let us consider a situation where the objective functional is a sum of convex functionals. An instance of such a situation is the *network utility maximization problem* [47, 52], in which the goal is to find the best allocation of network resources. Let us discuss this problem with the help of a concrete network topology modeled as a connected, undirected graph and its directed paths (Figure 1.30). In this graph, the vertices express the nodes of the network, such as routers

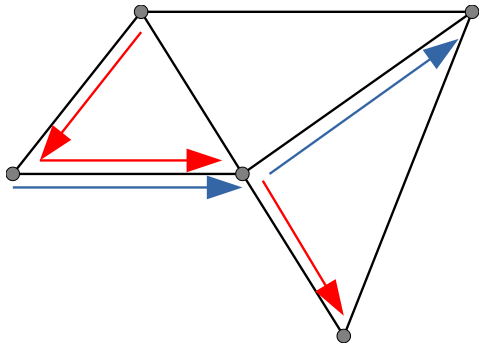
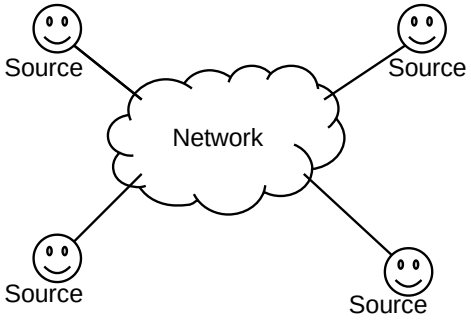


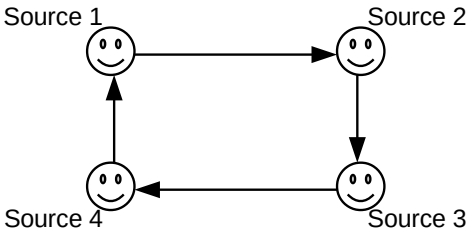
Fig. 1.30: Network topology of network utility maximization problem

and client computers, while the edges express links connecting pairs of nodes. The directed paths on this graph (illustrated as sequences of blue and red arrows) express communications flowing on the network. Here, each communication would be better if its utility value, which is the usable transmission rate for the communication, were higher. Hence, the sender, called the source of each communication, would like to maximize this value. Here, the utility does not become any better if too high a transmission rate is assigned to communication; i.e., there exists a threshold on the transmission rate for each communication. Accordingly, each utility functional may be modeled as a nonsmooth, convex functional [47, Definition 5]. Moreover, each link has a capacity and a negative amount of information cannot flow. Thus, each communication flowing on the one or more links must be within the limits of those links. This forms the constraints of the network utility maximization problem. We will further assume that two or more communications flow on the network. Therefore, the task is to maximize multiple utilities fairly. To ensure fairness, the objective functional of the network utility maximization problem should be modeled as a sum of individual utilities appropriately.

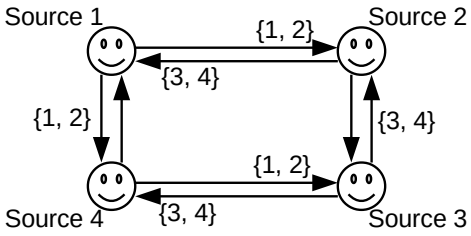
The network utility maximization problem should be solved in a distributed way, because we can not assume that an operator is present to manage the network so the sources must collaborate in some way to allocate the resources in the network to each other. Hence, to solve the constrained convex optimization problem, we would like to use a method that can run in a distributed computing environment. Let us examine this situation from the viewpoint of each source. Each source is connected to a certain kind of network, such as shown in Figure 1.31. The *incremental subgradient method* can be applied when there are enough links to establish a ring network. The behavior of this method is as follows. The previous source sends the current



(a) Networked system



(b) Communication appears in the incremental subgradient method



(c) Communication appears in the parallel subgradient method

Fig. 1.31: Networked systems and incremental and parallel subgradient methods

approximation to its neighbor source. The source that received the approximation improves it by using the subgradient method iterator with a subgradient of its own objective functional, and it sends it to the next source. By repeating this procedure cyclically, the incremental subgradient method minimizes the objective functional.

In the incremental subgradient method, other sources wait to receive the approximation from their previous source in order to improve their own approximations. On the other hand, the *parallel subgradient method* [24], where all sources can improve the current approximation independently and in parallel, can be used in a denser network. In the parallel subgradient method, all sources simultaneously improve the current approximation by using the subgradient of their own objective functional. Then, each source has each approximation improved with each objective functional. We can efficiently share these approximations by using, e.g., the `MPI_Allreduce` procedure [65, Subsection 5.9.6] when the connected network is dense enough. For example, suppose there are four sources: sources 1, 2, 3, and 4. First, sources 1 and 2 exchange their approximations, and at the same time, sources 3 and 4 exchange theirs. At this point, sources 1 and 2 have each other's information, and sources 3 and 4 have each other's as well. Next, sources 1 and 3 and sources 2 and 4 exchange all the information

they have. In this way, all sources can share their information by communicating two times, not four times, per source. This sharing scheme is called a butterfly [71, Section 5.6]. After sharing information, the sources simultaneously improve the shared information; then they share their results again.

Similarly to the subgradient method, the incremental and parallel subgradient methods [24, 67] use the metric projection onto the constraint set so that the generated sequence in in the constraint set. However, it may be difficult to solve the network utilities maximization problem. If the links all have the same capacity and if the communications are distinct, we can compute the metric projection onto the constraint set since it is a simple box. However, this situation is rare, and the network and the communications flowing on it are generally more complicated. In such case, we have to minimize the objective functional on the intersection of many constraints in order for the solution to satisfy all the capacity conditions required by all the links.

Reference [41] provides a variant of the parallel subgradient method for minimizing the sum of convex objective functionals over the intersection of the fixed point sets of nonexpansive mappings by using the idea behind the Krasnosel'skiĭ-Mann algorithm, and reference [43] provides a variant of the incremental subgradient method for minimizing the sum of convex objective functionals over the intersection of the fixed point sets of nonexpansive mappings. These methods do not require the metric projection; hence, they can be used to solve the optimization problem even if the metric projection onto the constraint set cannot be easily computed. As we saw, the fixed point set of nonexpansive mappings can express the intersection of two or more simple constraint sets, and the constraint with respect to each link of the network utility maximization problem is a simple box. Hence, we can use these methods for solving distributed problems such as the network utility maximization problem. Furthermore, reference [84] provides a variant of the parallel subgradient method for minimizing the sum of nonsmooth, convex objective functionals over the intersection of the fixed point set of quasi-nonexpansive mappings. The class of quasi-nonexpansive mappings is wider than the class of nonexpansive mappings. Its fixed point set can express a more complicated situation than the fixed point set of a nonexpansive mapping, e.g., the level set of a given continuous, convex functional [2, Proposition 29.41].

The problem of minimizing the sum of nonsmooth, convex functionals over the intersection of the fixed point set appears in many applications [23, 47, 48, 72]. The classifier ensemble problem is an interesting instance; it arises in the field of machine learning [23, 48]. The classification task is to fit a classifier to given training data. In other words, it can be considered to be the task of minimizing an objective functional constructed from the training data. This implies that the number of objective functionals coincides with the number of training data, and the task as a whole is to minimize the sum of these objective functionals. The main difficulties with this procedure are as follows:

- the size of the training data may be too large to load onto the memory of one computer;
- two or more indicators may have to be considered.

Some datasets in machine learning are too large to load onto the memory of one computer. For example, the size of the click dataset [64], on 53.5 billion HTTP requests made by users at Indiana University, reaches 2.5 TB even if it is compressed. For dealing with such a dataset, we can divide it into parts and distribute the parts to two or more computers. The Meiji University PC Cluster[*1] is one such system for doing this. Although each computer contained in it has 96 GB of memory (still much more than a typical personal computer), the cluster is composed of 24 computers, meaning it can handle 2.3 TB of data in total. Furthermore, each of its computers is independent of the others. Hence, for the whole computation, a process on one computer must collaborate with processes on the other computers. Here, the incremental and parallel subgradient methods and their variants can minimize the sum of distributed objective functionals. Hence, they can solve large problems by running them on a system like the PC cluster.

References [23, 48] tackle the problem of multiple indicators: the objective functionals on individual training data, the sparsity of the classifier, and its diversity. To deal with these indicators, references [23, 48] model the sparsity and diversity as the fixed point sets of certain mappings. Quasi-nonexpansive mappings can express the level set of a functional as their fixed point set. Hence, we can model the conditions under which the classifier will have a certain degree of sparsity and diversity into quasi-nonexpansive mappings [23, 48]. Hence, even if the problem has two or more indicators, we can deal with them by converting them into fixed point sets of certain mappings and using fixed point algorithms.

Section 1.3. Proposals of this thesis and their contributions

This thesis proposes three methods for solving constrained nonsmooth optimization problems. The first two are the *incremental and parallel line search subgradient algorithms*, and the third is the *fixed point quasiconvex subgradient method*. All are designed to solve constrained nonsmooth optimization problems whose constraint sets are expressed as fixed point sets of mappings. We will consider the advantages and disadvantages of each method and examine its results.

1.3.1 Incremental and parallel line search subgradient algorithms

Issues of the existing methods. Let us consider the minimization of the sum of convex objective functionals on some closed, convex constraint set. To solve this problem, we can use the incremental and parallel subgradient methods [24, 67], as we saw in the previous section. The incremental subgradient method sequentially and cyclically uses each part of the objective function, and the parallel subgradient

[*1] Meiji University PC Cluster `https://www.meiji.ac.jp/isys/hpc/pcc.html` (in Japanese).

method uses the parts independently in parallel. However, these methods use a step size, also known as the learning rate in the field of machine learning, to determine how far they should move the current approximation along the descent direction in each iteration, because the subgradient method iterator is used for improving the approximation. As we saw in the previous section, an excessively small or large step size suffers from slow convergence.

Furthermore, the incremental and parallel subgradient methods use the same step size for every composing objective functionals at each iteration [24, Step 1, Algorithm 3.1], [67, Definition (1.5)]. However, each functional may have different properties: the position of its minimizer may be far from the minimizer of the summed objective functional or other composing objective functionals, its shape may be different from the others, and so on. Of course, the suitable step size depends on these factors, and hence, it may be different for each composing objective functional. Let us consider the case where the minimizer of one composing objective functional (composing functional 1 in Figure 1.32) is on one side of the minimizer of the summed objective functional and the minimizer of another objective functional (composing functional 2 in Figure 1.32) is on the opposite side. As we said, the existing in-

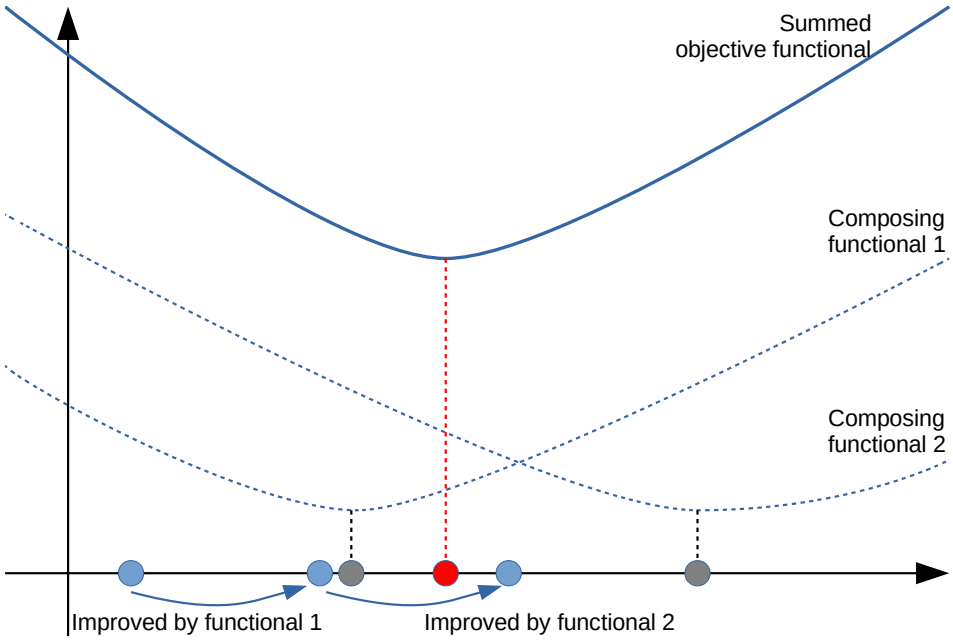


Fig. 1.32: One of the directions to the minimizers of the objective functionals may not lead the approximation to the minimizer of the summed objective functional.

cremental subgradient method uses the same step size for each composing objective functional. This implies that the ratios of the distance moved by the improvement to the norms of the subgradients are the same on each composing objective functional. For simplicity, let us assume that the norms of the chosen subgradients are the same. Then, the distances moved by making the improvements become the same. The incremental subgradient method improves the approximation by using each composing

functional in order. First, it uses the information of composing functional 1; the approximation approaches the minimizer of the summed objective functional since the minimizer of composing functional 1 is on the side the approximation is on. Next, the method uses the information of composing functional 2, but the approximation passes by the minimizer and moves to the opposite side of it since the minimizer of composing functional 2 is on that side. Since the descent direction by composing functional 2 is the opposite one to the minimizer of the summed objective functional, a kind of oscillation phenomenon appears in the generated sequence. This implies that an unsuitable step size like this may slow convergence.

Methods for overcoming the issues. A line search can be used to find an appropriate step size for each iteration. However, it cannot be used directly in constrained optimization since, as we saw in the previous section in Figure 1.29, the operations to enable the solution to fulfill the constraints distort the candidate line for choosing the step size. To overcome this issue, we propose new variants of the incremental and parallel subgradient methods that use the idea of the line search to find a suitable step size in each iteration.

Let us consider the step size used in the existing incremental and parallel subgradient method. To ensure convergence to the optimal solution, the existing methods use a diminishing step size (shown as the red line in Figure 1.33). However, the step sizes

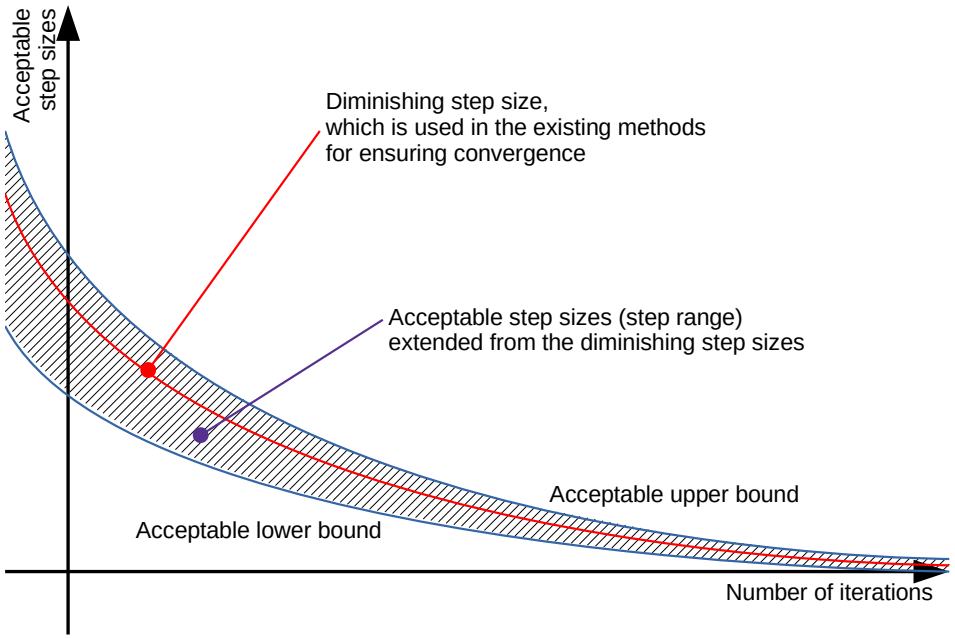


Fig. 1.33: Extending the notion of step sizes into step range

has to be decided before the method runs. For this reason, we must use the given step size even if it is not appropriate for the current approximation or the suitable step sizes for different composing functions are different. To overcome this issue, we propose the incremental and parallel line search subgradient algorithms, which ex-

tend the notion of the diminishing step size to a candidate line called the step range. This extension enables us to reduce the analysis involved in the step size selection to the diminishing step size rule. Our methods require only the step range to be given i.e., the lower and upper bounds of the step size and whose lower and upper bounds have similar properties of the diminishing step size. The methods choose a suitable step size from this step range at run-time. For any step size chosen from the step range, its lower and upper bounds satisfy the properties of the diminishing step size rule. Hence, we can perform a convergence analysis in the same way as when using a diminishing step size rule. This extension enables us to use a line search in the step range. At each iteration, and for each objective functional, the method chooses a suitable step size from the step range by using the line search, and this overcoming the above issues.

Advantages and disadvantages. The proposed methods have three merits that the existing ones don't have. First, the suitable step sizes chosen by the line search accelerate the algorithms and make their solutions better. In the existing methods, we must use the given step size even if it is not appropriate for the current approximation or the suitable step sizes for different composing functions are different. The proposed algorithms use suitable step sizes chosen by the line search and different step sizes for different composing functionals. Hence, they can always use appropriate step sizes, and this speeds convergence. The second merit is that we do not need to adjust the step size precisely. The existing methods use the given step size. Hence, for efficient convergence, we have to adjust the step size carefully before the method runs. In contrast, the proposed methods only need a step range, i.e. rough candidates, to converge efficiently, because the line search automatically chooses the learning rates from among this range. Finally, the proposed algorithms can be applied to difficult problems whose suitable step sizes cannot be chosen beforehand. Even when a suitable step size cannot be specified beforehand, the line search can algorithmically find one at run-time and make the algorithms converge efficiently to an optimal solution. In addition, if the step range is a singleton set, our methods coincide with the existing incremental and parallel subgradient algorithms [24, 67]. Hence, the step range is a generalization of the step sizes used in the existing algorithms.

Now let us consider the disadvantages of the proposed methods. They use the line search for choosing the step size from the step range at run-time. This implies that an additional computation is required at each iteration. Hence, the computation of each iteration is heavier than the existing ones. However, the parallel subgradient method can run in parallel for each composing objective functional when the computing environment is composed of two or more CPUs. This implies that making good use of parallelization alleviates the drawback of the line search procedure.

As an instance of problems appearing in a distributed computing environment, let us consider the joint task offloading scheduling and resource allocation problem in a multi-server mobile-edge computing network [88, Problem 11]. A multi-server mobile-edge computing network is composed of several base stations and mobile users connected to them (Figure 1.34). Base stations are connected to the internet directly

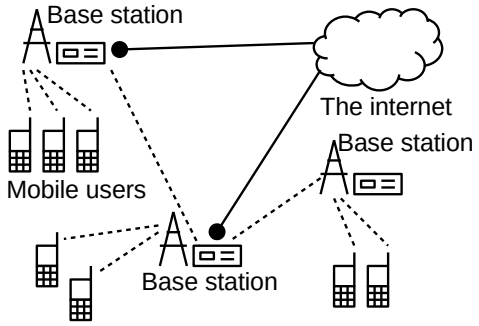


Fig. 1.34: Multi-server mobile-edge computing network

or via other stations, and each base station implements a computer called a Mobile-Edge Computing (MEC) server. Consider a situation where we provide a service to mobile users on the internet. If we could use the MEC servers implemented in the base stations as well as our own server on the internet, we could reduce the delay caused by the distance between our server and the mobile user and this would release our server from some of its burden [88, Section 1]. This idea can modeled as an optimization problem of minimizing the sum of all mobile users' utilities. In this situation, we can use the MEC servers, i.e., parallelization by these servers, and all of them are connected to each other by the network. Furthermore, there is a cost for two MEC servers communicating since they are separated. Hence, if the line search improves the effect of one iteration, parallelization could be expected to reduce the time needed for optimization. Moreover, the proposed methods can be used on other nonsmooth, constrained, convex optimization problems appearing in various applications [10, 39, 60, 62, 78, 83, 85]. This implies the usability of the proposed methods and the possibility of contributing to solving these problems.

1.3.2 Fixed point quasiconvex subgradient method

Issues of the existing methods. Optimization problems nowadays are not limited to only convex objectives. In particular, quasiconvex objective functionals appear in economics, engineering, and management science [33, 35]. Furthermore, there are various situations in which one optimizes ratio indicators, such as the debt/equity ratio in financial and corporate planning, inventory/sales and output/employee ratios in production planning, and cost/patient and nurse/patient ratios in healthcare and hospital planning [86]. Under certain conditions, these ratio indicators, fractional objective functionals in other words, have quasiconvexity [53, Lemma 3]. Here, let us consider minimizing a continuous, quasiconvex objective functional over a nonempty, closed, convex constraint set.

The quasiconvex subgradient method proposed by Kiwiel [53] is useful for solving this minimization problem. This method uses a normalized normal vector to the slice as a subgradient. A number of variants exist, such as the inexact subgradient method

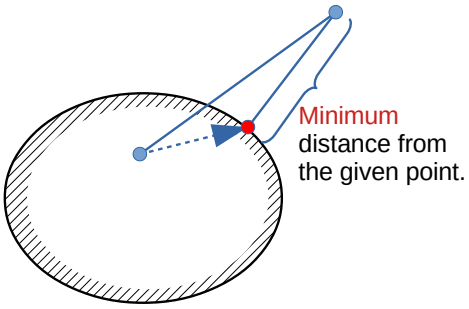


Fig. 1.35: Finding the metric projection is equivalent to minimizing the distance from the given point.

[33], the conditional subgradient method [35], and the stochastic subgradient method [34].

These methods must assume that the metric projection onto the constraint set is computable, because they use it to guarantee the solution is in the constraint set. The metric projection in this case is defined as a mapping which sends a given point to the nearest point inside the constraint set. This implies that we have to solve a subproblem of minimizing the distance from a given point subject to the solution being in the constraint set, as illustrated in Figure 1.35. This is an instance of constrained convex optimization problems. Certainly, there are some sets onto which the metric projections can be computed easily, such as boxes [2, Proposition 29.15], closed balls [2, Example 3.18 and Proposition 3.19], [82, Section 4], and closed half-spaces [2, Example 29.20]. However, practical problems often have various complicated sets on which computing metric projections is difficult[11, 39, 45, 46, 93]. Hence, we have to solve a constrained convex optimization problem at each iteration in such cases. This additional procedure increases the cost of computing one iteration.

Method for overcoming the issue. We propose the fixed point quasiconvex subgradient method for solving a constrained, nonsmooth, quasiconvex optimization problem lightly and quickly even when it is difficult to compute the metric projection onto the constraint set. If the constraint set can be expressed as a fixed point set of or the intersection of some fixed point sets of nonexpansive mappings, there are methods that use these nonexpansive mappings instead of the metric projection for convex optimization [41, 42, 43, 45]. These methods use nonexpansive mappings instead of metric projections onto the constraint sets. This implies that these methods can be expected to run more efficiently than the methods which use metric projection directly when the nonexpansive mappings can be more easily computed than the metric projections, since the additional procedure for solving the subproblem to find the metric projection onto the constraint set is not necessary. Inspired by these methods of convex optimization, we developed the fixed point quasiconvex subgradient method, a variant of the quasiconvex subgradient method [53], which uses the Krasnosel'skiĭ-Mann iterator [55, 63] to ensure the generated sequence converges to

a fixed point of the nonexpansive mapping. For minimizing a quasiconvex objective functional, we use the idea behind the quasiconvex subgradient method [53], which uses a normalized normal vector to the slice at the current approximation instead of the subgradient. To ensure that the generated sequence converges to a fixed point of the nonexpansive mapping, we use Krasnosel'skiĭ-Mann iterator [55, 63].

Advantages and disadvantages. The proposed methods have three merits that the existing ones don't have. First, it is a widely applicable to constrained quasiconvex optimization problems. It can minimize the objective functional even when it is a nonconvex functional, which the existing subgradient method [41, 42, 43, 45] for solving convex optimization problems cannot deal with. Hence, the proposed method has wider applicability than the existing methods with respect to the class of the objective functionals. As well, the proposed method can solve constrained quasiconvex optimization problems that have constraint sets the existing methods [33, 34, 35, 53] cannot deal with. Of course, the metric projection used in the existing methods [33, 34, 35, 53] is also nonexpansive, since the nonexpansive mappings are an extended notion of the metric projection. Hence, the proposed method can solve all of the problems that the existing methods [33, 34, 35, 41, 42, 43, 45, 53] can solve. Second, the proposed method can solve problems whose metric projections onto constraint sets cannot be easily computed. Like the existing method [41, 42, 43, 45] for convex optimization over fixed point sets of nonexpansive mappings, it uses a nonexpansive mapping instead of the metric projection and the Krasnosel'skiĭ-Mann iterator letting the approximation approach the fixed point of the projection. Accordingly, it does not need to solve the subproblem for finding the metric projection onto the constraint set at each iteration. This reduces the size of the computation in each iteration, thereby speeds convergence. Finally, the proposed method for quasiconvex optimization will be of help in developing the field of convex optimization and expand its applications. Convex optimization problems are actively studied [41, 42, 43, 45] and have various applications, such as signal recovery [12], machine learning [17, 23], and network resource allocation [39, 49]. Because any convex functional is also a quasiconvex functional, we can deal with these applications within the framework of quasiconvex optimization. This implies that it would contribute to solving constrained convex optimization problems.

Let us consider the disadvantages of the proposed methods. The Krasnosel'skiĭ-Mann iterator moves the given point toward the fixed point set, but it does not ensure that the mapped point is in the fixed point set. Hence, unlike in the existing methods using the metric projection [33, 34, 35, 53], the obtained solution will be near the constraint set but might not be in it. However, the convergence rate analysis of the distance to the fixed point set allows us to estimate how many iterations will be required for obtaining a solution near enough to the constraint set.

Our GitHub repository `https://github.com/iiduka-researches/201811-kaz` contains an implementation of the fixed point subgradient method and miscellaneous utilities for constructing nonexpansive mappings which express the desired constraint sets. By using these implementations, the reader can easily make a nonexpansive

mapping expressing his or her desired constraint set and optimize a constrained quasiconvex optimization problem. As a numerical example, we apply this implementation of the fixed point quasiconvex subgradient method to solve a concrete constrained quasiconvex optimization problem called the Cobb-Douglas production efficiency problem after this section. The goal of this problem is to find the most efficient production factors under funding level restrictions [33, Section 6]. The objective function represents the ratio between the total profit and the total cost as an efficiency indicator. The total profit is the numerator of the objective function and is modeled with the Cobb-Douglas production function on the production factors. The total cost is the denominator of the objective function and is modeled with the affine function on the production factors. Furthermore, there are a variety of constraints on the funding level [33, Section 6]. These constraints represent the duties and restrictions of each production project. All of these constraints must be satisfied; hence, the constrained set is composed as the intersection of many constraint sets, which can be expressed as the fixed point set of a nonexpansive mapping. Of course, all the constraint sets appearing in the Cobb-Douglas production efficiency problem are provided in our GitHub repository.

Section 1.4. Content after this chapter

This thesis is organized as follows. Chapter 2 presents the incremental and parallel line search subgradient algorithms and their convergence analyses. It also describes numerical experiments in which the proposed methods are applied to machine learning. Chapter 3 presents the fixed point quasiconvex subgradient method and discusses its convergence. Chapter 4 provides a convergence rate analysis of the fixed point quasiconvex subgradient method.

Chapter 2

Incremental and Parallel Line Search Subgradient Algorithms

There are many instances of optimization problems whose objective functional can be expressed as a sum of convex functionals, such as in learning with a support vector machine. Incremental and parallel subgradient algorithms are useful algorithms for solving them. The incremental subgradient algorithm sequentially and cyclically uses each of the parts of the objective function, and the parallel subgradient algorithm uses parts independently in parallel. However, they suffer from slow convergence because their learning rates must be determined before running them. This chapter proposes the modified algorithms of them to incorporate the line search algorithm, which automatically and algorithmically finds appropriate learning rates at run-time. These algorithms can be applied to constrained nonsmooth convex optimization problems appearing in tasks of learning support vector machines without adjusting the learning rates precisely. The proposed line search algorithm can determine learning rates to satisfy weaker conditions than the ones used in the existing machine learning algorithms. This implies that the two algorithms are generalizations of the existing incremental and parallel subgradient algorithms for solving constrained nonsmooth convex optimization problems. We show that they generate sequences that converge to a solution of the constrained nonsmooth convex optimization problem under certain conditions. The main contribution of this chapter is the provision of three kinds of experiment showing that the two algorithms can solve concrete experimental problems faster than the existing algorithms. First, we show that the proposed algorithms have performance advantages over the existing ones in solving a test problem. Second, we compare the proposed algorithms with a different algorithm Pegasos, which is designed to learn with a support vector machine efficiently, in terms of prediction accuracy, value of the objective function, and computational time. Finally, we use one of our algorithms to train a multilayer neural network and discuss its applicability to deep learning.

The contents of this chapter are based on

[26] K. Hishinuma and H. Iiduka. Flexible stepsize selection of subgradient meth-

ods for constrained convex optimization. the 10th Anniversary Conference on Nonlinear Analysis and Convex Analysis (Oral), 2017;

[28] K. Hishinuma and H. Iiduka. Convergence analysis of incremental and parallel line search subgradient methods in Hilbert space. *Journal of Nonlinear and Convex Analysis*, 20(9):1937–1947, 2019;

[30] K. Hishinuma and H. Iiduka. Incremental and parallel machine learning algorithms with automated learning rate adjustments. *Frontiers in Robotics and AI*, 6:77, 2019.

Section 2.1. Introduction

In this chapter, we consider a technique to adjust the learning rates that appear in subgradient algorithms for letting a generated sequence converge to an optimal solution. The subgradient algorithm [5, Section 8.2] and its variants [24, 67, 83] have been proposed as ways of solving the problem of minimizing a nonsmooth, convex function over a closed convex set by iterative processes like the steepest descent algorithm for dealing with a smooth, convex function. These algorithms iterate the current approximate solution by shifting it along a descent direction at that point by a given degree called a *learning rate*. Although descent directions are decided on the basis of the subgradient at each point, the learning rates are generally decided for theoretical reasons for ensuring the convergence of the generated sequence. This implies that subgradient algorithms can be run more efficiently if we can choose more suitable learning rates concerning the objective function at each iteration. Therefore, we should consider how to choose better learning rates while at the same time maintaining the convergence properties.

We can reduce a lot of practical problems to ones solvable with subgradient algorithms, that is, problems of minimizing a nonsmooth, convex function over a closed convex set. One of the important applications is learning with a support vector machine. Support vector machines are effective and popular classification learning tools [60, 62, 78, 83]. The task of learning with a support vector machine is cast as an empirical loss minimization with a penalty term for the norm of the classifier that is being learned [83, Problem (1)]. If this loss objective function is convex, we can handle this learning task by minimizing a nonsmooth, convex function over a closed convex set. There are practical optimization algorithms for solving this minimization problem, such as Pegasos [83], the incremental subgradient algorithm [67], and the parallel subgradient algorithm [24]. These algorithms are variants of the subgradient algorithm. They iteratively choose training examples and improve their approximation by using a part of the objective function which corresponds to the chosen examples. Not limited to machine learning, there exist many applications of minimizing a nonsmooth, convex function over a closed convex set, such as signal recovery [10], bandwidth allocation [39], and beamforming [85]. Hence, making the performance of these algorithms better would increase the efficiency of these applications. Here, we attempt to do so by modifying the selection of the learning rate.

Pegasos is a stochastic subgradient algorithm with a carefully chosen learning rate that is designed for efficiently learning with a support vector machine [83]. This learning rate is determined from the regularization constant of the penalty term. Hence, this algorithm can improve approximate solutions without having to adjust their learning rates for each individual learning task. However, it is specialized to learning with a support vector machine, and it cannot be applied to other applications such as deep learning.

The sequential minimal optimization (SMO) algorithm [76] is also used for learning with a support vector machine. This algorithm can be applied to a quadratic programming optimization problem appearing in learning with a dual form of a support vector machine and can solve it with a small amount of memory and quickly [13, 76]. However, this algorithm deals with the dual form of the optimization problem; as such, the number of objective variables is likely to be large when many instances are given to the learning task. Furthermore, the class of problem that this algorithm can deal with is limited to quadratic programming. This implies that it cannot be applied to general nonsmooth, convex programming.

In the field of mathematical optimization, the incremental and parallel subgradient algorithms [24, 67] are useful for solving problems involving the minimization of a nonsmooth, convex function over a closed convex set. The incremental subgradient algorithm [67] minimizes the objective function by using alternately one of the functions composing the summed objective, while the parallel subgradient algorithm [24] minimizes it by using all of the composing functions independently. Since the parallel subgradient algorithm treats each of the composing functions independently, computations with respect to each function can be parallelized. It is expected that parallelization shortens the computational time of learning. This implies that the parallel subgradient algorithm can learn support vector machines for larger datasets and/or in a shorter time compared with other algorithms.

A weak point of the incremental and parallel subgradient algorithms [24, 67] is that they need to have suitably adjusted learning rates in order to run efficiently. However, the suitable learning rate depends on various factors, such as the number of the composing objective functions, number of dimensions, the shape of each objective function and constraint set, and the selection of subgradients. This implies that it is too difficult to choose a suitable learning rate before run-time. In contrast, Pegasos [83] uses a concrete learning rate optimized for the task of learning with a support vector machine and does not require this learning rate to be adjusted. Therefore, it can be used more easily than the incremental and parallel subgradient algorithms [24, 67].

In unconstrained minimization algorithms, line searches are used to select a suitable learning rate [22, 95]. In particular, the *Wolfe conditions* [92] are learning rate criteria for the line search. The Wolfe conditions are such that the learning rate must satisfy a sufficient decrease condition and a curvature condition [68, Chapter 3]. The sufficient decrease condition is that the learning rate is acceptable only if its function value is below a linear function with a negative slope. This condition ensures that the algorithms update an approximation to a better one. However, it is not enough to

ensure that the algorithm makes reasonable progress because it will do so for all sufficiently small learning rates. Therefore, a curvature condition is invoked that generates a sequence further enough along the chosen direction.

Motivated by the idea of the line search, this chapter proposes novel incremental and parallel subgradient algorithms that can run efficiently without precise learning rate adjustments. Reference [14] describes a gradient-projection algorithm with a line search that minimizes the objective function. However, this algorithm assumes that the objective function is differentiable. In addition, it is designed for single-core computing; it is not useful in multi-core computing. Reference [3] proposes the radar subgradient algorithm, which is a variant of the subgradient algorithm including a procedure for finding an effective learning rate by using a line search at each iteration. The line search algorithm used in Reference [3] is inspired by the cutting-plane method and works out a learning rate with the first-order information. However, this algorithm deals with the whole objective function and cannot use a part of the objective function at each iteration. This implies that it cannot be used in applications that give information to the algorithm through a data stream. In addition, the line search algorithm used in Reference [3] may fail and is distinct from the line search proposed in this chapter. Hence, combining this line search algorithm with the one we propose may have a complementary effect when the properties of the optimization problem are disadvantageous to one of the algorithms. Reference [44] gives an algorithm for solving fixed point problems, covering the constrained minimization problem discussed in this chapter, with a line search. This algorithm has a fast convergence property, though it decides only the coefficient of the convex combination and is not designed for multi-core computing. The algorithm in [23, 24, 41, 42, 67] requires a suitable learning rate in order to converge efficiently. However, as we mentioned before, the learning rate is very difficult to adjust.

In contrast to previous reports, this chapter proposes incremental and parallel subgradient algorithms with a line search to find better learning rates than the ones used in the existing algorithms. To realize this proposal, we extend the concept of the *learning rate* to a *step-range*, which is a set of candidates for the learning rate. The line search procedure is given a step-range and chooses the most suitable learning rate among it at run-time. Using a line search with a step-range has three merits. First, the suitable learning rates chosen by the line search accelerate the algorithms and make their solutions better. Section 2.4 shows that the proposed algorithms gave better solutions than the one given by Pegasos [83] when they all ran the same number of iterations. The second merit is that we do not need to adjust the learning rate precisely. The existing incremental and parallel subgradient algorithms [24, 67] cannot converge efficiently without appropriate adjustments to their learning rates. This is their weak point in comparison with Pegasos [83]. In contrast, the proposed algorithms only need step-ranges, i.e. rough candidates, to converge efficiently, because the line search automatically chooses the learning rates from among the step-range. Hence, they can be easily used to learn support vector machines. Finally, the proposed algorithms can be applied to difficult problems whose suitable learning rates cannot be chosen beforehand. Section 2.3 provides a condition on the step-range composi-

tions to ensure they converge to an optimizer of the problem. Hence, even if a suitable learning rate cannot be specified beforehand, the line search can algorithmically find one at run-time and make the algorithms converge efficiently to an optimizer. We show that our algorithms converge to an optimizer to the problem when the step-range is diminishing. In addition, if the step-range is a singleton set, they coincide with the existing incremental and parallel subgradient algorithms [24, 67]. Hence, the step-range is a generalization of the learning rates used in the existing algorithms.

We compared the proposed algorithms with Pegasos [83] and the SMO algorithm on various datasets [15, 58, 61] for binary and multiclass classification. The results of the comparison demonstrated that the proposed algorithms perform better than the existing ones in terms of the value of the objective function for learning with a support vector machine and in terms of computational time. In particular, the parallel subgradient algorithm dramatically reduced the computational times of the learning tasks.

Stochastic subgradient algorithms are useful for learning with a multilayer neural network habitually [6]. The incremental subgradient algorithm is a specialization of the stochastic subgradient algorithm. Therefore, we can use one of our algorithms, a variant of the incremental subgradient algorithm, to train a multilayer neural network. We compared it with two other variants of the incremental subgradient algorithm. The results show that our algorithm can minimize the objective function of the trained neural network more than the others. This ability implies that it is also useful for training not only SVMs but also neural networks, including ones for deep learning.

This chapter is organized as follows. Section 2.2 gives the mathematical preliminaries and mathematical formulation of the main problem. Section 2.3 presents our algorithms. We also show the convergence analyses of these algorithms. Section 2.4 describes numerical comparisons of the proposed algorithms with the existing ones in Reference [24, 67, 83] using concrete machine learning datasets [15, 58, 61]. In this section, we also describe how to use one of the proposed algorithms to train a multilayer neural network for recognizing handwritten digits. Section 2.5 concludes this chapter.

Section 2.2. Mathematical preliminaries

Let $(H, \langle\cdot,\cdot\rangle)$ be a real Hilbert space with its induced norm defined by $\|x\| := \langle x, x\rangle^{\frac{1}{2}}$. We define the notation $\mathbb{R}_+ := (0, \infty)$ and $\mathbb{N} := \{1, 2, \ldots\}$. Let $x_n \to x$ denote that the sequence $\{x_n\}$ converges to x, and let $x_n \rightharpoonup x$ denote that the sequence $\{x_n\}$ converges weakly to x.

A *subgradient* g of a convex function $f : H \to \mathbb{R}$ at a point $x \in H$ is defined by $g \in H$ such that $f(x) + \langle y - x, g\rangle \leq f(y)$ for all $y \in H$. The set of all subgradients at x is denoted as $\partial f(x)$ [80], [87, Section 7.3].

The metric projection onto a nonempty, closed convex set $C \subset H$ is denoted by $P_C : H \to C$ and defined by $\|x - P_C(x)\| = \inf_{y\in C} \|x - y\|$ [2, Section 4.2, Chapter 28]. P_C satisfies the nonexpansivity condition [87, Subchapter 5.2]; i.e. $\|P_C(x) - P_C(y)\| \leq$

$\|x - y\|$ for all $x, y \in H$.

2.2.1 Main problem

Let $f_i : H \to [0, \infty)$ $(i = 1, 2, \ldots, K)$ be convex, continuous functions, and let C be a nonempty, closed convex subset of H. Then, we would like to

$$\begin{aligned} &\text{minimize } f(x) := \sum_{i=1}^{K} f_i(x) \\ &\text{subject to } x \in C. \end{aligned} \tag{2.1}$$

Let us discuss Problem (2.1) in the situation that a closed convex subset C of a real Hilbert space H is simple in the sense that P_C can be computed within a finite number of arithmetic operations. Examples of a simple, closed convex set C are a closed ball, a half-space, and the intersection of two half-spaces [2, Examples 3.16 and 3.21, and Proposition 28.19].

The task of learning with a support vector machine can also be cast as Problem (2.1) [83, Problem (1)]. Furthermore, there are a lot of applications not limited to learning with a support vector machine when f is nonsmooth but convex on H and when $C \subset H$ is simple. For example, minimizing the total variation of a signal over a convex set and Tykhonov-like problems with L^1-norms [12, I. Introduction] are able to be handled as Problem (2.1). Application of Problem (2.1) to learning with a support vector machine will be described in Section 2.4.

Throughout this chapter, we impose two assumptions: boundedness of the subgradients and the existence of an optimal solution.

Assumption 2.2.1 ([67, Assumption 2.1, Proposition 2.4])**.** We suppose that

(A1) for each $i = 1, 2, \ldots, K$, there exists $M_i > 0$ such that $\|g\| \leq M_i$ $(x \in C; g \in \partial f_i(x))$;
(A2) there exists at least one optimal solution, i.e., $\operatorname{argmin}_{x \in C} f(x) \neq \emptyset$.

In addition, we define a constant $M := \sum_{i=1}^{K} M_i$.

The first assumption is also used for analyzing the convergence of the existing incremental and parallel subgradient algorithms [24, 67]. Fortunately, this assumption is satisfied when the constraint set C is bounded [2, Proposition 16.17.(iii)].

Section 2.3. Proposed algorithms and their convergence analyses

2.3.1 Incremental subgradient algorithm

Algorithm 2.3.1 is the proposed variant of the incremental subgradient algorithm [67]. Let us compare Algorithm 2.3.1 with the existing one [67]. The difference is

Algorithm 2.3.1 Incremental subgradient algorithm

Require: $\forall n \in \mathbb{N}, [\underline{\lambda}_n, \overline{\lambda}_n] \subset \mathbb{R}_+$.
1: $n \leftarrow 1$, $x_1 \in C$.
2: **loop**
3: $y_{n,0} := x_n$.
4: **for** $i = 1, 2, \ldots, K$ **do** ▷ In sequence
5: $g_{n,i} \in \partial f_i(y_{n,i-1})$.
6: $\lambda_{n,i} \in [\underline{\lambda}_n, \overline{\lambda}_n]$. ▷ By a line search algorithm
7: $y_{n,i} := P_C(y_{n,i-1} - \lambda_{n,i} g_{n,i})$.
8: **end for**
9: $x_{n+1} := y_{n,K}$.
10: $n \leftarrow n + 1$.
11: **end loop**

step 6 of Algorithm 2.3.1. The learning rate λ_n of the existing algorithm must be decided before the algorithm runs. However, Algorithm 2.3.1 only needs the step-range $[\underline{\lambda}_n, \overline{\lambda}_n]$. A learning rate within the range used by Algorithm 2.3.1 can be automatically determined at run-time. Algorithm 2.3.1 coincides with the incremental subgradient algorithm when the given step-range $[\underline{\lambda}_n, \overline{\lambda}_n]$ is a singleton set, i.e. $\underline{\lambda}_n := \overline{\lambda}_n := \lambda_n$, which means that it is a generalization of the algorithm in [67]. In this case, Algorithm 2.3.1 chooses only one learning rate λ_n from the singleton step-range $[\underline{\lambda}_n, \overline{\lambda}_n] = \{\lambda_n\}$.

This difference has three merits. First, the suitably chosen learning rates in step 6 accelerate convergence and make the solutions more accurate. Second, Algorithm 2.3.1 does not require the learning rate to be precisely adjusted in order for it to converge efficiently, unlike the existing incremental subgradient algorithm [67]. Instead, Algorithm 2.3.1 only needs a rough step-range as the line search automatically chooses learning rates from among this range. Hence, it can easily be used to learn support vector machines. Finally, Algorithm 2.3.1 can be applied to problems in which a suitable learning rate cannot be chosen beforehand. Hence, even if the suitable learning rate cannot be specified, line search can algorithmically find this learning rate and make proposed algorithms converge efficiently to an optimizer.

Algorithm 2.3.1 has the following property, which is used for proving the main theorem.

Lemma 2.3.1. *Suppose that Assumption 2.2.1 holds. Let $\{x_n\}$ be a sequence generated by Algorithm 2.3.1. Then, for all $y \in C$ and for all $n \in \mathbb{N}$, the following inequality holds:*

$$\|x_{n+1} - y\|^2 \leq \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + \overline{\lambda}_n^2 M^2.$$

Proof. Fix $y \in C$ and $n \in \mathbb{N}$ arbitrarily. From the nonexpansivity of P_C, the definition of subgradients, and Assumption (A1), we have

$$\begin{aligned}
\|x_{n+1} - y\|^2 &= \|P_C(y_{n,K-1} - \lambda_{n,K} g_{n,K}) - P_C(y)\|^2 \\
&\leq \|y_{n,K-1} - y - \lambda_{n,K} g_{n,K}\|^2 \\
&= \|y_{n,K-1} - y\|^2 - 2\lambda_{n,K}\langle y_{n,K-1} - y, g_{n,K}\rangle + \lambda_{n,K}^2 \|g_{n,K}\|^2 \\
&\leq \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}\langle y_{n,i-1} - y, g_{n,i}\rangle + \sum_{i=1}^{K} \lambda_{n,i}^2 \|g_{n,i}\|^2 \\
&\leq \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(y_{n,i-1}) - f_i(y)) + \overline{\lambda}_n^2 \sum_{i=1}^{K} M_i^2,
\end{aligned}$$

where the second equation comes from $\|x - y\|^2 = \|x\|^2 - 2\langle x, y\rangle + \|y\|^2 \quad (x, y \in H)$. Using the definition of subgradients and the Cauchy-Schwarz inequality, we have

$$\begin{aligned}
&\|x_{n+1} - y\|^2 \\
&\leq \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(y_{n,i-1}) - f_i(x_n)) + \overline{\lambda}_n^2 \sum_{i=1}^{K} M_i^2. \\
&\leq \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + 2\overline{\lambda}_n \sum_{i=1}^{K} M_i \|y_{n,i-1} - x_n\| + \overline{\lambda}_n^2 \sum_{i=1}^{K} M_i^2.
\end{aligned}$$

Further, the nonexpansivity of P_C and the triangle inequality mean that, for all $i = 2, 3, \ldots, K$,

$$\begin{aligned}
\|y_{n,i-1} - x_n\| &= \|P_C(y_{n,i-2} - \lambda_{n,i-1} g_{n,i-1}) - P_C(x_n)\| \\
&\leq \|y_{n,i-2} - x_n - \lambda_{n,i-1} g_{n,i-1}\| \\
&\leq \|y_{n,i-2} - x_n\| + \lambda_{n,i-1}\|g_{n,i-1}\| \\
&\leq \|y_{n,i-2} - x_n\| + \overline{\lambda}_n M_{i-1} \\
&\leq \overline{\lambda}_n \sum_{j=1}^{i-1} M_j.
\end{aligned}$$

From the above inequality and the fact that $\|y_{n,0} - x_n\| = \|x_n - x_n\| = 0$, we find that

$$\begin{aligned}\|x_{n+1} - y\|^2 &\leq \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + 2\overline{\lambda}_n^2 \sum_{i=1}^{K} M_i \sum_{j=1}^{i-1} M_j + \overline{\lambda}_n^2 \sum_{i=1}^{K} M_i^2 \\ &= \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + \overline{\lambda}_n^2 \left(\sum_{i=1}^{K} M_i\right)^2 \\ &= \|x_n - y\|^2 - 2\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + \overline{\lambda}_n^2 M^2.\end{aligned}$$

This completes the proof. □

2.3.2 Parallel subgradient algorithm

Algorithm 2.3.2 below is an extension of the parallel subgradient algorithm [24]. The difference between Algorithm 2.3.2 and the algorithm in [24] is step 5. The

Algorithm 2.3.2 Parallel subgradient algorithm

Require: $\forall n \in \mathbb{N}, [\underline{\lambda}_n, \overline{\lambda}_n] \subset \mathbb{R}_+$.
1: $n \leftarrow 1$, $x_1 \in C$.
2: **loop**
3: **for all** $i \in \{1, 2, \ldots, K\}$ **do** ▷ Independently
4: $g_{n,i} \in \partial f_i(x_n)$.
5: $\lambda_{n,i} \in [\underline{\lambda}_n, \overline{\lambda}_n]$. ▷ By a line search algorithm
6: $y_{n,i} := P_C(x_n - \lambda_{n,i} g_{n,i})$.
7: **end for**
8: $x_{n+1} := \frac{1}{K} \sum_{i=1}^{K} y_{n,i}$.
9: $n \leftarrow n + 1$.
10: **end loop**

existing algorithm uses a given learning rate λ_n, while Algorithm 2.3.2 chooses a learning rate λ_n from the step range $[\underline{\lambda}_n, \overline{\lambda}_n]$ at run-time. We describe how to choose a suitable learning rate from the given step range in Subsection 2.3.3.

The common feature of Algorithm 2.3.2 and the parallel subgradient algorithm [24] is loop independence (step 3). This loop structure is not influenced by the computation order. Hence, each iteration of this loop can be computed in parallel. Therefore, parallelization using multi-core processing should be able to reduce the time needed for computing this loop procedure. Generally speaking, the main loop of Algorithm 2.3.2 is computationally heavier than the other subgradient algorithms including Pegasos, because it appends the learning rate selection (line search) procedure to the existing

one. However, parallelization alleviates this effect of the line search procedure (This is shown in Section 2.4).

The sequence generated by Algorithm 2.3.2 satisfies the following property.

Lemma 2.3.2. *Suppose that Assumption 2.2.1 holds. Let $\{x_n\}$ be a sequence generated by Algorithm 2.3.2. Then, for all $y \in C$ and for all $n \in \mathbb{N}$, the following inequality holds:*

$$\|x_{n+1} - y\|^2 \leq \|x_n - y\|^2 - \frac{2}{K}\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + \overline{\lambda}_n^2 M^2.$$

Proof. Fix $y \in C$ and $n \in \mathbb{N}$ arbitrarily. From the convexity of $\|\cdot\|^2$, the nonexpansivity of P_C, the definition of subgradients, and Assumption (A1), we have

$$\begin{aligned}
\|x_{n+1} - y\|^2 &= \left\| \frac{1}{K}\sum_{i=1}^{K} P_C(x_n - \lambda_{n,i} g_{n,i}) - P_C(y) \right\|^2 \\
&\leq \frac{1}{K}\sum_{i=1}^{K} \|x_n - y - \lambda_{n,i} g_{n,i}\|^2 \\
&= \frac{1}{K}\sum_{i=1}^{K} (\|x_n - y\|^2 - 2\lambda_{n,i}\langle x_n - y, g_{n,i}\rangle + \lambda_{n,i}^2 \|g_{n,i}\|^2) \\
&\leq \|x_n - y\|^2 - \frac{2}{K}\sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + \overline{\lambda}_n^2 M^2.
\end{aligned}$$

This completes the proof. □

2.3.3 Line search algorithms

Step 6 of Algorithm 2.3.1 and step 5 of Algorithm 2.3.2 are implemented as line search algorithms. They decide an efficient learning rate λ_n in $[\underline{\lambda}_n, \overline{\lambda}_n]$ by using $y_{n,i-1}$ in Algorithm 2.3.1 (or x_n in Algorithm 2.3.2), $g_{n,i}$, f_i and other accessible information on i. This is the principal idea of this chapter. We can use any algorithm that satisfies the above condition. The following are such examples.

The simplest line search is the *discrete argmin*, as shown in Algorithm 2.3.3. First, we set the ratio candidates $\{L_1, L_2, \ldots, L_k\} \subset [0, 1]$. In each iteration, we compute all of the candidate objectives for the learning rate $\lambda_{n,i} = L_t\overline{\lambda}_n + (1 - L_t)\underline{\lambda}_n \quad (t = 1, 2, \ldots, k)$ and take the best one.

Algorithm 2.3.4 is a line search based on the Wolfe conditions. It finds a learning rate that satisfies the sufficient decrease condition with logarithmic grids. Once this learning rate has been found, the algorithm stops and the learning rate it found is used in the caller algorithm. However, this algorithm may fail (step 8). To avoid such a failure, we can make the caller algorithm use $\underline{\lambda}_n$. This is the largest learning rate

Algorithm 2.3.3 Discrete argmin line search algorithm

1: $x_p := \begin{cases} y_{n,i-1} & \text{(Algorithm 2.3.1)}, \\ x_n & \text{(Algorithm 2.3.2)} \end{cases}$.
2: $\lambda_{n,i} \leftarrow L_1\overline{\lambda}_n + (1 - L_1)\underline{\lambda}_n$.
3: **for** $L_t \in \{L_2, L_3, \ldots, L_k\}$ **do**
4: $\quad t \leftarrow L_t\overline{\lambda}_n + (1 - L_t)\underline{\lambda}_n$.
5: $\quad$ **if** $f_i(P_C(x_p - tg_{n,i})) < f_i(P_C(x_p - \lambda_{n,i}g_{n,i}))$ **then**
6: $\quad\quad \lambda_{n,i} \leftarrow t$
7: $\quad$ **end if**
8: **end for**

Algorithm 2.3.4 Logarithmic-interval armijo line search algorithm

1: $x_p := \begin{cases} y_{n,i-1} & \text{(Algorithm 2.3.1)}, \\ x_n & \text{(Algorithm 2.3.2)} \end{cases}$.
2: **for** $I_R = 1, 1/a, 1/a^2, \ldots, 1/a^k$ **do**
3: $\quad \lambda_{n,i} \leftarrow I_R\overline{\lambda}_n + (1 - I_R)\underline{\lambda}_n$.
4: $\quad$ **if** $f_i(P_C(x_p - \lambda_{n,i}g_{n,i})) \leq f_i(x_p) - c_1\langle x_p - P_C(x_p - \lambda_{n,i}g_{n,i}), g_{n,i}\rangle$ **then**
5: $\quad\quad$ **stop** (success).
6: $\quad$ **end if**
7: **end for**
8: **stop** (failed).

of the candidates for making an effective update of the solution. The results of the experiments described in Section 2.4 demonstrate effectiveness of this algorithm[*1].

Here, let us consider whether the proposed algorithms can be applied to problems with a general convex objective function or not, that is, when the range of the objective function f is not limited to $[0, \infty)$ but $\mathbb{R}$. In this discussion, suppose that all the objective functions are bounded from below by some constant on the constraint set; i.e., there exists a constant $m \in \mathbb{R}$ such that $m \leq f_i(x)$ for all $i = 1, 2, \ldots, K$ and for any $x \in C$. Let us define a function $\hat{f}_i := f_i + m$ for each $i = 1, 2, \ldots, K$. Then, the function $\hat{f}_i$ is also convex and its range is nonnegative for all $i = 1, 2, \ldots, K$. For any

[*1] In this case, i.e., when we use Algorithm 2.3.1 or Algorithm 2.3.2 with Algorithm 2.3.4, we have to give a parameter set $\{\underline{\lambda}_n, \overline{\lambda}_n\}$ for the step-range, a constant c_1 appearing in the Armijo condition, a common ratio a and the number of trials k of Algorithm 2.3.4 as hyperparameters.

$x \in C$, the subdifferential $\partial\hat{f}(x)$ satisfies that

$$\begin{aligned}\partial\hat{f}(x) &= \{g \in H : \hat{f}(x) + \langle y - x, g\rangle \leq \hat{f}(y) \text{ for all } y \in H\}\\ &= \{g \in H : f(x) + m + \langle y - x, g\rangle \leq f(y) + m \text{ for all } y \in H\}\\ &= \{g \in H : f(x) + \langle y - x, g\rangle \leq f(y) \text{ for all } y \in H\}\\ &= \partial f(x).\end{aligned}$$

Hence, the subdifferential $\partial\hat{f}$ coincides with ∂f on the set C. This implies that, if the same learning rates are chosen from the step ranges, Algorithm 2.3.1 (or Algorithm 2.3.2) generates the same sequences when it is applied to the original problem or the problem whose objective function is replaced by the function $\hat{f}$. In particular, as shown in step 4, the step size decided by Algorithm 2.3.4 is influenced only by the relative difference between $f_i(P_C(x_p - tg_{n,i}))$ and $f_i(P_C(x_p - \lambda_{n,i}g_{n,i}))$. Hence, the line search algorithm decides the same learning rate in either case. Obviously, the set of optimal solutions $\operatorname{argmin}_{x\in C} f(x)$ coincides with $\operatorname{argmin}_{x\in C} \hat{f}(x)$. Therefore, the theorems presented in this chapter hold when the ranges of objective functions are not only the set of nonnegative reals but also any subset bounded from below.

2.3.4 Convergence analysis of Algorithms 2.3.1 and 2.3.2

Here, we first show that the limit inferiors of $\{f(x_n)\}$ generated by Algorithms 2.3.1 and 2.3.2 are equal to the optimal value of f. Next, we show that $\{x_n\}$ converges weakly to a solution of the main problem (2.1). The following assumption is used to show the convergence of Algorithms 2.3.1 and 2.3.2.

Assumption 2.3.1.

$$\sum_{n=1}^{\infty} \overline{\lambda}_n = \infty, \quad \sum_{n=1}^{\infty} \overline{\lambda}_n^2 < \infty, \quad \lim_{n\to\infty} \frac{\underline{\lambda}_n}{\overline{\lambda}_n} = 1, \quad \sum_{n=1}^{\infty} (\overline{\lambda}_n - \underline{\lambda}_n) < \infty.$$

The following lemma states that some subsequence of the objective function value of the generated sequence converges to the optimal value. This lemma is used to prove the main theorem described next.

Lemma 2.3.3. *Suppose that Assumptions 2.2.1 and 2.3.1 hold. For a sequence $\{x_n\}$, if there exists $\alpha \in \mathbb{R}_+$ such that, for all $y \in C$ and for all $n \in \mathbb{N}$,*

$$\|x_{n+1} - y\|^2 \leq \|x_n - y\|^2 - 2\alpha \sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(y)) + \overline{\lambda}_n^2 M^2, \tag{2.2}$$

then,

$$\underline{\lim}_{n\to\infty} f(x_n) = \min_{x\in C} f(x).$$

Proof. Assume that $\underline{\lim}_{n\to\infty}\sum_{i=1}^{K}(\lambda_{n,i}/\overline{\lambda}_n)f_i(x_n) \neq \min_{x\in C} f(x)$. Then, either $\underline{\lim}_{n\to\infty}\sum_{i=1}^{K}(\lambda_{n,i}/\overline{\lambda}_n)f_i(x_n) < \min_{x\in C} f(x)$ or $\min_{x\in C} f(x) < \underline{\lim}_{n\to\infty}\sum_{i=1}^{K}(\lambda_{n,i}/\overline{\lambda}_n)f_i(x_n)$ holds. First, we assume $\underline{\lim}_{n\to\infty}\sum_{i=1}^{K}(\lambda_{n,i}/\overline{\lambda}_n)f_i(x_n) < \min_{x\in C} f(x)$. Recall $\{x_n\} \subset C$ and the definition $f(x) := \sum_{i=1}^{K} f_i(x)$ in the main problem (2.1). The property of the limit inferior and [89, Exercise 4.1.31] ensure that

$$\begin{aligned}\min_{x\in C} f(x) &\leq \underline{\lim}_{n\to\infty}\ f(x_n)\\ &= \underline{\lim}_{n\to\infty}\frac{\underline{\lambda}_n}{\overline{\lambda}_n}\sum_{i=1}^{K} f_i(x_n)\end{aligned}$$

Further, from the nonnegativity of $f_i\,(i = 1, 2, \ldots, K)$, the fact that $\underline{\lambda}_n \leq \lambda_{n,i}$ and the assumption that $\underline{\lim}_{n\to\infty}\sum_{i=1}^{K}(\lambda_{n,i}/\overline{\lambda}_n)f_i(x_n) < \min_{x\in C} f(x)$ lead to

$$\begin{aligned}\min_{x\in C} f(x) &\leq \underline{\lim}_{n\to\infty}\sum_{i=1}^{K}\frac{\underline{\lambda}_n}{\overline{\lambda}_n} f_i(x_n)\\ &\leq \underline{\lim}_{n\to\infty}\sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n)\\ &< \min_{x\in C} f(x).\end{aligned}$$

This is a contradiction. Next, we assume $\min_{x\in C} f(x) < \underline{\lim}_{n\to\infty}\sum_{i=1}^{K}(\lambda_{n,i}/\overline{\lambda}_n)f_i(x_n)$ and let $\hat{y} \in \operatorname{argmin}_{x\in C} f(x)$. Then, there exists $\varepsilon > 0$ such that

$$f(\hat{y}) + 2\varepsilon = \underline{\lim}_{n\to\infty}\sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n).$$

From the definition of the limit inferior, there exists $n_0 \in \mathbb{N}$ such that, for all $n \in \mathbb{N}$, if $n_0 \leq n$, then

$$\underline{\lim}_{n\to\infty}\sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n) - \varepsilon < \sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n).$$

Now, $\lambda_{n,i}/\overline{\lambda}_n \leq 1$ and $0 \leq f_i(\hat{y}) \quad (i = 1, 2, \ldots, K)$ hold. Therefore, for all $n \in \mathbb{N}$, if $n_0 \leq n$, then

$$\begin{aligned}\varepsilon &= \underline{\lim}_{n\to\infty}\sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n) - \varepsilon - f(\hat{y})\\ &< \sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n) - \sum_{i=1}^{K} f_i(\hat{y})\\ &\leq \sum_{i=1}^{K}\frac{\lambda_{n,i}}{\overline{\lambda}_n}(f_i(x_n) - f_i(\hat{y})).\end{aligned}$$

From inequality (2.2), for all $n \in \mathbb{N}$, if $n_0 \leq n$, we have

$$\begin{aligned}\|x_{n+1} - \hat{y}\|^2 &\leq \|x_n - \hat{y}\|^2 - 2\alpha \sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(\hat{y})) + \overline{\lambda}_n^2 M^2 \\ &\leq \|x_n - \hat{y}\|^2 - 2\alpha\overline{\lambda}_n\varepsilon + \overline{\lambda}_n^2 M^2 \\ &= \|x_n - \hat{y}\|^2 - \overline{\lambda}_n(2\alpha\varepsilon - \overline{\lambda}_n M^2).\end{aligned}$$

From Assumption 2.3.1, $n_1 \in \mathbb{N}$ exists such that $n_0 \leq n_1$, and, for all $n \in \mathbb{N}$, if $n_1 \leq n$, $\overline{\lambda}_n \leq \alpha\varepsilon/M^2$. Hence, if $n_1 \leq n$, we have

$$\begin{aligned}0 &\leq \|x_{n+1} - \hat{y}\|^2 \\ &\leq \|x_n - \hat{y}\|^2 - \alpha\varepsilon\overline{\lambda}_n \\ &\leq \|x_{n_1} - \hat{y}\|^2 - \alpha\varepsilon \sum_{k=n_1}^{n} \overline{\lambda}_k.\end{aligned}$$

for all $n \in \mathbb{N}$. From Assumption 2.3.1, the right side diverges negatively, which is a contradiction. Overall, we have

$$\varliminf_{n\to\infty} \sum_{i=1}^{K} \frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n) = \min_{x\in C} f(x).$$

Next, let us assume that $\varliminf_{n\to\infty} \sum_{i=1}^{K} (\lambda_{n,i}/\overline{\lambda}_n) f_i(x_n) \neq \varliminf_{n\to\infty} f(x_n)$. Now, $\lambda_{n,i}/\overline{\lambda}_n \leq 1$ and $0 \leq f_i(x_n) \quad (i = 1, 2, \ldots, N)$ hold for all $n \in \mathbb{N}$. Therefore, we have

$$\varliminf_{n\to\infty} \sum_{i=1}^{K} \frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n) \leq \varliminf_{n\to\infty} f(x_n).$$

Hence, from the nonnegativity of f_i $(i = 1, 2, \ldots, K)$, the fact that $\underline{\lambda}_n \leq \lambda_{n,i}$, and [89, Exercise 4.1.31], we have

$$\begin{aligned}\varliminf_{n\to\infty} f(x_n) &= \varliminf_{n\to\infty} \frac{\underline{\lambda}_n}{\overline{\lambda}_n} f(x_n) \\ &\leq \varliminf_{n\to\infty} \sum_{i=1}^{K} \frac{\lambda_{n,i}}{\overline{\lambda}_n} f_i(x_n) \\ &< \varliminf_{n\to\infty} f(x_n).\end{aligned}$$

However, this is a contradiction. This completes the proof. □

The following is the main theorem of this chapter.

Theorem 2.3.1. *Suppose that Assumptions 2.2.1 and 2.3.1 hold. The sequence* $\{x_n\}$ *generated by Algorithm 2.3.1 or 2.3.2 converges weakly to an optimal solution to the main problem* (2.1)*.*

Proof. Let $\hat{y} \in \operatorname{argmin}_{x\in C} f(x)$ and fix $n \in \mathbb{N}$. From Lemmas 2.3.1 and 2.3.2, there exists $\alpha \in \mathbb{R}_+$ such that

$$\|x_{n+1} - \hat{y}\|^2 \leq \|x_n - \hat{y}\|^2 - 2\alpha \sum_{i=1}^{K} \lambda_{n,i}(f_i(x_n) - f_i(\hat{y})) + \overline{\lambda}_n^2 M^2.$$

By $0 \leq f_i(\hat{y}), f_i(x_n) \quad (i = 1, 2, \ldots, K)$, we have

$$\begin{aligned}
\|x_{n+1} - \hat{y}\|^2 &\leq \|x_n - \hat{y}\|^2 - 2\alpha\underline{\lambda}_n \sum_{i=1}^{K} f_i(x_n) + 2\alpha\overline{\lambda}_n \sum_{i=1}^{K} f_i(\hat{y}) + \overline{\lambda}_n^2 M^2 \\
&= \|x_n - \hat{y}\|^2 - 2\alpha\underline{\lambda}_n \sum_{i=1}^{K} (f_i(x_n) - f_i(\hat{y})) + 2\alpha(\overline{\lambda}_n - \underline{\lambda}_n) \sum_{i=1}^{K} f_i(\hat{y}) + \overline{\lambda}_n^2 M^2 \\
&\leq \|x_n - \hat{y}\|^2 + 2\alpha f(\hat{y})(\overline{\lambda}_n - \underline{\lambda}_n) + \overline{\lambda}_n^2 M^2 \\
&\leq \|x_1 - \hat{y}\|^2 + 2\alpha f(\hat{y}) \sum_{i=1}^{n} (\overline{\lambda}_i - \underline{\lambda}_i) + M^2 \sum_{i=1}^{n} \overline{\lambda}_i^2.
\end{aligned}$$

From Assumption 2.3.1, the left side of the above inequality is bounded. Hence, $\{x_n\}$ is bounded. Using [4, Lemma 1.7], $J \in \mathbb{R}$ exists for all $\hat{y} \in \operatorname{argmin}_{x\in C} f(x)$ such that $\lim_{n\to\infty} \|x_n - \hat{y}\| = J$. Moreover, from Lemma 2.3.3, a subsequence $\{f(x_{n_i})\} \subset \{f(x_n)\}$ exists such that $\lim_{i\to\infty} f(x_{n_i}) = f(\hat{y})$. From [2, Theorem 3.32], C is a weak closed set. Therefore, there exists a subsequence $\{x_{n_{i_j}}\} \subset \{x_{n_i}\}$ and a point $u \in C$ such that $x_{n_{i_j}} \rightharpoonup u$. Hence, from [2, Theorem 9.1, Proposition 9.33], we obtain

$$\begin{aligned}
\min_{x\in C} f(x) &\leq f(u) \\
&\leq \varliminf_{j\to\infty} f(x_{n_{i_j}}) \\
&= \min_{x\in C} f(x).
\end{aligned}$$

This implies that $u \in \operatorname{argmin}_{x\in C} f(x)$. Let $\{x_{n_{i_k}}\} \subset \{x_{n_i}\}$ be another subsequence and assume $x_{n_{i_k}} \rightharpoonup v \in \operatorname{argmin}_{x\in C} f(x)$ and $u \neq v$. From [70, Lemma 1], we have

$$\begin{aligned}
\lim_{n\to\infty} \|x_n - u\| &= \lim_{j\to\infty} \|x_{n_{i_j}} - u\| < \lim_{j\to\infty} \|x_{n_{i_j}} - v\| = \lim_{n\to\infty} \|x_n - v\| \\
&= \lim_{k\to\infty} \|x_{n_{i_k}} - v\| < \lim_{k\to\infty} \|x_{n_{i_k}} - u\| = \lim_{k\to\infty} \|x_n - u\|.
\end{aligned}$$

This is a contradiction. Accordingly, any subsequence of $\{x_{n_i}\}$ weakly converges to $u \in \operatorname{argmin}_{x\in C} f(x)$. Therefore, from [87, Theorem 5.4.1], $x_{n_i} \rightharpoonup u$. Now let

$\{x_{n_j}\} \subset \{x_n\}$ be another subsequence and assume $x_{n_j} \rightharpoonup w \neq u$. Then, from [70, Lemma 1], we have

$$\begin{aligned}\lim_{n\to\infty} \|x_n - u\| &= \lim_{i\to\infty} \|x_{n_i} - u\| < \lim_{i\to\infty} \|x_{n_i} - w\| = \lim_{n\to\infty} \|x_n - w\| \\ &= \lim_{j\to\infty} \|x_{n_j} - w\| < \lim_{j\to\infty} \|x_{n_j} - u\| = \lim_{n\to\infty} \|x_n - u\|.\end{aligned}$$

This is a contradiction. Therefore, any subsequence of $\{x_n\}$ weakly converges to $u \in \operatorname{argmin}_{x\in C} f(x)$. Hence, by [87, Theorem 5.4.1], $x_n \rightharpoonup u$. This completes the proof. □

2.3.5 Convergence rates

To show the convergence rates of Algorithms 2.3.1 and 2.3.2, we assume $\underline{\lambda}_n := \overline{\lambda}_n := 1/n$ for all $n \in \mathbb{N}$. We also assume the existence of $\mu \in (0, \infty)$ such that

$$f(x) - f(\hat{y}) \geq \mu \|x - \hat{y}\|^2 \quad \left(x \in C, \hat{y} \in \operatorname*{argmin}_{u\in C} f(u)\right). \tag{2.3}$$

The strong convexity of f implies Condition (2.3)[66, Inequality (16)]. First, We give the following lemma, which is required to show the convergence rates of Algorithms 2.3.1 and 2.3.2.

Lemma 2.3.4. ([66, Lemma 2.1], [77, Lemma 4]) *Let $\{u_n\} \subset [0, \infty)$ be such that*

$$u_{n+1} \leq \left(1 - \frac{p}{n}\right) u_n + \frac{d}{n^2} \quad (n \in \mathbb{N})$$

for some $p, d \in (0, \infty)$. Then

$$\begin{cases} u_n = O(\frac{1}{n^p}) & (p < 1), \\ u_n = O(\frac{\log n}{n}) & (p = 1), \\ u_n \leq \frac{d}{n(p-1)} + o(\frac{1}{n}) & (p > 1). \end{cases}$$

Next, we prove two propositions that show the convergence rates of Algorithms 2.3.1 and 2.3.2.

Proposition 2.3.1 (Convergence Rate of Algorithm 2.3.1)**.** *Let $\{x_n\}$ be a sequence generated by Algorithm 2.3.1 and $\hat{y} \in \operatorname{argmin}_{y\in C} f(y)$. Then, the following hold:*

$$\begin{cases} \|x_{n+1} - \hat{y}\| = O(\frac{1}{n^{2\mu}}) & (2\mu < 1), \\ \|x_{n+1} - \hat{y}\| = O(\frac{\log n}{n}) & (2\mu = 1), \\ \|x_{n+1} - \hat{y}\| \leq \frac{M^2}{n(2\mu-1)} + o(\frac{1}{n}) & (2\mu > 1). \end{cases}$$

Proof. From Lemma 2.3.1 and inequality (2.3), we have

$$\|x_{n+1} - \hat{y}\|^2 \leq \left(1 - \frac{2\mu}{n}\right) \|x_n - \hat{y}\|^2 + \frac{M^2}{n^2}.$$

for all $n \in \mathbb{N}$. Lemma 2.3.4 with $p := 2\mu, d := M^2$ completes the proof. □

This result implies that Algorithm 2.3.1 is in the same class of convergence efficiency as the incremental subgradient algorithm[66, Proposition 2.8].

Proposition 2.3.2 (Convergence Rate of Algorithm 2.3.2)**.** *Let $\{x_n\}$ be a sequence generated by Algorithm 2.3.2 and $\hat{y} \in \operatorname{argmin}_{y \in C} f(y)$. Then, the following hold:*

$$\begin{cases} \|x_{n+1} - \hat{y}\| = O(\frac{1}{n^{2\mu/K}}) & \left(\mu < \frac{K}{2}\right), \\ \|x_{n+1} - \hat{y}\| = O(\frac{\log n}{n}) & \left(\mu = \frac{K}{2}\right), \\ \|x_{n+1} - \hat{y}\| \leq \frac{M^2}{n(2\mu/K-1)} + o(\frac{1}{n}) & \left(\mu > \frac{K}{2}\right). \end{cases}$$

Proof. From Lemma 2.3.2 and inequality (2.3), we have

$$\|x_{n+1} - y\|^2 \leq \left(1 - \frac{2\mu}{nK}\right) \|x_n - \hat{y}\|^2 + \frac{M^2}{n^2}.$$

for all $n \in \mathbb{N}$. Lemma 2.3.4 with $p := 2\mu/K, d := M^2$ completes the proof. □ □

To above analyses assumed $\underline{\lambda}_n = \overline{\lambda}_n$. However, Algorithms 2.3.1 and 2.3.2 can use different values of $\underline{\lambda}_n$ and $\overline{\lambda}_n$. This implies that Algorithms 2.3.1 and 2.3.2 may converge faster than theoretical rates given here.

Section 2.4. Experiments

In this section, we present the results of experiments evaluating our algorithms and comparing them with the existing algorithms. For our experiments, we used a MacPro (Late 2013) computer with a 3GHz 8-Core Intel Xeon E5 CPU, 32GB 1866MHz DDR3 memory, and 500GB flash storage. The operating system was MacOS Sierra (version 10.12.6). The experimental codes were written in Python 3.6 and ran on the CPython implementation.

2.4.1 Validation of convergence properties with a simple problem

A concrete test problem with a closed-form solution is a good way to evaluate the performance of algorithms in detail [90, 91]. Here, we used the existing and proposed algorithms to solve a simple problem. The goals were to compare their performances under equal conditions, to use the best parameters for each algorithm calculated from the theoretical analyses, and to evaluate these algorithms with the detailed indicators such as the distance between an acquired solution and the actual solution of the test problem. The test problem is as follows.

Problem 2.4.1 (Test problem)**.** Let $f_i(x) := (i+1)x_i^2$ $(x \in \mathbb{R}^N; i = 1, 2, \ldots, K)$, $c \in \mathbb{R}^N$, and $r \in \mathbb{R}$. Then, we would like to

$$\begin{aligned} &\text{minimize } f(x) := \sum_{i=1}^{K} f_i(x) \\ &\text{subject to } \|x - c\| \leq r \text{ and } x_i = 0 \ (i = 3, 4, \ldots, N). \end{aligned}$$

This problem is obviously an instance of Problem 2.1. Of course, the continuity of the objective function and the boundedness of the constraint ensure the above problem satisfies Assumption 1. We set $c := (2, 1, 0, \ldots, 0)^\top$ and $r := 1$. The optimal solution is accordingly $x^\star = ((2 - /\sqrt{2})/2, (2 - \sqrt{2})/2, 0, \ldots, 0)$.

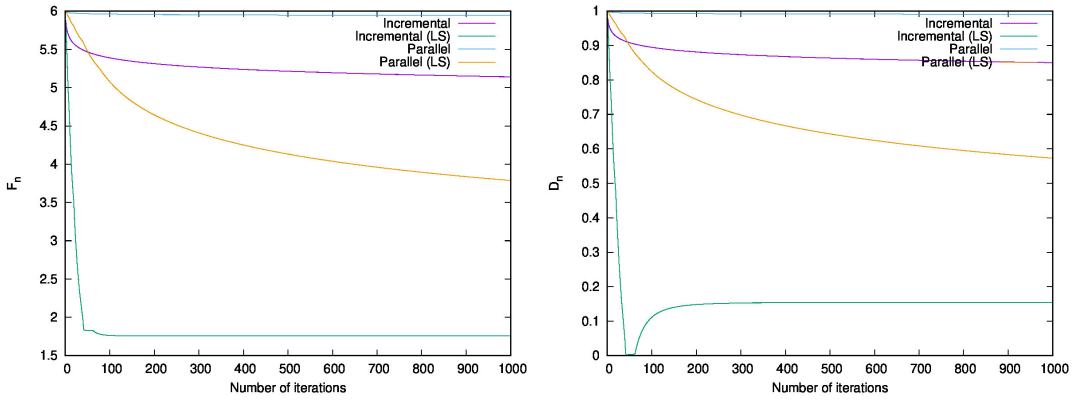
We set the number of dimensions to $N := 16$, i.e., equal to the number of logical cores of the experimental computer. We gave $x_1 := (2, 1, 0, \ldots, 0)^\top$, the center of the feasible set, as an initial point. We selected the incremental and parallel subgradient algorithms for comparison. These algorithms use a priori given learning rates; that is, they coincide with Algorithms 2.3.1, 2.3.2 with the settings $\underline{\lambda}_n = \overline{\lambda}_n = \lambda_n \in (0, \infty)$ $(n \in \mathbb{N})$. In this comparison, we gave learning rates of $\lambda_n := 1/(nN^2)$, which are appropriately chosen based on the following proposition related to the existing algorithms.

Proposition 2.4.1 ([40, Lemma 2.1], [94, Lemma 3.1])**.** *Suppose that $f : \mathbb{R}^N \to \mathbb{R}$ is c-strongly convex and differentiable, $\nabla f : \mathbb{R}^N \to \mathbb{R}^N$ is L-Lipschitz continuous, and $\mu \in (0, 2c/L^2)$. Define $T : \mathbb{R}^N \to \mathbb{R}^N$ by $T(x) := x - \mu \nabla f(x)$ $(x \in \mathbb{R}^N)$. Then, T is a contractive mapping.*

We set $\underline{\lambda}_n := 100/((n + 10000)N^2), \overline{\lambda}_n := 100/(nN^2)$ for each $n \in \mathbb{N}$ as the parameters of the proposed algorithms. This step-range contains the learning rates of the existing algorithms. We used Algorithm 2.3.4 with the parameters $c_1 := 0.99$, $a := 0.5$, and $k := 7$. We limited the iterations to 1,000 and evaluated the following indicators:

- F_n: value of the objective function, i.e., $F_n := \sum_{i=1}^{K} f_i(x_n)$,
- D_n: distance to the optimal solution, i.e., $D_n := \|x^\star - x_n\|$,
- T_n: running time of the algorithm.

The behaviors of $\{F_n\}$ and $\{D_n\}$ in each iteration n are shown in Figure 2.1. The result of Algorithm 2.3.1 dropped to the optimal dramatically in terms of both $\{F_n\}$ and $\{D_n\}$ within the first fifty iterations. The graphs of the other algorithms decreased similarly, but those of the algorithms with the line search decreased faster. Table 2.1 lists the running times of the existing and proposed algorithms for 1,000 iterations. Compared with the existing algorithms, the proposed algorithms needed a bit more time for running. However, they dramatically reduced the value of the objective function. Therefore, they converged faster that the existing algorithms.



(a) Behavior of F_n in each iteration n (b) Behavior of D_n in each iteration n

Fig. 2.1: Numerical comparison of running the existing and proposed algorithms on the test problem 2.4.1

Table 2.1: Running time of each algorithm in solving the test problem 2.1

Algorithm	Running time (T_{1000} [s])
Incremental subgradient algorithm	0.34341614
Algorithm 2.3.1	0.97763861
Parallel subgradient algorithm	0.11232432
Algorithm 2.3.2	0.24512658

2.4.2 Comparison of the existing and proposed algorithms in the task of learning with support vector machines

This subsection compares Algorithms 2.3.1 and 2.3.2 with Pegasos [83]. To evaluate their performance, we applied them to the following learning task.

Problem 2.4.2 (The task of learning with a support vector machine [83])**.** Let C be a positive real number. Given a training set $\{(x_i, y_i)\}$, where $x_i \in \mathbb{R}^N \quad (i = 1, 2, \ldots, K)$ and $y_i \in \{1, -1\} \quad (i = 1, 2, \ldots, K)$, we would like to

$$\text{minimize } f(w) := \frac{1}{C}\|w\|^2 + \frac{1}{K}\sum_{i=1}^{K} \max\{0, 1 - y_i \langle w, x_i \rangle\}$$
$$\text{subject to } w \in X := \{w : \|w\| \leq \sqrt{C}\}.$$

This optimization problem is introduced in [83] for learning with a support vector machine. The first term of the objective function is a penalty term that depends on the constraint set, and the second term is a loss function. The loss function returns higher values if the learner w can not classify an instance (x_i, y_i) correctly. The norm value

of the learner w does not affect the classification results due to the immutability of the signs of the decision function $\langle w, x_i \rangle$. Therefore, we can limit this value to a constant C. Now, let $f_i(w) := ((1/C)\|w\|^2 + \max\{0, 1 - y_i \langle w, x_i \rangle\})/K$. Then, $f = \sum_{i=1}^{K} f_i$ holds and Problem 2.4.2 can be handled as an instance of Problem (2.1).

We used the machine learning datasets shown in Table 2.2. The "australian" data

Table 2.2: Datasets used in our experiments

Name	#Instances	#Attributes	Missing Values	Attribute Characteristics
Iris (binary class)	100	4	No	Real
Iris (multiclass)	150	4	No	Real
Australian	590	14	No	Real
Horse-colic	368	27	Yes	Categorical, Integer, Real
Breast-cancer-wisconsin	699	10	Yes	Integer
Census-income	48842	14	Yes	Categorical, Integer
Internet-advertisements	3279	1558	Yes	Categorical, Integer, Real
MNIST	14780	784	No	Integer
RANDOM1	20	100	No	Real
RANDOM2	200	1000	No	Real

set is from LIBSVM Data [61]. The "MNIST" data set contains handwritten "0" and "1" digits and is provided by [58]. The "RANDOM1" and "RANDOM2" datasets were generated using the `sklearn.datasets.make_classification` function with a fixed `random_state`. The others are from the UCI Machine Learning Repository [15]. The number of classes of "iris (multiclass)" is three, and the others are binary classification datasets.

Missing values were complemented by using the `sklearn.impute.SimpleImputer` class. Categorical attributes were binarized using the `sklearn.preprocessing.OneHotEncoder` class. Each data set was scaled using the `sklearn.preprocessing.StandardScaler` class. These preprocessing methods and classes are from the scikit-learn [73] package for Python3.

The Pegasos algorithm used for this comparison is listed as Algorithm 2.4.1.

Algorithm 2.4.1 Pegasos [83, Fig. 1]

1: $n \leftarrow 1, w_1 \in X$.
2: **loop**
3: $\quad i_n \in \{1, 2, \ldots, K\}$. ▷ Chosen uniformly at random
4: $\quad g \in \partial f_{i_n}(w_n)$.
5: $\quad \lambda_n := C/n$.
6: $\quad w_{n+1} \leftarrow P_X(w_n - \lambda_n g_n)$.
7: $\quad n \leftarrow n + 1$.
8: **end loop**

We set $C := 10^{-1}$ and gave $\lambda_n := 10^{-1}/(n+10^8)$ to Algorithms 2.3.1 and 2.3.2. We

Table 2.3: Iris (binary class)

Algorithm	Time [sec]	Score (Training)	Score (Test)	Objective
Pegasos	0.12741225	1.00000000	1.00000000	0.95967555
Algorithm 2.3.1	0.28701049	1.00000000	1.00000000	0.94237229
Algorithm 2.3.2	0.03734168	1.00000000	1.00000000	0.91133305
SMO Algorithm	0.00515115	1.00000000	1.00000000	–

used Algorithm 2.3.4 with $c_1 := 0.99$ for the line search step in Algorithms 2.3.1 and 2.3.2. The main loops in Algorithms 2.3.1 and 2.4.1 were iterated $100K$ times, while the main loop in Algorithm 2.3.2 was iterated 100 times. This setting means that the algorithms could refer to each of the functions $f_i\,(i = 1, 2, \ldots, K)$ 1000 times.

We added scores of the SMO algorithm, one of the major algorithms for learning with a support vector machine, to the experimental results for each dataset. We used the implementation of the SMO algorithm in Python [*2] for calculating these scores.

First, let us look at the results for the iris (binary class) data set. Table 2.3 lists the computational times for learning, the classification scores on the training and test sets, and the values of the objective function. We used the `sklearn.model_selection.train_test_split` method provided by the scikit-learn package [73] to split the dataset into training and test sets. The number of instances in the training set was 30 and the number of instances in the test set was 70. The results indicate that Algorithm 2.3.2 performed better than Pegasos and Algorithm 2.3.1 in terms of computational time and value of the objective function. In addition, Algorithm 2.3.1 worked out a better approximation than Pegasos did in terms of the objective function. Hence, Algorithms 2.3.1 and 2.3.2 ran more efficiently than the existing algorithm. However, the SMO algorithm ran more quickly than the other algorithms, while keeping the highest score.

Next, let us look at the results of the multiclass classification using the iris (multiclass) dataset. Table 2.4 lists the computational times for learning and the classification scores on the training and test sets. We used the `sklearn.model_selection.train_test_split` method provided by the scikit-learn package [73] to split the dataset into training and test sets. To construct multiclass classifiers from Algorithm 2.3.1, 2.3.2, and 2.4.1, we used `sklearn.multiclass.OneVsRestClassifier` class which provides a construction of one-versus-the-rest (OvR) multiclass classifiers. In this experiment, the number of instances in the training set was 45 and the number of instances in the test set was 105. The results show that Algorithms 2.3.1 and 2.3.2 performed better than Pegasos with respect to their scores for the training and test sets. In addition, the computational time of Algorithm 2.3.2 was shorter than those of Pegasos and Algorithm 2.3.1. In this case, Algorithm 2.3.2 learned a classifier whose classification

[*2] `https://github.com/LasseRegin/SVM-w-SMO`

Table 2.4: Iris (multiclass; Algorithms 2.3.1, 2.3.2, and 2.4.1 are used as solvers for the subproblem appearing in this multiclass classification experiment.)

Algorithm	Time [sec]	Score (Training)	Score (Test)	Avg. Objective
Pegasos	0.59731907	0.77777778	0.79047619	0.98524879
Algorithm 2.3.1	1.30901892	0.77777778	0.80952381	0.97505393
Algorithm 2.3.2	0.08996129	0.80000000	0.81904762	0.95314453
SMO Algorithm	0.05340354	0.80000000	0.82857143	–

score is similar to the one of the SMO algorithm in almost same running time.

To compare the algorithms in detail, we conducted experiments on other datasets: australian, horse-colic, breast-cancer-wisconsin, census-income, internet-advertisements, MNIST, RANDOM1 and RANDOM2. We performed a stratified five-fold cross-validation with the `sklearn.model_selection.StratifiedKFold` class. Table 2.5 shows the averages of the computational times for learning, the classification scores on the test sets, and the values of the objective function for each dataset. TLE (time limit exceeded) in the table means that the experiment was compulsorily terminated because the running time of the SMO algorithm excessively exceeded those of the other algorithms. The classification scores are calculated using the following formula implemented as the `sklearn.base.ClassifierMixin.score` method,

$$(\text{Score}) := \frac{(\#\text{Accurate Instances})}{(\#\text{Instances})}.$$

This value is an increasing evaluation of goodness of fit [73, Section 4].

Let us evaluate the computational times for learning, the classification scores on the test sets, and the values of the objective function in order. For a detailed, fair, statistical comparison, we used an analysis of variance (ANOVA) test and Tukey–Kramer's honestly significant difference (HSD) test. We used the `scipy.stats.f_oneway` method in the SciPy library as the implementation of the ANOVA tests and the `statsmodels.stats.multicomp.pairwise_tukeyhsd` method in the StatsModels package as the implementation of Tukey–Kramer's HSD test. The ANOVA test examines whether the hypothesis that the given groups have the same population mean is rejected or not. Therefore, we can use it for finding an experimental result that has a significant difference. Tukey–Kramer's HSD test can be used to find specifically which pair has a significant difference in groups. We set 0.05 (5%) as the significance level for the ANOVA and Tukey–Kramer's HSD tests and used the results of each fold of the cross-validation for the statistical evaluations described below.

First, we consider the computation times for learning. All p-values computed by the ANOVA tests were much less than 0.05; this range was from 10^{-26} to 10^{-8}. This implies that a significant difference exists in terms of the computation time between

Table 2.5: Averages of computational times for learning, classification scores on the test sets, and values of the objective function for each dataset

Algorithm	Australian Time [sec]	Score	Objective	Horse-colic Time [sec]	Score	Objective
Pegasos	2.42488674	0.82920957	0.99754775	1.54011672	0.71261261	0.99908892
Algorithm 2.3.1	6.04898934	0.84939531	0.99070582	3.97349780	0.70713213	0.99907467
Algorithm 2.3.2	0.06925168	0.85227300	0.95032527	0.05841917	0.72620120	0.96485216
SMO Algorithm	1.63863547	0.86238696	–	4.96065068	0.71193694	–

Algorithm	Breast-cancer-wisconsin Time [sec]	Score	Objective	Census-income Time [sec]	Score	Objective
Pegasos	2.49116932	0.96843767	0.99228738	180.51582360	0.70191638	0.99995947
Algorithm 2.3.1	6.12068390	0.96558053	0.95931909	456.40569000	0.71029029	0.99909114
Algorithm 2.3.2	0.06906789	0.96558053	0.82970374	0.52751211	0.71031078	0.96254982
SMO Algorithm	0.98688517	0.96989708	–	TLE	–	–

Algorithm	Internet-advertisements Time [sec]	Score	Objective	MNIST Time [sec]	Score	Objective
Pegasos	17.90985671	0.95730497	0.99954933	69.15901990	0.98687356	0.99894996
Algorithm 2.3.1	44.18707687	0.59436744	0.99972300	153.40252700	0.99032472	0.99827819
Algorithm 2.3.2	0.30783534	0.95517036	0.87879677	0.87765352	0.99269253	0.60622818
SMO Algorithm	TLE	–	–	TLE	–	–

Algorithm	RANDOM1 Time [sec]	Score	Objective	RANDOM2 Time [sec]	Score	Objective
Pegasos	0.34232559	0.88000000	0.99113491	3.99642010	0.84699855	0.99941910
Algorithm 2.3.1	1.38527165	0.86000000	0.99789224	17.21619467	0.68499385	0.99982958
Algorithm 2.3.2	0.03952229	0.90000000	0.93026020	0.04702928	0.87099875	0.95740116
SMO Algorithm	0.09197520	0.84000000	–	16.82656507	0.79700312	–

the algorithms for every dataset. The results of the Tukey–Kramer's HSD tests showed that the computation times of Algorithm 2.3.2 for the australian, horse-colic, breast-cancer-wisconsin, census-income, internet-advertisements, and MNIST datasets were significantly shorter than those of Pegasos, Algorithm 2.3.1 and the SMO algorithm. However, the null hypotheses about Algorithm 2.3.2 and the SMO algorithm for the RANDOM1 dataset, and Algorithm 2.3.2 and Pegasos for the RANDOM2 dataset were not rejected. Therefore, for most of the practical datasets, Algorithm 2.3.2 runs significantly faster than the existing algorithms. However, it seems that there are a few cases where the computation time of the Algorithm 2.3.2 roughly equals those of the existing algorithms.

Next, we consider the classification scores on the test sets. The ANOVA tests indicate that significant differences may exist in the census-income, internet-advertisements, and RANDOM2 datasets. However, Tukey-Kramer's HSD test could not reject the null hypotheses between any two algorithms for the census-income dataset. The results of the Tukey-Kramer's HSD tests showed that the scores of Algorithm 2.3.1 were significantly worse than those of the other algorithms for the internet-advertisements and RANDOM2 datasets. Moreover, they showed that the scores of Algorithm 2.3.2 were significantly better than those of Algorithm 2.3.1 and the SMO algorithm for the RANDOM2 dataset. Although each algorithm may have advantages or disadvantages compared with the others on certain datasets, the classification scores of the four algorithms were roughly similar as a whole.

Next, we consider the values of the objective function. All p-values computed by the ANOVA tests were much less than 0.05; this range was from 10^{-32} to 10^{-12}. This implies that a significant difference exists in terms of the values of the objective function between the algorithms for every dataset. The results of the Tukey-Kramer's HSD tests showed that the values of the objective function of Algorithm 2.3.2 were significantly lower than those of Pegasos and Algorithm 2.3.1 for all datasets. Therefore, Algorithm 2.3.1 reduced the value of objective function more than the other algorithms.

Figure 2.2 illustrates a box-plot comparison of Pegasos, Algorithm 2.3.1, Algorithm 2.3.2, and the SMO Algorithm in terms of classification scores on the test sets. We used the results of all folds of the cross-validations and all datasets shown in Table 5 for making this box-plot comparison. The horizontal lines in the boxes represent the median scores, and the boxes represent the upper and lower quartiles of the resulting scores. Similar to the above discussion of the average, we find that Algorithm 2 has the best median of the classification scores among the four algorithms. The results of Pegasos were similar to those of Algorithm 2.3.2; however, the computation time of Algorithm 2.3.2 was dramatically shorter than that of Pegasos. Therefore, box-plot comparison also shows that Algorithm 2.3.2 is the most useful algorithm for learning with a support vector machine.

In conclusion, the above comparison indicates that, whichever algorithm we use, we can obtain classifiers whose classification abilities are similar. However, Algorithm 2.3.2 runs faster than the other algorithms, and it reduces the value of the objective function more. Therefore, the series of experiments and considerations lead

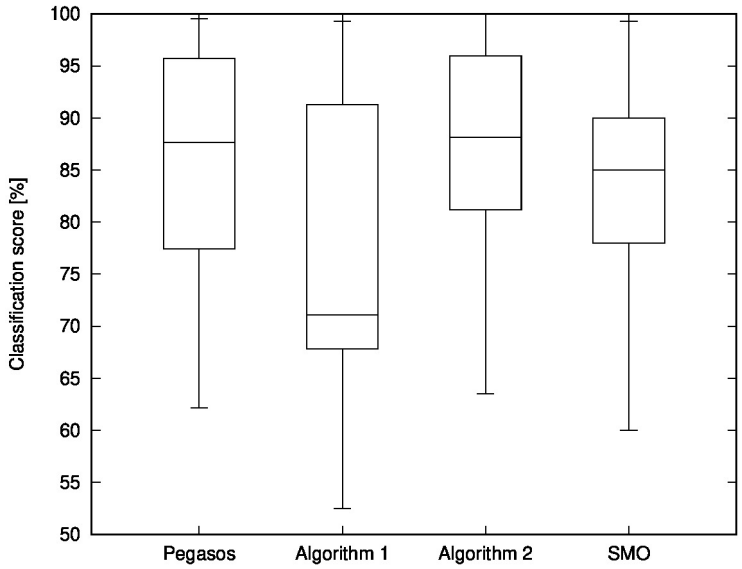


Fig. 2.2: Box-plot comparison of Pegasos, Algorithm 2.3.1, Algorithm 2.3.2 and SMO Algorithm in terms of classification scores on the test sets

us to conclude that Algorithm 2.3.2 is useful for learning with a support vector machine.

2.4.3 Application to learning multilayer neural networks

Let us consider using the proposed algorithms to learn a multilayer neural network with. Our algorithms are not limited to being used for learning support vector machines; they can also be used for optimizing general functions. Therefore, we can also use them for learning a multilayer neural network. Here, we should note that the incremental subgradient algorithm is a specialization of the stochastic subgradient algorithm, which is a useful algorithm for learning a neural network. Hence, we decided to apply it to a concrete task for learning a multilayer neural network and evaluate its applicability to learning deep neural networks.

We used the MNIST database [58] of handwritten digits for this experiment. The goal is recognizing what Arabic numerals are written on the given images. To achieve this goal, we can use 60,000 examples contained in the training set. Each example is composed of a 28×28 image that expresses a handwritten digit and its corresponding label that is an integer number from zero to nine. For the evaluation and comparison of the learning results, we used a test set containing 10,000 examples formatted in the same way.

We constructed and trained a multilayer neural network shown in Figure 2.3 for learning the MNIST database. We used three Affine layers with two ReLU (Rectified Linear Unit) activation functions and, for the output, a Softmax activation function. We used the cross-entropy error function as the objective function for training the

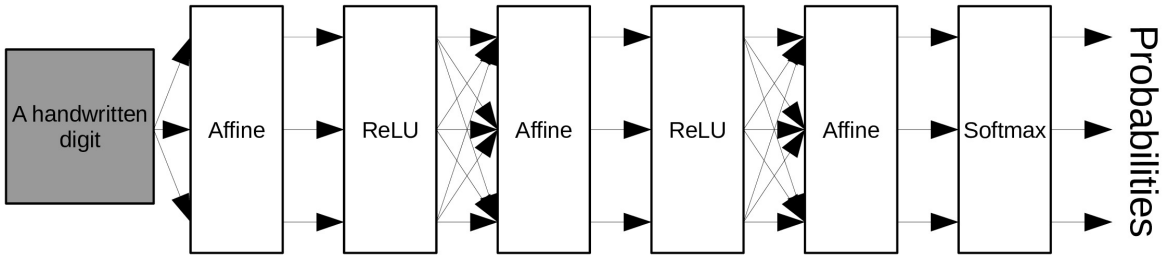


Fig. 2.3: Neural network diagram used to recognizing the MNIST handwritten digits

neural networks.

An Affine layer $A_{W,b}$ transforms a given vector $x \in \mathbb{R}^n$ into

$$A_{W,b}(x) := Wx + b$$

with the parameter $W \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, where n is the number of dimensions of the input vector and m is the number of dimensions of the output vector. The first Affine layer transforms a $784(= 28 \times 28)$-dimensional vector, which expresses a given image, into a 300-dimensional vector. The second Affine layer transforms a 300-dimensional vector into a 100-dimensional vector. The third Affine layer transforms a 100-dimensional vectors into a 10-dimensional vector, which expresses each probability that the given image is the corresponding number. We used the number of dimensions described in [57] for each Affine layer.

The ReLU function transforms each element x_k $(k = 1, 2, \ldots, n)$ of a given vector $x \in \mathbb{R}^n$ into $\max\{0, x_k\}$. The Softmax function transforms a given vector $x := (x_k)_{k=1}^n \in \mathbb{R}^n$ as follows:

$$\mathrm{Softmax}(x) := \frac{1}{\sum_{k=1}^n e^{x_k}} (e^{x_1}, e^{x_2}, \ldots, e^{x_k})^\top,$$

where the number e is the Napier's constant. We define the cross-entropy error $E : \mathbb{R}^n \to \mathbb{R}$, which is used as the objective function for training neural networks, as follows:

$$E(x) := -\sum_{k=0}^{9} y_k \log(x_k),$$

where the vector $x := (x_k)_{k=0}^9 \in \mathbb{R}^{10}$ is the output of the current neural network and y_k $(k = 0, 1, 2, \ldots, 9)$ is one if the label is k and zero otherwise.

In this experiment, we wanted to minimize the cross-entropy error of the training dataset concerning the parameters W_k, b_k for each Affine layer $A_k (k = 1, 2, 3)$. The number of dimensions of the parameters is $784 \times 300 + 300 = 235500$ for the first Affine layer, $300 \times 100 + 100 = 30100$ for the second Affine layer, and $100 \times 10 + 10 = 1010$ for the third Affine layer. Hence, the total number of dimensions of the variables for this minimization problem is $235500 + 30100 + 1010 = 266610$.

We ran Algorithm 2.3.1 with the Discrete Argmin Line Search described in Algorithm 2.3.3 and compared its behaviors when we used a constant learning rate $\lambda_{n,i} := 0.1$, diminishing learning rate $\lambda_{n,i} := (0.1 \times 20)/n$, and learning rates found by the line search in the step-range $[(0.1 \times 20)/(n+100), (0.1 \times 20)/n]$ for the number of iterations $(n = 1, 2, \ldots)$. We set the coefficients for each step-size such that these upper bounds would be equal to each other when the algorithm exits. To use the proposed algorithm, we have to compute the subgradients of the objective function. Here, we used approximations of them worked out by the backpropagation algorithm.

We used the computer described in Subsection 2.1 for these experiments. We wrote the experimental codes in Python 3.6.6 with the NumPy 1.15.4 library. We divided the datasets into 600 mini-batches, each of which contained 100 examples; in other words, we solved the problem to minimize the sum of 600 objective functions. We converted and flattened the handwritten digit images into vectors and divided their elements by 255 for regularization. The parameters for each Affine layer were initialized using a Gaussian distribution of mean zero and variance 0.01.

Figure 2.4 shows the behavior of the values of the objective function for each iteration. The violet line labelled "Constant" shows the result of using the constant

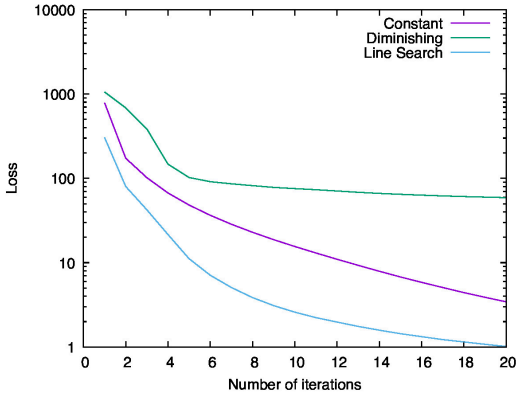
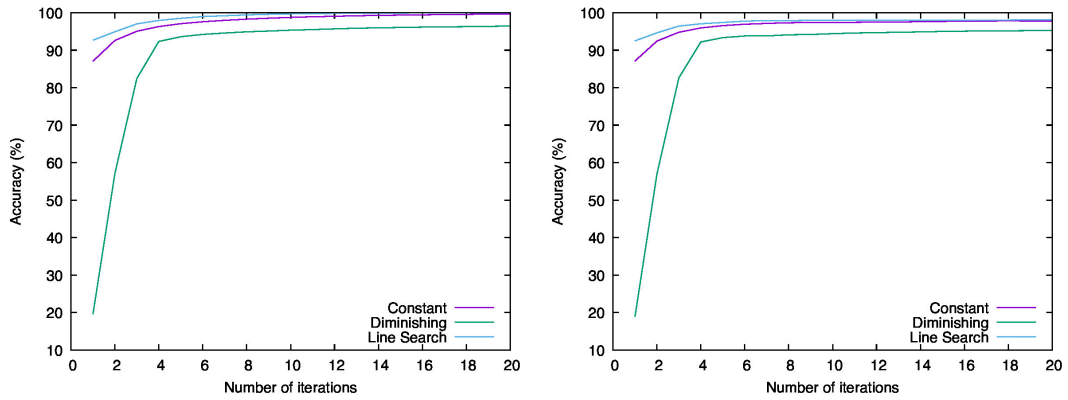


Fig. 2.4: Behavior of the values of the objective function for each iteration

learning rate, while the green line labelled "Diminishing" shows the result of using the diminishing learning rate, and the cyan line labelled "Line Search" shows that of using the learning rate computed with the line search. Overall, we can see that all the results decrease monotonously. This implies that Algorithm 2.3.1 can minimize the objective function with any of the above learning rate settings. The range of reduction of the result by using the diminishing learning rate is less than others. One possible reason is that learning rate becomes too small to minimize the objective function sufficiently. Indeed, from the second to fourth iteration, the result for the diminishing learning rate fell steeply, but this variation became smaller and smaller after the sixth iteration. In contrast to this result, the results for the constant learning rate and the learning rate computed with the line search minimized the objective function continuously and dramatically. In particular, we can see that the line search found the most efficient learning rates of these experiments.

Next, let us examine the classification accuracies. Figure 2.5 shows the behavior



(a) Result for training data

(b) Result for test data

Fig. 2.5: Behavior of the classification accuracies for training and test data

of the classification accuracies for the training and test data. The left-hand graph (Figure 2.5a) shows the classification accuracies for the training data and the right-hand graph (Figure 2.5b) shows those for the test data. The legends of these graphs are the same as in Figure 2.4. We can see that all the results increased, heading for 100%. For both data, the score of "Line Search" is higher than others and the score of "Diminishing" is the lowest. This order is the same as what we saw in Figure 2. Therefore, using the learning rates computed by the line search makes us able to minimize the objective function most and to achieve the best parameters for the neural network to recognize the handwritten digits.

Section 2.5. Conclusion

We proposed novel incremental and parallel subgradient algorithms with a line search that determines suitable learning rates automatically, algorithmically, and appropriately for learning support vector machines. We showed that the algorithms converge to optimal solutions of constrained nonsmooth convex optimization problems appearing in the task of learning support vector machines. Experiments justified the claimed advantages of the proposed algorithms. We compared them with a machine learning algorithm Pegasos, which is designed to learn with a support vector machine efficiently, in terms of prediction accuracy, value of the objective function, and computational time. Regarding the parallel subgradient algorithm in particular, the issue of the computational overhead of the line search can be resolved using multi-core computing. Furthermore, we confirmed that we can apply our incremental subgradient algorithm with the line search to a neural network and they can train it effectively. Overall, our algorithms are useful for efficiently learning a support vector machine and for training a neural network including deep learning.

Chapter 3

Fixed Point Quasiconvex Subgradient Method

Constrained quasiconvex optimization problems appear in many fields, such as economics, engineering, and management science. In particular, fractional programming, which models ratio indicators such as the profit/cost ratio as fractional objective functions, is an important instance. Subgradient methods and their variants are useful ways for solving these problems efficiently. Many complicated constraint sets onto which it is hard to compute the metric projections in a realistic amount of time appear in these applications. This implies that the existing methods cannot be applied to quasiconvex optimization over a complicated set. Meanwhile, thanks to fixed point theory, we can construct a computable nonexpansive mapping whose fixed point set coincides with a complicated constraint set. This chapter proposes an algorithm that uses a computable nonexpansive mapping for solving a constrained quasiconvex optimization problem. We provide convergence analyses for constant and diminishing step-size rules. Numerical comparisons between the proposed algorithm and an existing algorithm show that the proposed algorithm runs stably and quickly even when the running time of the existing algorithm exceeds the time limit.

The contents of this chapter are based on

[25] K. Hishinuma and H. Iiduka. Convergence property, computational performance, and usability of fixed point quasiconvex subgradient method. the 6th Asian Conference on Nonlinear Analysis and Optimization (Oral), 2017;

[27] K. Hishinuma and H. Iiduka. Iterative method for solving constrained quasiconvex optimization problems based on the Krasnosel'skiĭ-Mann fixed point approximation method. RIMS Workshop on Nonlinear Analysis and Convex Analysis (Oral), 2017;

[31] K. Hishinuma and H. Iiduka. Fixed point quasiconvex subgradient method. *European Journal of Operational Research*, 282(2):428–437, 2020.

Section 3.1. Introduction

This chapter considers the constrained quasiconvex optimization problem. This problem is composed of a quasiconvex objective functional and a closed convex constraint set. We call a functional of which any slice is convex a quasiconvex functional, and the class of this functional is a generalization of convex functionals. Quasiconvex functionals inherit some nice properties of convex functionals [35]. However, they do not have all the important properties of convex functionals, such as convexity of the sum of convex functionals, or give a guarantee of the coincidence of local optimality and global optimality. Therefore, the constrained quasiconvex optimization problem is difficult to solve in general.

Fractional programming is an important instance of constrained quasiconvex optimization problems. In economics, there are various situations in which one optimizes ratio indicators, such as the debt/equity ratio (in financial and corporate planning), inventory/sales and output/employee ratios (in production planning), and cost/patient and nurse/patient ratios (in health care and hospital planning) [86]. Under certain conditions, these ratio indicators, fractional objective functionals in other words, have quasiconvexity [53, Lemma 3]. Therefore, these problems can be dealt with as constrained quasiconvex optimizations. Here, we will examine the numerical behaviors of the existing and proposed algorithms when they are applied to the Cobb-Douglas production efficiency problem [7, Problem (3.13)], [33, Problem (6.1)], [86, Section 1.7], which is an instance of a fractional programming and constrained quasiconvex optimization problem. Furthermore, the demand for techniques to solve optimization problems is nowadays not only limited to convex objectives. In particular, optimization problems whose objective functionals are quasiconvex have appeared in economics, engineering, and management science [33, 35]. Therefore, this chapter builds an algorithm that can efficiently solve constrained quasiconvex optimization problems even if they have some complexity.

Subgradient methods with the usual Fenchel subdifferential, an expansion of the gradient for nonsmooth functionals, are useful for solving problems in convex optimization [5, Section 8.2], [41, 42, 43, 45, 56]. We need to use an alternative notion of the usual Fenchel subdifferential since the usual Fenchel subdifferential is defined for a convex functional [8, Subsection 2.1], [53, Subsection 2.2], [81, Proposition 8.12]. Indeed, the usual Fenchel subdifferential may be empty even for a differential nonconvex functional [8, Subsection 2.1]. This implies that we cannot use it directly to solve quasiconvex optimization problems. Fortunately, various, extended subdifferentials for nonconvex functionals have been proposed [74, Section 4], [81, Definition 8.3]. As an instance of them, we can define subdifferentials for quasiconvex or more general functionals by the procedure described in [74, Section 4] to construct them with directional derivatives. These subdifferentials inherit some of the properties, called axioms for subdifferentials [74, Axioms (S_1–S_4)], from the usual Fenchel subdifferential for convex functionals; however, they may not be easily computable. Furthermore, an essential issue is that a local minimizer might not coincide with the global minimizer in quasiconvex optimization. This issue reduces the subdifferential till it contains only

one vector, i.e., the zero vector, meaning that the methods lose any clue as to the direction of the global minimizer. Hence, we cannot ensure the convergence of the generated sequence to the global minimizer of the quasiconvex optimization problem when the usual subgradient methods are used. For the unconstrained quasiconvex optimization problem, Konnov [54] introduced a subgradient method that uses a normalized normal vector to the slice at a current approximation as a subgradient. This idea overcomes the above issue, since there certainly exists a nonzero normal vector to the slice which indicates the direction to the minimizer even if the current approximation is a non-global local minimizer.

Kiwiel [53] proposed a subgradient method that uses a normalized normal vector to the slice as a subgradient (we will call it a subgradient throughout this chapter) for solving the constrained quasiconvex optimization problem. Hu et al. [33] analyzed its convergence properties when inexact subgradients are used and/or when it includes computational errors. Furthermore, a number of subgradient-method variants exist for solving quasiconvex optimization problems, such as the conditional subgradient methods [35] and the stochastic subgradient method [34].

The existing methods assume the computability of the metric projection onto the constraint set, because they use the metric projection to guarantee that the solution is in the constraint set. The metric projection onto the constraint set is defined as a mapping which translates a given point into the nearest point inside the constraint set. Therefore, in general, we have to solve a subproblem of minimizing the distance from a given point subject to the solution being in the constraint set. Certainly, there are some sets onto which the metric projections can be computed easily, such as boxes [2, Proposition 29.15], closed balls [2, Example 3.18 and Proposition 3.19], [82, Section 4], and closed half-spaces [2, Example 29.20]. However, various complicated sets on which computing the metric projections is difficult appear in practical problems [11, 39, 45, 46, 93]. Therefore, we have to develop a new algorithm that can run lightly and quickly even when it is difficult to compute the metric projection onto the constraint set.

On the other hand, if the constraint set can be expressed as a fixed point set of or the intersection of some fixed point sets of nonexpansive mappings, there are algorithms that use these nonexpansive mappings instead of the metric projection for convex optimization [41, 42, 43, 45]. Fixed point sets of nonexpansive mappings have great powers of expression. Any metric projection onto a closed convex set is also a nonexpansive mapping whose fixed point set coincides with these sets [2, Proposition 4.16]. We can build a nonexpansive mapping whose fixed point set coincides with the intersection of the fixed point sets of two or more given nonexpansive mappings [2, Proposition 4.9, 4.47]. Furthermore, there are complicated convex sets called generalized convex feasible sets that are defined by closed convex sets whose intersection may be empty. They can also be expressed using concrete nonexpansive mappings [45, Definition (8)], [93, Definition (50)]. The algorithms listed at the beginning of this paragraph use nonexpansive mappings instead of metric projections onto the constraint sets. Therefore, if these nonexpansive mappings can be more easily computed than the metric projections, it can also be expected that their algorithms will run

more efficiently than algorithms which use metric projections directly.

The existing algorithms for solving convex optimization problems over fixed point sets of nonexpansive mappings are realized by combining a fixed point iterator, which generates a sequence converging to some fixed point of a given nonexpansive mapping, with the existing subgradient methods. The Krasnosel'skiĭ-Mann iterator [55, 63] and Halpern iterator [20] are useful fixed point iterators for finding a fixed point of given nonexpansive mapping. Both generate a sequence converging to some fixed point of a given nonexpansive mapping.

In contrast to the existing literature, this chapter proposes a novel algorithm which minimizes a given quasiconvex functional over the fixed point set of a given nonexpansive mapping. To realize this algorithm, we combine the Krasnosel'skiĭ-Mann iterator [55, 63] with the existing subgradient method [53] for solving quasiconvex optimization problems. The goal of this chapter is to show that our algorithm can solve constrained quasiconvex optimization problems whose constraint set is too complex for the existing algorithms to solve in a realistic amount of time.

This chapter offers three contributions. The first is to provide a widely applicable algorithm for solving constrained quasiconvex optimization problems. The nonexpansive mappings are an extended notion of the metric projection, since the metric projection is also nonexpansive. Therefore, this chapter allows more varied modeling for constrained quasiconvex optimization problems.

The second contribution is to present the theoretical convergence properties of our algorithm. We analyzed the convergence properties for constant and diminishing step-size rules. These results show by how much the error increases when a constant step-size rule is adopted what conditions are required for the generated sequence to converge to the solution of the optimization problem.

The last contribution is to overcome the issue of the existing methods; that is, we show that the proposed algorithm can solve problems whose metric projections onto constraint sets cannot be easily computed. We conduct a numerical comparison of our algorithm and the existing algorithm. The results show that our algorithm can solve actual problems even when the constraint sets are too complex to find the metric projection onto them and when the existing algorithm cannot run in a realistic amount of time.

This chapter is organized as follows. Section 3.2 gives the mathematical preliminaries. Section 3.3 defines our algorithm and presents its convergence analyses. Section 3.4 shows numerical comparisons between the proposed algorithm and the existing subgradient method, by solving a constrained quasiconvex optimization problem named the Cobb-Douglas production efficiency problem. Section 3.5 concludes this chapter.

Section 3.2. Mathematical preliminaries

First, we present the main problem considered in this chapter, i.e., Problem 3.2.1, which is called a *constrained quasiconvex optimization problem.*

Problem 3.2.1. Let H be a real Hilbert space with inner product $\langle\cdot,\cdot\rangle$ and its induced norm $\|\cdot\|$, and let f be a continuous functional on H. In addition, suppose that the functional f has *quasiconvexity*, i.e. $f((1-\alpha)x+\alpha y) \leq \max\{f(x), f(y)\}$ holds for any $x, y \in H$ and for any $\alpha \in [0,1]$. Let X, D be nonempty closed convex subsets of H. Then, we would like to

$$\text{minimize } f(x) \text{ subject to } x \in X \cap D.$$

We define the *set of minima* and the *minimum value* of Problem 3.2.1 by $X^\star := \operatorname{argmin}_{x\in X\cap D} f(x)$ and $f_\star := \inf_{x\in X\cap D} f(x)$, respectively.

We use the following notation in this chapter. $\mathbb{N}$ is the set of natural numbers without zero, and $\mathbb{R}$ is the set of real numbers. $\mathbf{B} := \{x \in H : \|x\| \leq 1\}$ is the unit ball in this Hilbert space, and $\mathbf{S} := \{x \in H : \|x\| = 1\}$ is the unit sphere in that space. Id is the identity mapping of H onto itself. The boundary of a set $C \subset H$ is denoted by $\operatorname{bd} C$, the closure of this set is denoted by $\operatorname{cl} C$, and the diameter of this set is denoted by $\operatorname{diam}(C) := \sup\{\|u-v\| : u, v \in C\}$.

The metric projection onto a closed, convex set $C \subset H$ is denoted by P_C and defined as $P_C(x) \in C$ and $\|x - P_C(x)\| = \inf_{y\in C} \|x-y\|$ for any $x \in H$. For any $\alpha \in \mathbb{R}$, the α-slice of a functional $f : H \to \mathbb{R}$ is denoted by $\operatorname{lev}_{<\alpha} f := \{x \in H : f(x) < \alpha\}$ and the α-trench of a functional $f : H \to \mathbb{R}$ is denoted by $\operatorname{lev}_{\leq\alpha} f := \{x \in H : f(x) \leq \alpha\}$. We call a functional $f : H \to \mathbb{R}$ is coercive if $\lim_{\|x\|\to\infty} f(x) = \infty$ [2, Definition 11.11]. The coerciveness of a functional is equivalent to the boundedness of all trenches of it [2, Proposition 11.12]. The fixed point set of a mapping $T : H \to H$ is denoted by $\operatorname{Fix}(T) := \{x \in H : T(x) = x\}$.

This chapter makes full use of the nonexpansivity of some mapping for analyzing the convergence of the proposed algorithm. Hence, let us define two kinds of nonexpansive condition. A mapping $T : H \to H$ is said to be nonexpansive if $\|T(x) - T(y)\| \leq \|x-y\|$ for any $x, y \in H$, and it is said to be firmly nonexpansive if $\|T(x)-T(y)\|^2 + \|(\mathrm{Id}-T)x - (\mathrm{Id}-T)y\|^2 \leq \|x-y\|^2$ for any $x, y \in H$. Obviously, a firmly nonexpansive mapping is also a nonexpansive mapping [2, Subchapter 4.1]. The properties of these nonexpansivities are described in detail in [2, Chapter 4], [87, Chapter 6].

Useful algorithms for solving Problem 3.2.1 were proposed in [33, 35, 53, 54]. However, they assume that the metric projection onto the set $X \cap D$ can be computed explicitly. Unfortunately, there are many complicated convex sets onto which constructing and/or computing the projection are difficult [11, 45, 46, 93]. This chapter assumes a weaker and expanded condition for the constraint set X, only requiring the existence of a certain nonexpansive mapping expressing this set. Below, we list the conditions assumed throughout in this chapter.

Assumption 3.2.1. We suppose that

(A1) the effective domain $\operatorname{dom}(f) := \{x \in H : f(x) < \infty\}$ coincides with the whole space H;

(A2) there exists some firmly nonexpansive mapping $T : H \to H$ whose fixed point

set $\mathrm{Fix}(T)$ coincides with the constraint set X;

(A3) the constraint set $X = \mathrm{Fix}(T)$ and the feasible set $X \cap D$ are nonempty and there exists at least one minima, i.e. $X^\star \neq \emptyset$.

Assumptions (A2–3) mean that any closed convex sets which can be expressed as a fixed point set of some (firmly) nonexpansive mapping are accepted as constraint sets. Fixed point sets of nonexpansive mappings can express a variety of constraint sets, including not only the sets onto which the metric projections can be calculated such as is used in the existing literature [33, 35, 53], but also complicated sets onto which metric projections cannot be easily calculated [11, 45, 46, 93].

We can construct more complex sets by combining simpler nonexpansive mappings. The following proposition gives the fundamental, variously applicable transformations for building nonexpansive mappings.

Proposition 3.2.1. *Let $T_1, T_2, \ldots, T_N : H \to H$ be nonexpansive mappings (including metric projections onto some convex sets), and suppose that the intersection of these fixed point sets is nonempty. Let $\alpha \in (0, 1/2]$. Then,*

(T1) *the mapping $\sum_{i=1}^N T_i/N$ is also a nonexpansive mapping, and its fixed point set coincides with $\bigcap_{i=1}^N \mathrm{Fix}(T_i)$ [2, Propositions 4.9 and 4.47];*

(T2) *the mapping $\alpha\,\mathrm{Id} + (1-\alpha)T_1$ is a firmly nonexpansive mapping, and its fixed point set coincides with $\mathrm{Fix}(T_1)$ [2, Remark 4.37 and Proposition 4.47].*

The transformation (T1) ensures that we can make a nonexpansive mapping whose fixed point set coincides with the intersection of the fixed point sets of two or more nonexpansive mappings. The transformation (T2) provides us with a way to convert any nonexpansive mapping into a firmly nonexpansive mapping whose fixed point sets correspond with the given one. Our GitHub repository (URL: `https://github.com/iiduka-researches/201811-kaz`) provides these implementations of the transformations (T1 and T2) as higher-order functions `average` and `firm_up` in Python. By using our code, the reader can easily make a nonexpansive mapping expressing his or her desired constraint set.

Furthermore, let us examine an instance of convex sets that can be expressed as fixed point sets of some nonexpansive mappings, called the *generalized convex feasible sets* [45, Definition (10)], [93, Subsection 4.B]. Here, let us consider several closed convex sets $X_i \subset H$ for $i = 0, 1, \ldots, K$, and suppose that the metric projections $\{P_{X_i}\}_{i=0}^K$ onto these convex sets $\{X_i\}_{i=0}^K$ can be easily calculated. If the intersection of these sets is not empty, we can use the transformation and construction procedures described before to make a nonexpansive mapping whose fixed point set coincides with it. Hence, let us consider the opposite case; that is, there is a possibility that the intersection of the sets $\{X_i\}_{i=1}^K$ is empty. Then, we cannot use the straightforward way because the emptiness of the constraint set violates Assumption (A3). To design an alternative constraint set, let us define a functional [45, Definition (8)], [93,

Definition (50)]

$$g(x) := \frac{1}{2}\sum_{i=1}^{K} w_i \left(\min_{y\in X_i} \|x-y\|\right)^2 \quad (x \in H), \tag{3.1}$$

where $\sum_{i=1}^{K} w_i = 1$. This functional g stands for the mean square value from the point x to the sets $\{X_i\}_{i=1}^K$ with respect to the weights $\{w_i\}_{i=1}^K$. Therefore, we can consider the set of points which minimize this functional [45, Definition (10)], [93, Definition (50)],

$$X_g := \left\{ x \in X_0 : g(x) = \min_{y\in X_0} g(y) \right\},$$

as an alternative constraint set in terms of the mean square norm. This set is called the generalized convex feasible set. We can construct a nonexpansive mapping whose fixed point set coincides with this set, and thus, we can deal with the minimization problem over this constraint set by using the algorithm presented later. The way to construct this nonexpansive mapping is described in [45, Definition (9)], [93, Definition (50)].

Various subdifferentials have been proposed for quasiconvex functionals, such as the classical subdifferential [53, Definition (10)], the Greenberg-Pierskalla subdifferential [18, Section 3] and its variants such as the star subdifferential [8, Definition 7], Plastria's lower subdifferential [75, Section 2], and so on [8, Subsection 2.1], [53, Definition (6)–(9)], [74, Section 5]. The Greenberg-Pierskalla subdifferential is one of the most important concepts of subdifferentials for generalized convex functionals because it is a general notion that can be easily handled [74, Section 5]. However, it does not account for the norm of its subgradients and gives only directions. Plastria's lower subdifferential is proposed as another important concept whose properties are closer to those of the usual Fenchel subdifferential for convex functionals [74, Section 5]. In this chapter, conforming to [33, 35, 54], we use the *subdifferential* defined as the normal cones to the slice of the functional f. That is, given a point $x \in H$, we call the set

$$\partial^\star f(x) := \{g \in H : \langle g, y-x\rangle \le 0 \ (y \in \mathrm{lev}_{<f(x)} f)\}$$

the subdifferential of the quasiconvex functional f at a point $x \in H$ [33, Definition 2.3], [35, Definition 2.1], [53, Definition (9)], [54, Section 1]. We also call its element a *subgradient*.

This subdifferential $\partial^\star f(\cdot)$ has some favorable properties, as listed in the following proposition.

Proposition 3.2.2 ([2, Proposition 6.2.(iv)], [53, Lemma 3], [74, Propositions 6 and 8])**.** *Suppose that Assumption 3.2.1 holds, and assume that f is a continuous quasiconvex functional. Then, the following hold.*

(P1) *$\partial^\star f(\cdot)$ coincides with the closure of the Greenberg-Pierskalla subdifferential, i.e., the union of the Greenberg-Pierskalla subdifferential and the singleton set $\{0\}$.*

(P2) *The Greenberg-Pierskalla subdifferential, Plastria's lower subdifferential, and the usual Fenchel subdifferential are contained in the subdifferential* $\partial^\star f(x)$ *for any* $x \in H$.[*1]

(P3) *For any* $x \in H$, *the subdifferential* $\partial^\star f(x)$ *is nonempty, and also contains some nonzero vector.*

(P4) $\partial^\star f(\cdot)$ *is a nonempty closed convex cone.*

First, we defined the subdifferential $\partial^\star f(\cdot)$ as the closure of the Greenberg-Pierskalla subdifferential, such as is shown in Proposition (P1). This implies that the subdifferential $\partial^\star f(\cdot)$ is an extension of the Greenberg-Pierskalla subdifferential, and some properties of this subdifferential can also be used. For example, $\partial^\star f(x)$ coincides with the whole space H if x is a minimizer of f. This proposition ensures that the subdifferential $\partial^\star f(\cdot)$ coincides with the closure of the Greenberg-Pierskalla subdifferential, which is not always closed [33, Subsection 2.1]. Hence, this subdifferential $\partial^\star f(\cdot)$ overcomes the problem of the non-closedness of the Greenberg-Pierskalla subdifferential; it has been used in the recent literature [33, 34, 35]. Furthermore, as shown in Proposition (P2), the subdifferential is also a superset of Plastria's lower subdifferential and the usual Fenchel subdifferential. Since the subdifferential $\partial^\star f(\cdot)$ is a cone as shown in Proposition (P4), this property ensures that every arbitrarily scaled element of the Plastria's lower subdifferential or the usual Fenchel subdifferential can be used as a subgradient in the discussion of this chapter. Proposition (P3) ensures the existence of nonzero subgradients at all points. This fact guarantees that the algorithm described later can always find a subgradient, which is required for the computation. In addition, Proposition (P4) shows that the normalized vector of a subgradient is also a subgradient. Our algorithm implicitly uses this property for choosing a subgradient whose norm is 1.

A subgradient in $\partial^\star f(\cdot)$ is computable when, for example, the functional is formed as a fractional function, a typical instance of a quasiconvex function, with concrete conditions. The following proposition gives the conditions for the quasiconvexity and subgradient computability of fractional functions.

Proposition 3.2.3 ([53, Lemmas 3 (i) and 4])**.** *Let* a *be a convex functional on* H, *and let* b *be a finite, positive functional on* H. *Suppose that* $f(x) := a(x)/b(x)$ *for any* $x \in H$, *the interior of* $\mathrm{dom}(f)$ *is convex, and one of the following conditions holds:*

(i) b *is affine;*
(ii) a *is nonnegative on the interior of* $\mathrm{dom}(f)$ *and* b *is concave;*
(iii) a *is nonpositive on the interior of* $\mathrm{dom}(f)$ *and* b *is convex.*

Then, the functional f *is a quasiconvex functional on the interior of* $\mathrm{dom}(f)$. *Furthermore, the functional* $(a - \alpha b)(\cdot)$ *is convex and* $\partial(a - \alpha b)(x) \subset \partial^\star f(x)$ *for any* $x \in H$, *where* $\alpha := f(x)$ *and* $\partial(a - \alpha b)$ *is the usual Fenchel subdifferential of the functional* $(a - \alpha b)(\cdot)$.

[*1] Furthermore, the other four kinds of subdifferential presented in [74, Section 5] are also contained in the subdifferential $\partial^\star f(x)$. Please refer to [74, Proposition 15] for more details.

The following defines a property named the Hölder condition of a functional. This property is used in turn to describe some of the properties of the quasi-subgradient and to establish the convergence of subgradient methods [35, Section 2].

Definition 3.2.1 (Hölder condition [54, Definition 1])**.** A functional $f : H \to \mathbb{R}$ is said to satisfy the *Hölder condition* with degree $\beta > 0$ at a point $x \in H$ on a set $M \subset H$ if there exists a number $L \in \mathbb{R}$ such that

$$|f(z) - f(x)| \leq L \|z - x\|^{\beta} \quad (z \in M).$$

The Hölder condition with degree 1 is equivalent to Lipschitz continuity. Furthermore, when f is a convex functional, it is also equivalent to the bounded subgradient assumption frequently assumed in convergence analyses of subgradient methods for solving convex optimization problems [35, Section 2]. For more details on this property, see Example 3.3.1 described later.

The following Proposition 3.2.4 is a key lemma which relates the distance to the set of minima to its functional value. While nearly the same assertion in Euclidian spaces is presented in [54, Proposition 2.1], this proposition extends it to Hilbert spaces. We should remark that the condition for a point x is slightly modified from the original one for the later discussion. Nevertheless, we can similarly prove this proposition.

Proposition 3.2.4 ([54, Proposition 2.1])**.** *Suppose that the functional f satisfies the Hölder condition with degree $\beta > 0$ at a point $x^\star \in X^\star$ on the set* $\mathrm{cl}(\mathrm{lev}_{<f(x)} f)$ *for some point $x \in H$ such that $f_\star < f(x)$. Then, we have*

$$f(x) - f_\star \leq L \langle g, x - x^\star \rangle^{\beta} \quad (g \in \partial^\star f(x) \cap \mathbf{S}).$$

Proof. Fix $g \in \partial^\star f(x) \cap \mathbf{S}$ arbitrarily. The continuity and quasiconvexity of f imply its level set $\mathrm{lev}_{<f(x)} f$ is open and convex. This $\mathrm{lev}_{<f(x)} f$ is not an empty set, since it has at least the point $x^\star$. The continuity of f also ensures $\mathrm{bd}(\mathrm{lev}_{<f(x)} f) \neq \emptyset$.

Set $r := \inf\{\|x^\star - u\| : u \in \mathrm{bd}(\mathrm{lev}_{<f(x)} f)\}$. Then, there exists a sequence $\{u_k\} \subset \mathrm{bd}(\mathrm{lev}_{<f(x)} f)$ such that $\|x^\star - u_k\| \leq r + 1/k$ for all $k \in \mathbb{N}$. The openness of $\mathrm{lev}_{<f(x)} f$ implies that it is a distinct set from its boundary, i.e., $f(x) \leq f(u)$ for any $u \in \mathrm{bd}(\mathrm{lev}_{<f(x)} f)$. Hence,

$$\begin{aligned} f(x) - f_\star &\leq f(u_k) - f_\star \\ &\leq L \|u_k - x^\star\|^{\beta} \\ &< L \left(r + \frac{1}{k} \right)^{\beta} \end{aligned}$$

for all $k \in \mathbb{N}$. It follows that

$$f(x) - f_\star \leq L r^{\beta}. \tag{3.2}$$

From the definition of r, the open ball with center $x^\star$ and radius r is contained inside $\mathrm{lev}_{<f(x)} f$. Therefore, $x^\star + (1 - 1/k) r g \in \mathrm{lev}_{<f(x)} f$ holds for any $k \in \mathbb{N}$, and

we have

$$\left(1-\frac{1}{k}\right)r-\langle g,x-x^\star\rangle=\left(1-\frac{1}{k}\right)r\|g\|^2-\langle g,x-x^\star\rangle$$
$$=\left\langle g,x^\star+\left(1-\frac{1}{k}\right)rg-x\right\rangle\leq 0$$

for all $k\in\mathbb{N}$. This implies that $r\leq\langle g,x-x^\star\rangle$. Applying this inequality to inequality (3.2) gives $f(x)-f_\star\leq L\langle g,x-x^\star\rangle^\beta$. This completes the proof. □

The following propositions are used to prove the theorems presented later.

Proposition 3.2.5 ([70, Lemma 1])**.** *Let $\{x_k\}$ be a sequence in the Hilbert space H and suppose that it converges weakly to x. Then for any $y\neq x$, $\liminf_{k\to\infty}\|x_k-x\|<\liminf_{k\to\infty}\|x_k-y\|$.*

Proposition 3.2.6 ([2, Proposition 10.25])**.** *Every quasiconvex continuous functional on a real Hilbert space H has weakly lower semicontinuity. That is to say, we have $f(x)\leq\liminf_{n\to\infty}f(x_n)$ for any sequence $\{x_n\}\subset H$ which converges weakly to a point $x\in H$ if the functional f is a quasiconvex continuous functional on a real Hilbert space H.*

Proposition 3.2.7 ([2, Proposition 11.8])**.** *Let C be a convex subset of H, and let f be a strictly quasiconvex functional on the Hilbert space H, i.e., $f(\alpha x+(1-\alpha)y)<\max\{f(x),f(y)\}$ for any $\alpha\in(0,1)$ and for any two distinct points $x,y\in H$. Then, f has at most one minimizer over C.*

Section 3.3. Quasiconvex subgradient method over a fixed point set

We propose Algorithm 3.3.1 for solving Problem 3.2.1 with Assumption 3.2.1. This

Algorithm 3.3.1 Fixed point quasiconvex subgradient method for solving Problem 3.2.1

Require:
$f\colon H\to\mathbb{R}$, $T\colon H\to H$, $D\subset H$;
$\{v_k\}\subset(0,\infty)$, $\{\alpha_k\}\subset(0,1]$.

Ensure:
$\{x_k\}\subset D$.

1: $x_1\in D$.
2: **for** $k=1,2,\ldots$ **do**
3: $\quad g_k\in\partial^\star f(x_k)\cap\mathbf{S}$.
4: $\quad x_{k+1}:=P_D(\alpha_k x_k+(1-\alpha_k)T(x_k-v_k g_k))$.
5: **end for**

algorithm iteratively generates the next point x_{k+1} from the current approximation x_k in order to improve it. Specifically, step 3 of this algorithm finds a regularized subgradient of the functional f at the current approximation x_k. Step 4 is composed of two improving iterators: one is the subgradient method iterator $x_k - v_k g_k$ to improve approximations with respect to the functional value, and the other is the Krasnosel'skiĭ-Mann iterator [55, 63] $\alpha_k \operatorname{Id} + (1-\alpha_k)T$ to improve approximations with respect to the distance to the fixed point set $\operatorname{Fix}(T)$. To ensure that the generated sequence is contained in the set D, we project each generated point onto this set (this operation is optional because the metric projection operator P_D coincides with the identity mapping Id if the set D is the whole space H). By repeating steps 3–4, this algorithm generates a sequence converging to a point in the solution set $X^\star$.

Assumption 3.2.1 supposes that the effective domain $\operatorname{dom}(f)$ coincides with the whole space H. Nevertheless, we can also apply to this algorithm a functional whose effective domain does not fill H. For example, let us consider a case that the effective domain $\operatorname{dom}(f)$ differs from the whole space H, the set D is contained inside this domain, and the initial point x_1 is an element of this set. Then, we can consider the computation of Algorithm 3.3.1 is limited in the set D, a subset of the effective domain $\operatorname{dom}(f)$. This implies that, with appropriate approximation of f, we can apply a functional whose effective domain does not fill the whole space of this algorithm when the range of the nonexpansive mapping T is contained in this domain. We will illustrate an example of making an approximate function as Example 3.3.3.

Before moving on to the convergence analyses, we will give the assumptions and lemmas describing the fundamental properties of Algorithm 3.3.1.

Assumption 3.3.1. (A4) For any $k \in \mathbb{N}$ such that $f_\star < f(x_k)$ and for all $x^\star \in X^\star$, the functional f satisfies the Hölder condition with degree $\beta > 0$ at the point $x^\star$ on the set $\operatorname{cl}(\operatorname{lev}_{<f(x_k)} f)$.

(A5) The generated sequence $\{x_k\}$ is bounded.

(A6) The real sequence $\{\alpha_k\} \subset (0, 1]$ satisfies $0 < \liminf_{k\to\infty} \alpha_k \leq \limsup_{k\to\infty} \alpha_k < 1$.

In the following, we have to ensure that Assumption (A5), i.e., the boundedness of the sequence generated by Algorithm 3.3.1, holds. If we know the estimated range of the solution candidates, the simplest way to bound the generated sequence is to let the set D be a closed ball with a large enough diameter. We can compute the metric projection onto a closed ball easily [2, Example 3.18]. For example, giving $10^{16}\mathbf{B}$ as the set D to Algorithm 3.3.1 satisfies Assumption (A5).

Even when the boundedness of the set D cannot be guaranteed, we can ensure the boundedness of the generated sequence if the objective functional is coercive. Here, we present a sufficient condition for ensuring the boundedness of the sequence generated by Algorithm 3.3.1 without supposing the boundedness of the set D.

Proposition 3.3.1. *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 1. Suppose that Assumption 3.3.1 holds and there exists a number $k_0 \in \mathbb{N}$ such that $v_k < 1$ for all $k \geq k_0$. Assume that f is coercive. If one of the following holds,*

(i) Assumption (A4) holds,
(ii) the whole space H is an N-dimensional Euclidean space $\mathbb{R}^n$,

then Assumption (A5) is satisfied.

Proof. Let us prove this proposition by dividing it into case (i) and (ii). First, suppose that case (i) holds. We will proceed by way of contradiction and suppose that the sequence $\{x_k\}$ is unbounded. Then, there exists a subsequence $\{x_{k_i}\}$ of the sequence $\{x_k\}$ such that $\lim_{i\to\infty} \|x_{k_i}\| = \infty$. The coercivity of f implies that $\lim_{i\to\infty} f(x_{k_i}) = \infty$. Fix $x^\star \in X^\star$ arbitrarily. Assumption (A4) guarantees that

$$|f(z) - f_\star| \leq L \|z - x^\star\|^\beta$$

for all $z \in \mathrm{cl}(\mathrm{lev}_{<f(x_i)} f)$ and for any $i \in \mathbb{N}$. Since $\lim_{i\to\infty} f(x_{k_i}) = \infty$, we have

$$|f(z) - f_\star| \leq L \|z - x^\star\|^\beta$$

for all $z \in H$. Considering the point z appearing in the above inequality to be limited in the set $x^\star + \mathbf{B}$, we obtain

$$\begin{aligned} f(z) &\leq f_\star + L \|z - x^\star\|^\beta \\ &\leq f_\star + L \end{aligned}$$

for all $z \in x^\star + \mathbf{B}$. Set δ to be the right side of the above inequality, i.e., $\delta := f_\star + L$. Then, the set $x^\star + \mathbf{B}$ is obviously a subset of the bounded trench $\mathrm{lev}_{\leq\delta} f$.

From the assumption of this proposition, there exists a number $k_0 \in \mathbb{N}$ such that $v_k < 1$ for all $k \geq k_0$. For each $k \geq k_0$, let us consider the two separate cases: the case where the point x_k belongs to the trench $\mathrm{lev}_{\leq\delta} f$, and its negation. First, let us consider the positive case; i.e., $x_k \in \mathrm{lev}_{\leq\delta} f$ for $k \geq k_0$. The nonexpansivity of P_D and T and the fact that $x^\star \in \mathrm{Fix}(P_D) \cap \mathrm{Fix}(T)$ ensure that

$$\begin{aligned} \|x_{k+1} - x^\star\| &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - x^\star\| \\ &\leq \|\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k) \|T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k) \|x_k - v_k g_k - x^\star\| \\ &\leq \|x_k - x^\star\| + (1-\alpha_k) v_k \end{aligned}$$

for any $k \geq k_0$ such that $x_k \in \mathrm{lev}_{\leq\delta} f$. Here, both x_k and $x^\star$ belong to the bounded trench $\mathrm{lev}_{\leq\delta} f$, and both $1-\alpha$ and v_k are less than or equal to 1. Hence,

$$\|x_{k+1} - x^\star\| \leq \mathrm{diam}(\mathrm{lev}_{\leq\delta} f) + 1 < \infty$$

holds for any $k \geq k_0$ such that $x_k \in \mathrm{lev}_{\leq\delta} f$. Next, let us consider the negative case; i.e., $x_k \notin \mathrm{lev}_{\leq\delta} f$ for $k \geq k_0$. The nonexpansivity of P_D and T and the fact that

$x^\star \in \mathrm{Fix}(P_D) \cap \mathrm{Fix}(T)$ ensure that

$$\begin{aligned}\|x_{k+1} - x^\star\| &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - x^\star\| \\ &\leq \|\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k)\|T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k)\|x_k - v_k g_k - x^\star\|\end{aligned}$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Let us consider the right term of the right side of the above inequality. Its squared value is bounded from above as follows:

$$\begin{aligned}\|x_k - v_k g_k - x^\star\|^2 &= \|x_k - x^\star\|^2 - 2v_k \langle g_k, x_k - x^\star\rangle + v_k \langle g_k, v_k g_k\rangle \\ &= \|x_k - x^\star\|^2 - v_k \langle g_k, x_k - x^\star\rangle - v_k \langle g_k, x_k - (x^\star + v_k g_k)\rangle\end{aligned}$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Here, $x_k \notin \mathrm{lev}_{\leq \delta} f$ implies that $f_\star \leq f_\star + L = \delta < f(x_k)$ for $f \geq k_0$. Therefore, we have $x^\star \in \mathrm{lev}_{\leq f(x_k)} f$ and $x^\star + v_k g_k \in x^\star + \mathbf{B} \subset \mathrm{lev}_{\leq f(x_k)} f$ for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Since the definition of $g_k \in \partial^\star f(x_k) \cap \mathbf{S}$ together with the preceding discussion implies that $\langle g_k, x_k - x^\star\rangle \geq 0$ and $\langle g_k, x_k - (x^\star + v_k g_k)\rangle \geq 0$, we have

$$\|x_k - v_k g_k - x^\star\| \leq \|x_k - x^\star\|$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Hence, we have

$$\|x_{k+1} - x^\star\| \leq \|x_k - x^\star\|$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. From the results for the both cases where $x_k \in \mathrm{lev}_{\leq \delta} f$ or not, we have

$$\begin{aligned}\|x_k - x^\star\| &\leq \max\{\|x_1 - x^\star\|, \|x_2 - x^\star\|, \ldots, \|x_{k_0} - x^\star\|, \mathrm{diam}(\mathrm{lev}_{\leq \delta} f) + 1\} \\ &< \infty\end{aligned}$$

for all $k \in \mathbb{N}$. However, this contradicts the assumption that the sequence $\{x_k\}$ is unbounded. Therefore, we arrive at a contradiction and the boundedness of the sequence $\{x_k\}$ has been proved under this case.

Now let us suppose that case (ii) holds. In an N-dimensional Euclidean space $\mathbb{R}^N$, every closed, bounded set is compact [87, Problem 3.1.6]. Therefore, the nonempty, compact set $x^\star + \mathbf{B}$ contains a point $\bar{x}$ such that $f(\bar{x}) = \max_{x \in x^\star + \mathbf{B}} f(x)$ [87, Theorem 2.5.7]. Set $\delta := f(\bar{x}) \geq f_\star$. Then, the set $x^\star + \mathbf{B}$ is a subset of the bounded trench $\mathrm{lev}_{\leq \delta} f = \mathrm{lev}_{\leq f(\bar{x})} f$. From the assumption of this proposition, there exists a number $k_0 \in \mathbb{N}$ such that $v_k < 1$ for all $k \geq k_0$. For each $k \geq k_0$, let us consider the two separate cases: the case where the point x_k belongs to the trench $\mathrm{lev}_{\leq \delta} f$, and its negation. First, let us consider the positive case; i.e., $x_k \in \mathrm{lev}_{\leq \delta} f$ for $k \geq k_0$. The

nonexpansivity of P_D and T and the fact that $x^\star \in \mathrm{Fix}(P_D) \cap \mathrm{Fix}(T)$ ensure that

$$\begin{aligned}\|x_{k+1} - x^\star\| &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - x^\star\| \\ &\leq \|\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k)\|T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k)\|x_k - v_k g_k - x^\star\| \\ &\leq \|x_k - x^\star\| + (1-\alpha_k)v_k\end{aligned}$$

for any $k \geq k_0$ such that $x_k \in \mathrm{lev}_{\leq \delta} f$. Here, both x_k and $x^\star$ belong to the bounded trench $\mathrm{lev}_{\leq \delta} f$, and both $1-\alpha$ and v_k are less than or equal to 1. Hence,

$$\|x_{k+1} - x^\star\| \leq \mathrm{diam}(\mathrm{lev}_{\leq \delta} f) + 1 < \infty$$

holds for any $k \geq k_0$ such that $x_k \in \mathrm{lev}_{\leq \delta} f$. Next, let us consider the negative case; i.e., $x_k \notin \mathrm{lev}_{\leq \delta} f$ for $k \geq k_0$. The nonexpansivity of P_D and T and the fact that $x^\star \in \mathrm{Fix}(P_D) \cap \mathrm{Fix}(T)$ ensure that

$$\begin{aligned}\|x_{k+1} - x^\star\| &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - x^\star\| \\ &\leq \|\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k)\|T(x_k - v_k g_k) - x^\star\| \\ &\leq \alpha_k \|x_k - x^\star\| + (1-\alpha_k)\|x_k - v_k g_k - x^\star\|\end{aligned}$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Let us consider the right term of the right side of the above inequality. Its squared value is bounded from above as follows:

$$\begin{aligned}\|x_k - v_k g_k - x^\star\|^2 &= \|x_k - x^\star\|^2 - 2v_k \langle g_k, x_k - x^\star\rangle + v_k \langle g_k, v_k g_k\rangle \\ &= \|x_k - x^\star\|^2 - v_k \langle g_k, x_k - x^\star\rangle - v_k \langle g_k, x_k - (x^\star + v_k g_k)\rangle\end{aligned}$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Here, $x_k \notin \mathrm{lev}_{\leq \delta} f$ implies that $f_\star \leq f_\star + L = \delta < f(x_k)$ for $f \geq k_0$. Therefore, we have $x^\star \in \mathrm{lev}_{\leq f(x_k)} f$ and $x^\star + v_k g_k \in x^\star + \mathbf{B} \subset \mathrm{lev}_{\leq f(x_k)} f$ for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Since the definition of $g_k \in \partial^\star f(x_k) \cap \mathbf{S}$ together with the preceding discussion implies that $\langle g_k, x_k - x^\star\rangle \geq 0$ and $\langle g_k, x_k - (x^\star + v_k g_k)\rangle \geq 0$, we have

$$\|x_k - v_k g_k - x^\star\| \leq \|x_k - x^\star\|$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. Hence, we have

$$\|x_{k+1} - x^\star\| \leq \|x_k - x^\star\|$$

for any $k \geq k_0$ such that $x_k \notin \mathrm{lev}_{\leq \delta} f$. From the results for the both cases where $x_k \in \mathrm{lev}_{\leq \delta} f$ or not, we have

$$\begin{aligned}\|x_k - x^\star\| &\leq \max\{\|x_1 - x^\star\|, \|x_2 - x^\star\|, \ldots, \|x_{k_0} - x^\star\|, \mathrm{diam}(\mathrm{lev}_{\leq \delta} f) + 1\} \\ &< \infty\end{aligned}$$

for all $k \in \mathbb{N}$. This implies that the sequence $\{x_k\}$ is bounded, and this completes the proof. □

The assumption on the step-size, i.e., the existence of a number k_0 such that $v_k < 1$ for all $k \geq k_0$, is satisfied whenever we adopt a diminishing step-size rule which is discussed in Subsection 3.3.2 since the step-size sequence decreasingly converges to zero. Furthermore, it is also satisfied when we adopt a constant step-size rule which is discussed in Subsection 3.3.1 with a small enough constant step-size $v < 1$. Overall, the coerciveness of the objective functional f (and the smallness of the step-sizes) is a sufficient condition for ensuring the boundedness of the generated sequence in the convergence analyses of constant or diminishing step-size rules.

Here let us give some examples which satisfy Assumption 3.3.1. The first example shows the applicability of Algorithm 3.3.1 to constrained convex optimization problems.

Example 3.3.1 ([2, Remark 4.34.(iii), Propositions 4.47 and 16.20])**.** *Suppose that f is a continuous, convex functional on H, $\tilde{T}$ is a nonexpansive mapping of H into itself, and D is a closed, bounded, convex subset of H. Set $T := (\mathrm{Id} + \tilde{T})/2$. Furthermore, assume that the feasible set $\mathrm{Fix}(T) \cap D$ is nonempty. Set $\alpha_k := 1/2$ for every $k \in \mathbb{N}$. If H is finite-dimensional, or if the usual Fenchel subdifferential of f maps every bounded subset of H to a bounded set, then T is a firmly nonexpansive mapping, $\mathrm{Fix}(T)$ coincides with the fixed point set $\mathrm{Fix}(T)$, and Assumption 3.3.1 holds.*

This example shows that, if the set D is bounded, our algorithm can be applied to nonsmooth convex optimization problems over fixed point sets of nonexpansive mappings [41, 42, 43, 45]. The following discussion is predicated upon Assumption (A5), i.e., that the generated sequence is bounded, since it is required for evaluating the distance between the generated sequence and the fixed point set (Lemma 3.3.2). Indeed, the existing analysis of the fixed point subgradient method for convex optimization assumes the generated sequence is bounded [40, Assumption (A2)], [41, Assumption 3.1]. However, we can guarantee the boundedness. As we described above, the simplest D that satisfies the requirements is a ball with a large enough diameter.

The transformation $\tilde{T} \mapsto (\mathrm{Id} + \tilde{T})/2$ converts the given nonexpansive mapping $\tilde{T}$ into a corresponding firmly nonexpansive mapping T whose fixed point set coincides with the given one [2, Remark 4.37 and Proposition 4.47]. This implies that any nonexpansive mapping whose fixed point set coincides with the constraint set can be used as $\tilde{T}$. Of course, since any metric projection operator is a firmly nonexpansive mapping [2, Proposition 4.16], we can solve an optimization problem over the constraint set onto which the metric projection can be computed. In Section 3.4, we will describe a concrete example of constructing a firmly nonexpansive mapping. Furthermore, our algorithm extends the existing subgradient methods for convex optimization, since any of the usual Fenchel subgradients of a convex functional is also a subgradient as defined in this chapter. Hence, the convergence analyses of our algorithm (described later) will be very useful for not only quasiconvex optimization [33, 35, 53, 54] but also convex optimization [41, 42, 43, 45].

Let us consider the simplest example of a (nonconvex) quasiconvex objective functional. The next example shows that a typical quasiconvex functional, called the capped-l_1 norm, appearing in sparse regularization of machine learning tasks [9, Equation (25)], [96, Appendix C.3.1] can be minimized using Algorithm 3.3.1.

Example 3.3.2. *Let* $f(x) := \min\{\|x\|, \alpha\}$ *for some* $\alpha > 0$*, and let* $T := \mathrm{Id}$*. Set* $\{v_k\} \subset (0, \alpha]$ *and* $\alpha_k := 1/2$ *for all* $k \in \mathbb{N}$*. Use* $g_k := x_k / \|x_k\| \in \partial^\star f(x_k) \cap \mathbf{S}$ *for each* $k \in \mathbb{N}$ *until* x_k *reaches the solution. Then, this setting satisfies Assumption 3.3.1.*

Proof. The fixed point set $\mathrm{Fix}(T) = \mathrm{Fix}(\mathrm{Id})$ is obviously the whole space H. Therefore, the minimum value of f is 0 and its minimizer is only the origin. For any $z \in H$, we have

$$\begin{aligned} |f(z) - f(x^\star)| &= \min\{\|z\|, \alpha\} \\ &\leq \|z - x^\star\|. \end{aligned}$$

This implies that f satisfies the Hölder condition with degree 1 at its minimizer $x^\star$ on the whole space H. Expanding $\|x_{k+1}\|^2 = \|x_k - (v_k/2) g_k\|^2$ and using Proposition 3.2.4 with the above result, we have

$$\begin{aligned} \|x_{k+1}\|^2 &= \|x_k\|^2 - v_k \langle g_k, x_k - x^\star \rangle + \left(\frac{v_k}{2}\right)^2 \\ &\leq \|x_k\|^2 - v_k \min\{\|x_k\|, \alpha\} + \left(\frac{v_k}{2}\right)^2 \end{aligned}$$

for all $k \in \mathbb{N}$. When the index $k \in \mathbb{N}$ satisfies $\|x_k\| \leq \alpha$,

$$\begin{aligned} \|x_{k+1}\|^2 &\leq \|x_k\|^2 - v_k \|x_k\| + \left(\frac{v_k}{2}\right)^2 \\ &\leq \frac{5}{4}\alpha^2 \end{aligned}$$

holds. The nonnegativeness of both sides of the above inequality ensures that $\|x_{k+1}\|$ is bounded from above by $\sqrt{5}\alpha/2$ for any $k \in \mathbb{N}$ satisfying $\|x_k\| \leq \alpha$. In the opposite case, i.e. if the index $k \in \mathbb{N}$ satisfies $\alpha < \|x_k\|$,

$$\begin{aligned} \|x_{k+1}\|^2 &\leq \|x_k\|^2 - v_k \alpha + \left(\frac{v_k}{2}\right)^2 \\ &= \|x_k\|^2 - v_k \left(\alpha - \frac{v_k}{4}\right) \end{aligned}$$

holds. Therefore, we have $\|x_{k+1}\| \leq \|x_k\|$ for any $k \in \mathbb{N}$ satisfying $\alpha < \|x_k\|$. Combining the conclusions of both cases, we can see that the sequence $\{\|x_k\|\}$ is bounded from above by $\max\{\|x_1\|, \sqrt{5}\alpha/2\}$. This completes the proof. □

Here we should remark that Assumption 3.3.1 does not guarantee that the sequence generated by Algorithm 3.3.1 converges to some optimum. For example, let us consider the case where $f(x) := \min\{|x|, 1\}$ for any real $x \in \mathbb{R}$ and the initial point

$x_1 := 3/2$. Even though it violates the assumption of Example 3.3.2, let us assume $v_k := 2$ for all $k \in \mathbb{N}$. Then, this setting still satisfies Assumption 3.3.1. However, as illustrated in Figure 3.1, we can see that the generated sequence does not converge to an optimum. In each step, the approximation is moved in the direction of the origin

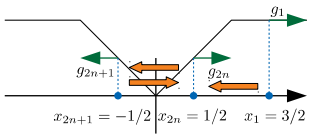


Fig. 3.1: Illustration of case where the generated sequence repeats $1/2$ and $-1/2$ and does not converge to the origin.

by 1. In this counterexample, the first step moves the initial point $3/2$ to the point $1/2$. After that, the algorithm eternally iterates so as to move approximations to their symmetric point with respect to the origin. Therefore, the generated sequence repeats $1/2$ and $-1/2$ and does not converge to the optimum 0. The convergence theorems presented later describe what is required to make the generated sequence converge to the optimum and/or how much error can occur in the solution.

Finally, we present a concrete application that can be dealt with as a quasiconvex optimization problem. The following fractional programming problem is called the Cobb-Douglas production efficiency problem and satisfies Assumption 3.3.1.

Example 3.3.3 ([7, Problem (3.13)], [33, Problem (6.1))], [86, Section 1.7])**.** *Let us consider the problem in an Euclidean space; i.e., suppose that* $H := \mathbb{R}^n$. *Set* $D := [0, M]^n$ *for some* $M > 0$, *and set* $\alpha_k := 1/2$ *for any* $k \in \mathbb{N}$. *We give two positive scalars* $a_0, c_0 > 0$ *and two* n*-dimensional positive vectors in advance,* $a, c \in (0, \infty)^n$ *such that* $\sum_{i=1}^n a_i = 1$. *Let*

$$f(x) := \begin{cases} \dfrac{-a_0 \prod_{j=1}^n x_j^{a_j}}{\langle c, x \rangle + c_0} & (x \in [0, \infty)^n), \\ 0 & (\text{otherwise}), \end{cases}$$

and assume that T *is a firmly nonexpansive mapping from* $\mathbb{R}^n$ *to itself and* $\mathrm{Fix}(T) \cap D \neq \emptyset$ *holds. Run Algorithm 3.3.1 with an initial point* $x_1 \in [0, M]^n$. *Then, Assumption 3.3.1 holds.*

Proof. The generated sequence $\{x_k\}$ is obviously bounded, since it is contained inside $[0, M]^n$. Fix $x^\star \in X^\star$ arbitrarily. The assumptions of this example imply that the feasible set $\mathrm{Fix}(T) \cap D$ has a point $u \in (0, \infty)^n$. Hence, the functional value of this point $f(u)$ is strictly less than $f(v) = 0$ for any $v \in \mathrm{bd}([0, \infty)^n)$. This means that $X^\star \subset (0, \infty)^n$ holds; that is, any optimum $x^\star \in X^\star$ belongs to the interior of $[0, \infty)^n$. Therefore, there exists some $\delta > 0$ such that the closed ball with center $x^\star$ and radius 2δ is contained inside $[0, \infty)^n$. Let us discuss the satisfiability of the Hölder condition on the set $[\delta, \infty)^n$ and on its complement $\overline{[\delta, \infty)^n}$ separately (see Figure 3.2 for the

relation between the point $x^\star$ and these two disjoint sets). First, fix $z \in [\delta, \infty)^n$

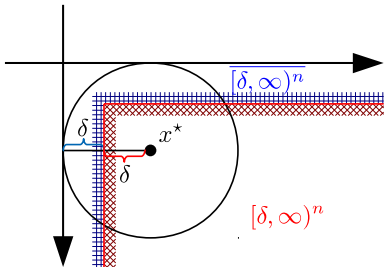


Fig. 3.2: Relation between the point $x^\star$, the set $\overline{[\delta, \infty)^n}$, and its complement

arbitrarily. Denoting the element-wise (Hadamard) division of x and y by $x \oslash y$, the gradient of the functional f can be written as

$$\nabla f(x) = \frac{-a_0 \prod_{j=1}^n x_j^{a_j}}{(\langle c, x\rangle + c_0)^2} c + \frac{-a_0 \prod_{j=1}^n x_j^{a_j}}{\langle c, x\rangle + c_0} (a \oslash x) \quad (x \in [\delta, \infty)^n).$$

Letting $\tilde{c}$ be a vector whose elements are each the minimum of c, an upper bound of the norms of the gradients can be evaluated by using the theorem of arithmetic and geometric means [21, Inequality (2.5.2)], as follows:

$$\begin{aligned}
\|\nabla f(x)\| &\le \frac{a_0 \prod_{j=1}^n x_j^{a_j}}{(\langle c, x\rangle + c_0)^2} \|c\| + \frac{a_0 \prod_{j=1}^n x_j^{a_j}}{\langle c, x\rangle + c_0} \|a \oslash x\| \\
&\le \frac{a_0}{\|\tilde{c}\|} \left(\frac{1}{\langle c, x\rangle + c_0} \|c\| + \|a \oslash x\| \right) \\
&\le \frac{a_0}{\|\tilde{c}\|} \left(\frac{1}{\delta \|\tilde{c}\| + c_0} \|c\| + \frac{\|a\|}{\delta} \right) < \infty \quad (x \in [\delta, \infty)^n).
\end{aligned}$$

This implies that the image $(\|\nabla f(\cdot)\|)([\delta, \infty)^n)$ is bounded. The mean value theorem [68, Inequality (A.55)] ensures the existence of some $\alpha \in (0, 1)$ such that

$$f(z) = f(x^\star) + \langle \nabla f((1-\alpha)x^\star + \alpha z), z - x\rangle .$$

The convexity of $[\delta, \infty)$ and the boundedness of $\|\nabla f(\cdot)\|$ on $[\delta, \infty)^n$ imply that $\|\nabla f((1-\alpha)x^\star + \alpha z)\| \le (a_0/\|\tilde{c}\|)(\|c\|/(\delta\|\tilde{c}\| + c_0) + \|a\|/\delta)$. Therefore, the Cauchy-Schwarz inequality gives us the desired inequality as follows:

$$\begin{aligned}
|f(z) - f(x^\star)| &= |\langle \nabla f((1-\alpha)x^\star + \alpha z), z - x\rangle| \\
&\le \|\nabla f((1-\alpha)x^\star + \alpha z)\| \, \|z - x\| \\
&\le \frac{a_0}{\|\tilde{c}\|} \left(\frac{1}{\delta \|\tilde{c}\| + c_0} \|c\| + \frac{\|a\|}{\delta} \right) \|z - x\| .
\end{aligned}$$

This implies that the functional f satisfies the Hölder condition with degree 1 at the point $x^\star$ on the set $[\delta, \infty)^n$.

Next, fix $\bar{z} \in \overline{[\delta, \infty)^n}$ arbitrarily. The closed ball with center $x^\star$ and radius δ is contained inside $[\delta, \infty)^n$. Hence, $\|x^\star - z\| \geq \delta$ holds. From the definition of f, the maximum value of f is 0 and its range is less than or equal to 0. Therefore,

$$|f(z) - f(x^\star)| \leq \frac{f(x^\star)}{\delta} \|x^\star - z\|$$

holds. Hence, letting $L := \max\{(a_0 / \|\tilde{c}\|)(\|c\| / (\delta \|\tilde{c}\| + c_0) + \|a\| / \delta), f(x^\star/\delta)\}$, it is clear that f satisfies the Hölder condition with degree 1 at the point $x^\star$ on the set $\mathbb{R}^n$. This completes the proof. □

In Section 3.4, we will define the Cobb-Douglas production efficiency problem and show the numerical behavior of Algorithm 3.3.1 when it solves a concrete instance of this problem. Hence, we will put off explaining the problem in detail till later and limit ourselves here to a brief description.

The Cobb-Douglas production efficiency problem was introduced by Bradley and Frey [7]. Hu, Yang, and Sim proposed an algorithm for solving this problem by regarding it as a constrained quasiconvex optimization problem [33]. The fact that the objective functional f is quasiconvex allows us to treat it as a quasiconvex optimization problem. The study by Hu et al. [33] is also based on this fact; it ensures the quasiconvexity of the objective functional. However, the existing results including that of [33] assume that the projection onto the constraint set is easily computable.

Let us consider the meaning of this objective function. The numerator expresses the total profit defined by the production factors x_j for each $j = 1, 2, \ldots, n$. This numerator is modeled with the Cobb-Douglas production function. In this problem, we consider the total cost for the production activities to be an affine function with respect to the production factors $\{x_j\}_{j=1}^n$. This cost function is set as the denominator of the objective function. Hence, the objective function f represents the ratio of the total profit and the total cost. In addition, it is known that the numerator, i.e., the Cobb-Douglas production function, is convex [59, Section 2], and therefore, Proposition 3.2.3 guarantees that f is quasiconvex.

The sequence generated by Algorithm 3.3.1 must be contained in the box $[0, M]^n$, because its complement includes points that make the denominator of the function f zero. Therefore, we set the domain to $D = [0, M]^n \subset [0, \infty)^n$. However, from the definition of f, setting $f(x) := 0$ when x is out of the set $[0, \infty)^n$ makes it possible to expand its domain to the whole space while maintaining continuity. This is an example of an appropriate approximation of the objective function described on page 74.

The following lemmas show the fundamental properties of Algorithm 3.3.1.

Lemma 3.3.1. *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 3.3.1. Suppose that Assumptions 3.2.1 and (A4) hold. Then, for any $k \in \mathbb{N}$ that satisfies $f_\star < f(x_k)$, the following inequality holds.*

$$\|x_{k+1} - x^\star\|^2 \leq \|x_k - x^\star\|^2 - 2v_k(1 - \alpha_k) \left(\frac{f(x_k) - f_\star}{L} \right)^{\frac{1}{\beta}} + (1 - \alpha_k) v_k^2.$$

Proof. Fix $x^\star \in X^\star$ and $k \in \mathbb{N}$ arbitrarily. The convexity of $\|\cdot\|^2$ and the nonexpansivity of P_D and T ensure that

$$\begin{aligned}
\|x_{k+1} - x^\star\|^2 &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - P_D(x^\star)\|^2 \\
&\leq \|\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k) - x^\star\|^2 \\
&\leq \alpha_k \|x_k - x^\star\|^2 + (1-\alpha_k)\|T(x_k - v_k g_k) - T(x^\star)\|^2 \\
&\leq \alpha_k \|x_k - x^\star\|^2 + (1-\alpha_k)\|x_k - x^\star - v_k g_k\|^2 \\
&= \|x_k - x^\star\|^2 - 2v_k(1-\alpha_k)\langle g_k, x_k - x^\star\rangle + (1-\alpha_k)v_k^2. \qquad (3.3)
\end{aligned}$$

On the other hand, Assumption (A4) and Proposition 3.2.4 ensure that

$$\left(\frac{f(x_k) - f_\star}{L}\right)^{\frac{1}{\beta}} \leq \langle g_k, x_k - x^\star\rangle .$$

Applying this inequality to inequality (3.3), we obtain

$$\|x_{k+1} - x^\star\|^2 \leq \|x_k - x^\star\|^2 - 2v_k(1-\alpha_k)\left(\frac{f(x_k) - f_\star}{L}\right)^{\frac{1}{\beta}} + (1-\alpha_k)v_k^2.$$

This completes the proof. □

Similar lemmas are presented in the literature that discuss algorithms which use the metric projection onto the constraint set [34, Lemma 3.1], [35, Lemma 3.2], [53, Lemma 6]. This implies that the generated sequence is guaranteed to be contained in the constraint set. Therefore, the proofs of these lemmas use a property which ensures $x_k \in X \cap D$ and $f_\star \leq f(x_k)$. However, the algorithm presented here generates a sequence that may not be in the fixed point set of T, in other words, it may be out of the feasible set. The convergence analyses are carefully divided into cases where $f_\star < f(x_k)$ holds and cases where it does not hold. The above lemma describes that the existing results hold for the proposed algorithm only if the positive case $f_\star < f(x_k)$ holds and even if the containedness of the generated sequence in the fixed point set cannot be guaranteed.

Lemma 3.3.2. *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 3.3.1. Suppose that Assumptions 3.2.1 and (A5) hold and the real sequence $\{v_k\}$ is bounded. Then, for each $x \in \mathrm{Fix}(T) \cap D$, there exists $M_1 \geq 0$ such that*

$$\|x_{k+1} - x\|^2 \leq \|x_k - x\|^2 - (1-\alpha_k)\|x_k - T(x_k - v_k g_k)\|^2 + v_k M_1 \quad (k \in \mathbb{N}).$$

Proof. Fix $x \in \mathrm{Fix}(T) \cap D$ and $k \in \mathbb{N}$ arbitrarily, and set $M_1 := \sup_{k\in\mathbb{N}}(2\,|\langle g_k, x - T(x_k - v_k g_k)\rangle|)$. The boundedness of $\{x_k\}$ and $\{v_k\}$ ensures that there exist $M_2, M_3 \in \mathbb{R}$ such that

$$\|x_j\| \leq M_2, \quad v_j \leq M_3$$

for all $j \in \mathbb{N}$. Therefore,

$$\begin{aligned}\|x - T(x_j - v_j g_j)\| &\leq \|x\| + \|x_j\| + v_j \\ &\leq \|x\| + M_2 + M_3 < \infty\end{aligned}$$

for all $j \in \mathbb{N}$. The Cauchy-Schwarz inequality, together with this boundedness of the real sequence $\{\|x - T(x_k - v_k g_k)\|\}$, indicates that $M_1 \leq \|x\| + M_2 + M_3 < \infty$.

From the convexity of $\|\cdot\|^2$, we have

$$\begin{aligned}\|x_{k+1} - x\|^2 &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - P_D(x)\|^2 \\ &\leq \|\alpha_k(x_k - x) + (1-\alpha_k)(T(x_k - v_k g_k) - x)\|^2 \\ &\leq \alpha_k \|x_k - x\|^2 + (1-\alpha_k)\|T(x_k - v_k g_k) - x\|^2 . \qquad (3.4)\end{aligned}$$

Let us consider the term $\|T(x_k - v_k g_k) - x\|^2$. Using the firm nonexpansivity of T, we expand this term into

$$\begin{aligned}&\|T(x_k - v_k g_k) - x\|^2 \\ &\leq \|x_k - v_k g_k - x\|^2 - \|(\mathrm{Id} - T)(x_k - v_k g_k) - (\mathrm{Id} - T)(x)\|^2 \\ &= \|x_k - x\|^2 - 2v_k \langle g_k, x_k - x\rangle + v_k^2 \\ &\quad - \|x_k - T(x_k - v_k g_k)\|^2 + 2v_k \langle g_k, x_k - T(x_k - v_k g_k)\rangle - v_k^2 \\ &= \|x_k - x\|^2 - \|x_k - T(x_k - v_k g_k)\|^2 + 2v_k \langle g_k, x - T(x_k - v_k g_k)\rangle .\end{aligned}$$

In view of the definition of M_1, the set $\{2\langle g_k, x - T(x_k - v_k g_k)\rangle : k \in \mathbb{N}\}$ is bounded from above by it. Therefore, we obtain

$$\|T(x_k - v_k g_k) - x\|^2 \leq \|x_k - x\|^2 - \|x_k - T(x_k - v_k g_k)\|^2 + v_k M_1.$$

Applying this inequality to inequality (3.4) yields the desired inequality:

$$\begin{aligned}&\|x_{k+1} - x\|^2 \\ &\leq \alpha_k \|x_k - x\|^2 + (1-\alpha_k)\left(\|x_k - x\|^2 - \|x_k - T(x_k - v_k g_k)\|^2 + v_k M_1\right) \\ &\leq \|x_k - x\|^2 - (1-\alpha_k)\|x_k - T(x_k - v_k g_k)\|^2 + v_k M_1\end{aligned}$$

This completes the proof. □

3.3.1 Constant step-size rule

The following theorem shows how precise the generated solution is when the constant step-size rule is used.

Theorem 3.3.1. *Let $v > 0$ and $v_k := v$ for all $k \in \mathbb{N}$ and $\{x_k\} \subset H$ be a sequence generated by Algorithm 3.3.1. Suppose that Assumptions 3.2.1 and 3.3.1 hold. Then, the sequence $\{x_k\}$ satisfies*

$$\liminf_{k\to\infty} f(x_k) \leq f_\star + L\left(\frac{v}{2}\right)^\beta, \text{ and } \liminf_{k\to\infty} \|x_k - T(x_k)\|^2 \leq Mv$$

for some $M \geq 0$.

Proof. We prove each inequality in order. First, we consider whether the inequality

$$\liminf_{k\to\infty} f(x_k) \leq f_\star + L\left(\frac{v}{2}\right)^\beta \tag{3.5}$$

holds. We will proceed by way of contradiction. Suppose that the inequality does not hold; i.e.,

$$f_\star + L\left(\frac{v}{2}\right)^\beta < \liminf_{k\to\infty} f(x_k).$$

The left-hand side of this inequality is strictly less than the right-hand side. Hence, with the positivity of L, we can choose a positive δ_1 such that

$$f_\star + L\left(\frac{v}{2} + \delta_1\right)^\beta < \liminf_{k\to\infty} f(x_k)$$

holds. The property of the limit inferior guarantees that there exists $k_0 \in \mathbb{N}$ such that

$$f_\star + L\left(\frac{v}{2} + \delta_1\right)^\beta < f(x_k) \quad (k \geq k_0). \tag{3.6}$$

Obviously, this implies that $f_\star < f(x_k)$ for any $k \geq k_0$. Therefore, all assumptions of Lemma 3.3.1 are satisfied when $k \geq k_0$, and the following inequality holds for some $x^\star \in X^\star \neq \emptyset$:

$$\begin{aligned}
&\|x_{k+1} - x^\star\|^2 \\
&\leq \|x_k - x^\star\|^2 - 2v(1-\alpha_k)\left(\frac{f(x_k) - f_\star}{L}\right)^{\frac{1}{\beta}} + (1-\alpha_k)v^2
\end{aligned}$$

for all $k \geq k_0$. Applying inequality (3.6) to the above inequality, we have

$$\begin{aligned}
\|x_{k+1} - x^\star\|^2 &\leq \|x_k - x^\star\|^2 - 2v\delta_1(1-\alpha_k) \\
&\leq \|x_{k_0} - x^\star\|^2 - 2v\delta_1 \sum_{j=k_0}^{k} (1-\alpha_k)
\end{aligned} \tag{3.7}$$

for all $k \geq k_0$. From Assumption (A6), $\limsup_{k\to\infty} \alpha_k < 1$ holds. Hence, a starting index $k_1 \in \mathbb{N}$ greater than k_0 exists such that the subsequence $\{\alpha_k\}_{k\geq k_1}$ is bounded

above by some positive real that is strictly less than 1. This means that inequality (3.7) does not hold for large enough $k \geq k_1$, and we have arrived at a contradiction. Therefore, inequality (3.5) holds.

Next, let us prove the remaining part of this theorem, in other words, show that the inequality

$$\liminf_{k\to\infty} \|x_k - T(x_k)\|^2 \leq Mv$$

holds for some positive real $M > 0$. Fix $x \in \mathrm{Fix}(T)$ arbitrarily. Lemma 3.3.2 guarantees the existence of a nonnegative real $M_1 \geq 0$ such that

$$\|x_{k+1} - x\|^2 \leq \|x_k - x\|^2 - (1-\alpha_k)\|x_k - T(x_k - vg_k)\|^2 + vM_1 \tag{3.8}$$

for all $k \in \mathbb{N}$. In view of Assumption (A6), $\liminf_{k\to\infty}(1-\alpha_k)$ is positive; i.e., it is not equal to zero. We will again proceed by way of contradiction and suppose that

$$\liminf_{k\to\infty} \|x_k - T(x_k - vg_k)\|^2 \leq \frac{2vM_1}{\liminf_{k\to\infty}(1-\alpha_k)} \tag{3.9}$$

does not hold and

$$\frac{2vM_1}{\liminf_{k\to\infty}(1-\alpha_k)} < \liminf_{k\to\infty} \|x_k - T(x_k - vg_k)\|^2$$

holds. In the same ways choosing δ_1 in the first part of this proof, we can find a positive $\delta_2 > 0$ that satisfies

$$\frac{2vM_1}{\liminf_{k\to\infty}(1-\alpha_k)} + \delta_2 < \liminf_{k\to\infty} \|x_k - T(x_k - vg_k)\|^2 .$$

The property of the limit inferior guarantees that a positive number $k_2 \in \mathbb{N}$ exists such that

$$\frac{2vM_1}{\liminf_{k\to\infty}(1-\alpha_k)} + \delta_2 < \|x_k - T(x_k - vg_k)\|^2$$

for all $k \geq k_2$. Applying the above inequality to inequality (3.8), we obtain

$$\begin{aligned}
&\|x_{k+1} - x\|^2 \\
&\leq \|x_k - x\|^2 - (1-\alpha_k)\left(\frac{2vM_1}{\liminf_{k\to\infty}(1-\alpha_k)} + \delta_2\right) + vM_1
\end{aligned}$$

for all $k \geq k_2$. The fundamental property of the limit inferior also ensures the existence of a number $k_3 \in \mathbb{N}$ larger than k_2 such that $\liminf_{k\to\infty}(1-\alpha_k)/2 < (1-\alpha_k)$ for any $k \geq k_3$. Therefore, we have

$$\begin{aligned}
\|x_{k+1} - x\|^2 &\leq \|x_k - x\|^2 - \frac{\delta_2}{2}\liminf_{k\to\infty}(1-\alpha_k) \\
&\leq \|x_{k_3} - x\|^2 - \frac{\delta_2}{2}(k - k_3 + 1)\liminf_{k\to\infty}(1-\alpha_k)
\end{aligned}$$

for all $k \geq k_3$. Since the above inequality does not hold for large enough $k \geq k_3$, we have arrived at a contradiction. Therefore, inequality (3.9) holds.

With this inequality, let us evaluate the squared distance between an element of the generated sequence and its transformed point by the nonexpansive mapping T. Using the triangle inequality and the nonexpansivity of T, we have

$$\begin{aligned}\|x_k - T(x_k)\|^2 &\leq (\|x_k - T(x_k - vg_k)\| + \|T(x_k - vg_k) - T(x_k)\|)^2 \\ &\leq (\|x_k - T(x_k - vg_k)\| + v)^2 \\ &= \|x_k - T(x_k - vg_k)\|^2 + 2v\|x_k - T(x_k - vg_k)\| + v^2\end{aligned}$$

for any $k \in \mathbb{N}$. Now, since x is a fixed point of the mapping T, we expand the second term of the above expression as $\|x_k - T(x_k - vg_k)\| \leq 2\|x_k - x\| + v$ for any $k \in \mathbb{N}$. Furthermore, Assumption (A5) ensures the existence of a positive real $M_2 > 0$ bounding the set $\{\|x_k - x\| : k \in \mathbb{N}\}$ from above. Hence, we finally obtain

$$\|x_k - T(x_k)\|^2 \leq \|x_k - T(x_k - vg_k)\|^2 + 4v(M_2 + v)$$

for any $k \in \mathbb{N}$. Taking the limit inferior of both sides of the above inequality yields

$$\liminf_{k\to\infty} \|x_k - T(x_k)\|^2 \leq \liminf_{k\to\infty} \|x_k - T(x_k - vg_k)\|^2 + 4v(M_2 + v).$$

Set $M := 2(M_1/\liminf_{k\to\infty}(1 - \alpha_k) + 2(M_2 + v)) \in \mathbb{R}$. Applying inequality (3.9) to the above, we obtain the desired inequality as follows:

$$\begin{aligned}\liminf_{k\to\infty} \|x_k - T(x_k)\|^2 &\leq 2\left(\frac{M_1}{\liminf_{k\to\infty}(1 - \alpha_k)} + 2(M_2 + v)\right)v \\ &\leq Mv.\end{aligned}$$

This completes the proof. □

3.3.2 Diminishing step-size rule

Finally, we prove the weak convergence theorem of Algorithm 3.3.1. To let the generated sequence weakly converge to some optimum, we can use a specific step-size called a diminishing step-size. The following theorem describes this condition and shows that the generated sequence weakly converges.

Theorem 3.3.2. *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 3.3.1. Suppose that*

(i) Assumptions 3.2.1 and 3.3.1 hold,
(ii) and the real sequence $\{v_k\} \subset (0, \infty)$ satisfies

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

Then, there exists a subsequence of the generated sequence $\{x_k\}$ which converges weakly to a point in $X^\star$. In addition, if

(iii) the whole space H is an N-dimensional Euclidean space $\mathbb{R}^N$,
(iv) and the solution $x^\star \in X^\star$ is unique,

then, the whole sequence $\{x_k\}$ converges to this unique solution $x^\star$.

Assumption (A3) and Proposition 3.2.7 show that the strict quasiconvexity of the objective function is a sufficient condition for the uniqueness of the solution $x^\star \in X^\star$.

Before proving the above theorem, we prove the following lemma which will be needed later.

Lemma 3.3.3. *Suppose that Assumptions 3.2.1 and 3.3.1 hold, and suppose that the real sequence $\{v_k\} \subset (0, \infty)$ satisfies*

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

Let $\{x_k\} \subset H$ be the sequence generated by Algorithm 3.3.1 with this real sequence $\{v_k\}$. Then,

$$\liminf_{k\to\infty} f(x_k) \leq f_\star$$

holds.

Proof. We will proceed by way of contradiction and suppose that the conclusion $\liminf_{k\to\infty} f(x_k) \leq f_\star$ does not hold, that is, $\liminf_{k\to\infty} f(x_k) > f_\star$. There exists a positive number $\delta > 0$ and an index $k_0 \in \mathbb{N}$ such that $f_\star + \delta < f(x_k)$ for all $k \geq k_0$. Furthermore, the assumption that the real sequence $\{v_k\}$ converges to zero guarantees the existence of an index $k_1 \geq k_0$ such that $v_k < (\delta/L)^{1/\beta}$ for all $k \geq k_1$. Applying these two inequalities to Lemma 3.3.1, we have

$$\begin{aligned}
\|x_{k+1} - x^\star\|^2 &\leq \|x_k - x^\star\|^2 - 2v_k(1-\alpha_k)\left(\frac{f(x_k) - f_\star}{L}\right)^{\frac{1}{\beta}} + (1-\alpha_k)v_k^2 \\
&< \|x_k - x^\star\|^2 - v_k(1-\alpha_k)\left(\frac{\delta}{L}\right)^{\frac{1}{\beta}} \\
&< \|x_{k_1} - x^\star\|^2 - \left(\frac{\delta}{L}\right)^{\frac{1}{\beta}} \sum_{n=k_1}^{k} v_k(1-\alpha_k) \quad (k \geq k_1).
\end{aligned}$$

Assumption (A6) guarantees the existence of a positive number $\underline{\alpha}$ and an index $k_2 \geq k_1$ such that $\underline{\alpha} < 1 - \alpha_k$ for any $k \geq k_2$. This implies that the above inequality does not hold for a sufficiently large $k \geq k_2$; hence, we arrive at a contradiction. This completes the proof. □

Now let us prove Theorem 3.3.2 with the above result.

Proof of Theorem 3.3.2. Let the limit superior of the real sequence $\{\alpha_k\}$ be denoted by $\bar{\alpha} \in (0,1)$. Fix $x^\star \in X^\star$ arbitrarily. We will prove the assertion by separating the problem into two cases: the case where there exists a number $k_0 \in \mathbb{N}$ such that $\|x_{k+1} - x^\star\| \leq \|x_k - x^\star\|$ for all $k \geq k_0$, and its negation.

First, let us consider the positive case; i.e., there exists a number $k_0 \in \mathbb{N}$ such that $\|x_{k+1} - x^\star\| \leq \|x_k - x^\star\|$ for all $k \geq k_0$. The property of the limit superior guarantees the existence of a number $k_1 \geq k_0$ such that $\alpha_k < \bar{\alpha} + (1-\bar{\alpha})/2$ for all $k \geq k_1$. Therefore, applying this relationship between $\bar{\alpha}$ and α_k to Lemma 3.3.2, an estimate of $\|x_k - T(x_k - v_k g_k)\|$ for any $k \geq k_1$ can be obtained as follows:

$$\begin{aligned}\frac{1}{2}(1-\bar{\alpha})\left\|x_k - T(x_k - v_k g_k)\right\|^2 &\leq (1-\alpha_k)\left\|x_k - T(x_k - v_k g_k)\right\|^2 \\ &\leq \|x_k - x^\star\|^2 - \|x_{k+1} - x^\star\|^2 + v_k M_1. \end{aligned} \tag{3.10}$$

In this case, the monotonicity and boundedness of the subsequence $\{\|x_k - x^\star\|\}_{k \geq k_1}$ are assured. Hence, this subsequence converges to some nonnegative real. Since we have assumed that the real sequence $\{v_k\}$ converges to zero, the left-hand side of inequality (3.10) converges to zero. On the other hand, $\|x_k - T(x_k)\|$ for each $k \in \mathbb{N}$ can be expanded with the triangle inequality and by noting the nonexpansivity of T as follows:

$$\begin{aligned}\|x_k - T(x_k)\| &\leq \|x_k - T(x_k - v_k g_k)\| + \|T(x_k - v_k g_k) - T(x_k)\| \\ &\leq \|x_k - T(x_k - v_k g_k)\| + v_k.\end{aligned}$$

From the assumption of this theorem and the previous discussion, both terms on the right-hand side above converge to zero with respect to k. Hence, we find that the real sequence $\{\|x_k - T(x_k)\|\}$ converges to zero.

The property of the limit inferior of the real sequence $\{f(x_k)\}$ guarantees the existence of a subsequence $\{f(x_{k_i})\}$ converging to $\liminf_{k\to\infty} f(x_k)$. Note that Lemma 3.3.3 asserts that this limit inferior is less than or equal to the minimum value $f_\star$. There exist a point $u \in H$ and a subsequence $\{x_{k_{i_j}}\} \subset \{x_{k_i}\}$ converging weakly to the point u, since $\{x_{k_i}\}$ is a bounded sequence in H. Now, suppose that u is not a fixed point of T. Since the real sequence $\{\|x_k - T(x_k)\|\}$ converges to zero and T is a nonexpansive mapping, Proposition 3.2.5 produces a contradiction as follows:

$$\begin{aligned}\liminf_{j\to\infty}\left\|x_{k_{i_j}} - u\right\| &< \liminf_{j\to\infty}\left\|x_{k_{i_j}} - T(u)\right\| \\ &\leq \liminf_{j\to\infty}\left(\left\|x_{k_{i_j}} - T\left(x_{k_{i_j}}\right)\right\| + \left\|T\left(x_{k_{i_j}}\right) - T(u)\right\|\right) \\ &\leq \liminf_{j\to\infty}\left\|x_{k_{i_j}} - u\right\|.\end{aligned}$$

Therefore, we can see that u is a fixed point of T. Proposition 3.2.6 means that the objective functional f has weakly lower semicontinuity. The weak convergence of the

sequence $\{x_{k_{i_j}}\}$ implies

$$f(u) \leq \liminf_{j\to\infty} f(x_{k_{i_j}}) = \lim_{i\to\infty} f(x_{k_i}) \leq f_\star;$$

that is, u is an optimum.

To deal with the positive case, let us consider the weak convergence of $\{x_k\}$ and its subsequences. Take another subsequence $\{x_{k_{i_l}}\} \subset \{x_{k_i}\}$ that converges weakly to a point $v \in H$. A similar discussion to the one for obtaining $u \in X^\star$ ensures that the point v is also an optimum. To show the uniqueness of the weak accumulation points of the sequence $\{x_{k_i}\}$, let us assume that $u \neq v$. Since the sequence $\{\|x_k - x^\star\|\}$ converges, Proposition 3.2.5 leads us to a contradiction:

$$\begin{aligned}
\lim_{k\to\infty} \|x_k - u\| &= \lim_{j\to\infty} \left\|x_{k_{i_j}} - u\right\| < \lim_{j\to\infty} \left\|x_{k_{i_j}} - v\right\| \\
&= \lim_{j\to\infty} \|x_k - v\| = \lim_{l\to\infty} \left\|x_{k_{i_l}} - v\right\| < \lim_{l\to\infty} \left\|x_{k_{i_l}} - u\right\| \\
&= \lim_{k\to\infty} \|x_k - u\| .
\end{aligned}$$

Hence, we can see that u is the same point as v, and the uniqueness of all weak accumulation points of the sequence $\{x_{k_i}\}$ is proven. This uniqueness implies that the sequence $\{x_{k_i}\}$ converges weakly to $u \in X^\star$. Now take another subsequence $\{x_{k_m}\} \subset \{x_k\}$ that converges weakly to a point $w \in H$, and suppose that u is a different point from v. From the fact that the sequence $\{\|x_k - x^\star\|\}$ converges and from Proposition 3.2.5, we can deduce that

$$\begin{aligned}
\lim_{k\to\infty} \|x_k - u\| &= \lim_{i\to\infty} \|x_{k_i} - u\| < \lim_{i\to\infty} \|x_{k_i} - w\| \\
&= \lim_{j\to\infty} \|x_k - w\| = \lim_{m\to\infty} \|x_{k_m} - w\| < \lim_{m\to\infty} \|x_{k_m} - u\| \\
&= \lim_{k\to\infty} \|x_k - u\| .
\end{aligned}$$

However, this is a contradiction. Hence, the sequence $\{x_k\}$ converges weakly to some optimum. This proves the positive case.

Next, let us consider the negative case, in other words, the case where a subsequence $\{x_{k_i}\} \subset \{x_k\}$ exists that satisfies $\|x_{k_i} - x^\star\| < \|x_{k_i+1} - x^\star\|$ for all $i \in \mathbb{N}$. A similar discussion to the one for finding the number k_1 in the positive case guarantees the existence of $i_0 \in \mathbb{N}$ satisfying $\alpha_{k_i} < \bar{\alpha} + (1 - \bar{\alpha})/2$ for all $i \geq i_0$. The distances from the point $x^\star$ to each point x_{k_i+1} where $i \geq i_0$ from Lemma 3.3.2 are as follows:

$$\begin{aligned}
\|x_{k_i+1} - x\|^2 &\leq \|x_{k_i} - x\|^2 - (1 - \alpha_{k_i}) \|x_{k_i} - T(x_{k_i} - v_{k_i} g_{k_i})\|^2 + v_{k_i} M_1 \\
&\leq \|x_{k_i} - x\|^2 - \frac{1}{2}(1 - \bar{\alpha}) \|x_{k_i} - T(x_{k_i} - v_{k_i} g_{k_i})\|^2 + v_{k_i} M_1 .
\end{aligned}$$

Here, we have assumed that $\|x_{k_i+1} - x^\star\|$ is greater than $\|x_{k_i} - x^\star\|$ for all $i \in \mathbb{N}$ and the real sequence $\{v_k\}$ converges to zero. Thus, the above inequality implies that the

real sequence $\{\|x_{k_i} - T(x_{k_i} - v_{k_i} g_{k_i})\|\}$ converges to zero with respect to i. On the other hand, the distances $\|x_{k_i} - T(x_{k_i})\|$ for each $i \in \mathbb{N}$ can be estimated from the nonexpansivity of T as follows:

$$\begin{aligned}\|x_{k_i} - T(x_{k_i})\| &\leq \|x_{k_i} - T(x_{k_i} - v_{k_i} g_{k_i})\| + \|T(x_{k_i} - v_{k_i} g_{k_i}) - T(x_{k_i})\| \\ &\leq \|x_{k_i} - T(x_{k_i} - v_{k_i} g_{k_i})\| + v_{k_i}.\end{aligned}$$

Since both terms of the right-hand side of the above inequality converge to zero, the left-hand side also converges to zero.

We will proceed by way of contradiction; suppose that $\limsup_{i\to\infty} f(x_{k_i}) > f_\star$. This implies the existence of $\delta > 0$ and a subsequence $\{x_{k_{i_j}}\} \subset \{x_{k_i}\}$ such that $f_\star + \delta < f(x_{k_{i_j}})$ for all $j \in \mathbb{N}$. Since $\|x_{k_i} - x^\star\| < \|x_{k_i+1} - x^\star\|$ and $f_\star < f(x_{k_{i_j}})$ hold for any $j \in \mathbb{N}$, we can use Lemma 3.3.1 to get

$$v_{k_{i_j}} \left(1 - \alpha_{k_{i_j}}\right) \left(2 \left(\frac{\delta}{L}\right)^{\frac{1}{\beta}} - v_{k_{i_j}}\right) < 0$$

for all $j \in \mathbb{N}$. The above inequality does not hold for sufficiently large $j \in \mathbb{N}$, since the real sequence $\{v_k\}$ converges to zero. Therefore, we arrive at a contradiction, and thus, $\limsup_{i\to\infty} f(x_{k_i}) \leq f_\star$.

The boundedness of the sequence $\{x_{k_i}\}$ guarantees the existence of a subsequence $\{x_{k_{i_j}}\} \subset \{x_{k_i}\}$ that weakly converges to some point $u \in H$. To show that u is a fixed point of the mapping T, let us assume that it is not. Recall that the real sequence $\{\|x_{k_i} - T(x_{k_i})\|\}$ converges to zero. Hence, the nonexpansivity of T together with Proposition 3.2.5 produces a contradiction,

$$\begin{aligned}\liminf_{j\to\infty} \left\|x_{k_{i_j}} - u\right\| &< \liminf_{j\to\infty} \left\|x_{k_{i_j}} - T(u)\right\| \\ &\leq \liminf_{j\to\infty} \left(\left\|x_{k_{i_j}} - T\left(x_{k_{i_{j_l}}}\right)\right\| + \left\|T\left(x_{k_{i_j}}\right) - T(u)\right\|\right) \\ &\leq \liminf_{j\to\infty} \left\|x_{k_{i_j}} - u\right\|.\end{aligned}$$

Therefore, we have $u \in \mathrm{Fix}(T)$. In addition, Proposition 3.2.6 means that the objective functional f has weakly lower semicontinuity. Hence,

$$f(u) \leq \liminf_{j\to\infty} f(x_{k_{i_j}}) \leq \limsup_{i\to\infty} f(x_{k_i}) \leq f_\star$$

holds. This implies conclusively that there exists a subsequence of $\{x_k\}$ which weakly converges to the optimum $u \in X^\star$.

Let us prove the additional statement of this theorem. The following proof is played under the assumption that the solution $x^\star \in X^\star$ is unique and the whole space is an N-dimensional Euclidean space, i.e., $H = \mathbb{R}^N$.

The existence of a subsequence $\{x_{k_i}\}$ that weakly converges to a unique solution $x^\star$ has been guaranteed. In Euclidean space, weak convergence coincides with strong

convergence. Therefore, the sequence $\{x_{k_i}\}$ converges to a unique $x^\star$. If some number $k_0 \in \mathbb{N}$ exists such that $\|x_{k+1} - x^\star\| \leq \|x_k - x^\star\|$ for all $k \geq k_0$, the whole sequence $\{x_k\}$ converges to a point in $X^\star$. Let us consider the opposite case. Let $\{k_i\} \subset \mathbb{N}$ be the sequence of all indexes satisfying $\|x_{k_i} - x^\star\| < \|x_{k_i+1} - x^\star\|$ (and $k_i < k_{i+1}$ for any $i \in \mathbb{N}$). According to the assumption, this sequence is infinite. The sequence $\{x_{k_i}\}$ is now bounded, and this implies that it has a subsequence converging to a unique optimum $x^\star \in X^\star$. The above discussion ensures that any converging subsequence of $\{x_{k_i}\}$ converges to a unique optimum $x^\star \in X^\star$. This further implies that $\{x_{k_i}\}$ also converges to this optimum $x^\star \in X^\star$. From the assumed settings, $\|x_{j+1} - x^\star\| \leq \|x_j - x^\star\|$ holds for any index $j \in \mathbb{N}$ that does not belong to the set $\{k_i\}$. The convergence of the sequence $\{x_{k_i}\}$ means that, for any $\epsilon > 0$, there exists an index $\hat{i} \in \mathbb{N}$ such that $\left\|x_{k_{\hat{i}}} - x^\star\right\| < \epsilon$. Furthermore, for any index $k \geq k_{\hat{i}}$ that does not belong to the set $\{k_i\}$, $\left\|x_{k_{\hat{i}}} - x^\star\right\| \leq \|x_{k_i} - x^\star\| < \epsilon$ also holds for $i := \max\{k_i : k_i \leq k\} \geq k_{\hat{i}}$. This implies that the whole sequence $\{x_k\}$ converges to a unique optimum $x^\star \in X^\star$. This completes the proof. □

Section 3.4. Numerical experiments

To confirm that Algorithm 3.3.1 converges to the optimum and evaluate its performance, we ran it and an existing algorithm [53, Algorithm (14)] on a concrete constrained quasiconvex optimization problem, i.e., the Cobb-Douglas production efficiency problem [7, Problem (3.13)], [33, Problem (6.1)], [86, Section 1.7]. Let us redefine Example 3.3.3 with concrete constraints as follows.

Problem 3.4.1 ([7, Problem (3.13)], [33, Problem (6.1)], [86, Section 1.7])**.** Suppose that $H := \mathbb{R}^n$. Let $a_0, c_0 > 0$ and let $a, c \in (0,\infty)^n$ such that $\sum_{i=1}^n a_i = 1$. Furthermore, let $b_i \in [0,\infty)^n$, $\underline{p}_i \in [0,\infty)^n$, and $\overline{p}_i \in (0,\infty]^n$ for $i = 1, 2, \ldots, m$. Then, we would like to

$$\begin{aligned} \text{minimize } f(x) &:= \begin{cases} \frac{-a_0 \prod_{j=1}^n x_j^{a_j}}{\langle c, x\rangle + c_0} & (x \in [0,\infty)^n), \\ 0 & (\text{otherwise}), \end{cases} \\ \text{subject to } \underline{p}_i &\leq \langle b_i, x\rangle \leq \overline{p}_i \quad (i = 1, 2, \ldots, m), \\ x &\in D := [0, M]^n, \end{aligned}$$

where $M > 0$.

This problem is an instance of Example 3.3.3. Indeed, metric projections onto any closed half spaces, including boxes, can be computed explicitly [2, Example 29.20], and the transformations (T1–2) enable us to build a firmly nonexpansive mapping whose fixed point set coincides with their intersection. Our GitHub repository, `https://github.com/iiduka-researches/201811-kaz`, provides the means to make a metric projection onto a given half space and the transformations (T1–2). We used these implementation in the following experiments.

Before discussing our experiments, let us examine the background of this problem. The goal is to find the most efficient production factors under funding-level restrictions [33, Section 6]. As mentioned in Section 3.3, the objective function f represents the ratio between the total profit (what is obtained) and the total cost (how much expenditure is required) as an efficiency indicator. We also described how the total profit and the total cost are modeled in Section 3.3. The total profit is the numerator of the objective function and is modeled with the Cobb-Douglas production function on the production factors $x \in \mathbb{R}^n$. The total cost is the denominator of the objective function and is modeled with the affine function on the production factors $x \in \mathbb{R}^n$. There are a variety of constraints on the funding level [33, Section 6]. These constraints represent the duties and restrictions of each production project $i = 1, 2, \ldots, m$. These indicators are modeled with affine functions and we set two parameters $\underline{p}_i, \overline{q}_i$ as lower and upper bounding constraints to the indicator of each project $i = 1, 2, \ldots, m$.

We conducted numerical experiments in three cases. First, in the unbounded constraint case, which is treated in the existing literature [33], we set $M := 100$ and did not guarantee the uniqueness of the optima, which is required for letting the generated sequence converge. Second, in the bounded constraint case, we set $M := 100$ and guaranteed the uniqueness of optima, as shown in Section 3.3. In practice, we cannot manufacture products infinitely because there are many restrictions on the amount of materials, capital, human resources, number and/or capacity of machines, environments, and so on. Therefore, this case has realistic experimental assumptions for optimizing production efficiency. Furthermore, we conducted an optimization over the generalized convex feasible sets.

We compared Algorithm 3.3.1 with Algorithm 3.4.1 called the exact quasi-subgradient method. In order for it to run, this algorithm requires a computation

Algorithm 3.4.1 The exact quasi-subgradient method [53, Algorithm (14)]

Require:
f: $\mathbb{R}^n \to \mathbb{R}$, $X \cap D \subset \mathbb{R}^n$: the feasible set;
P_X: $\mathbb{R}^n \to \mathbb{R}^n$, the metric projection onto the constraint set X;
$\{v_k\} \subset (0, \infty)$, $\{\alpha_k\} \subset (0, 1]$.
Ensure:
$\{x_k\} \subset \mathbb{R}^n$.
1: $x_1 \in \mathbb{R}^n$.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad g_k \in \partial^\star f(x_k) \cap \mathbf{S}$.
4: $\quad x_{k+1} := P_{X \cap D}(x_k - v_k g_k)$.
5: **end for**

of the metric projection onto the feasible set $X \cap D$. Here, we used a trust-region algorithm for constrained optimization, `trust-constr`, implemented as the `scipy.optimize.minimize` solver provided by the SciPy fundamental library for scientific computing [51]. This algorithm also requires the step-sizes $\{v_k\}$ for it to

run. We set its error tolerance a tenth of v_k for each $k = 1, 2, \ldots$. That is, we solved the subproblem to

$$\begin{aligned} &\text{find } \|x_{k+1} - (x_k - v_k g_k)\|^2 \leq \min_{u \in X} \|u - (x_k - v_k g_k)\|^2 + \frac{v_k}{10} \\ &\text{subject to } x_{k+1} \in X \cap D, \end{aligned}$$

as the computation of the metric projection in step 4 with the existing optimization solver.

In contrast to the existing algorithm which finds the metric projection onto the constraint set in the above way, we constructed a firmly nonexpansive mapping whose fixed point set coincides with the constraint set and gave it to Algorithm 3.3.1. Here, the constraint set is the intersection of the half-spaces $\{x : \underline{p}_i \leq \langle b_i, x \rangle\}$ and $\{x : \langle b_i, x \rangle \leq \overline{p}_i\}$ for $i = 1, 2, \ldots, m$. The metric projection onto each half-space can be easily computed [2, Example 29.20]:

$$P_{\{x:\underline{p}_i \leq \langle b_i, x \rangle\}}(x) := \begin{cases} x & (\underline{p}_i \leq \langle b_i, x \rangle), \\ x + \frac{\underline{p}_i - \langle b_i, x \rangle}{\|b_i\|^2} b_i & (\text{otherwise}) \end{cases},$$

$$P_{\{x:\langle b_i, x \rangle \leq \overline{p}_i\}}(x) := \begin{cases} x & (\langle b_i, x \rangle \leq \overline{p_i}), \\ x + \frac{\overline{p}_i - \langle b_i, x \rangle}{\|b_i\|^2} b_i & (\text{otherwise}) \end{cases}$$

for all $i = 1, 2, \ldots, m$ and for any $x \in H$. To construct a nonexpansive mapping whose fixed point set coincides with the intersection of the above sets, we use the transformation (T1) and construct

$$\tilde{T}(x) := \frac{1}{m} \sum_{i=1}^{m} \frac{P_{\{x:\underline{p}_i \leq \langle b_i, x \rangle\}}(x) + P_{\{x:\langle b_i, x \rangle \leq \overline{p}_i\}}(x)}{2}$$

for any $x \in H$. This mapping $\tilde{T}$ is nonexpansive, but not firmly nonexpansive. Therefore, we convert the nonexpansive mapping $\tilde{T}$ into the corresponding firmly nonexpansive one T by using the transformation (T2):

$$T := \frac{\mathrm{Id} + \tilde{T}}{2}.$$

We gave T to Algorithm 3.3.1 in the experiment.

Our experimental environment was as follows: Python 3.6.6 with NumPy 1.15.0 [69] and SciPy 1.1.0 [51] libraries on macOS High Sierra version 10.13.6 on Mac Pro (Late 2013) with a 3 GHz 8 Cores Intel Xeon E5 CPU and 32GB 1800MHz DDR3 ECC memory. We used the `time.process_time` method for the evaluating computational time of each algorithm. The method was implemented with the `clock_gettime(2)` system call and had a 10^{-6} second resolution. Our GitHub repository, `https://github.com/iiduka-researches/201811-kaz`, provides the codes that were used in the experiments. It has the implementations of Algorithms 3.3.1 and 3.4.1 and

miscellaneous utilities including higher-order functions to be used for composing a nonexpansive mapping.

We ran Algorithms 3.3.1 and 3.4.1 with five different randomly chosen initial points, limited their computational time to ten seconds, and evaluated the average of the computed number of iterations k and the following values:

$$V_{\mathrm{func}} := \frac{1}{8}\sum_{i=1}^{8} f(x^{\star}_{(i)}), \quad V_{\mathrm{dist}} := \frac{1}{8}\sum_{i=1}^{8} \left\| x^{\star}_{(i)} - T(x^{\star}_{(i)}) \right\|,$$

where $x^{\star}_{(i)}$ is the solution obtained for each sampling $i = 1, 2, \ldots, 8$.

3.4.1 Unbounded constraint case

Here, we ran Algorithms 3.3.1 and 3.4.1 on Problem 3.4.1 with the following settings: $n := 100$; $m := 100$; $a_0, c_0 \in (0, 10]$, $\tilde{a} \in (0, 1]^n$, $c \in (0, 10]^n$ were chosen randomly; $a := \tilde{a} / \sum_{i=1}^{n} a_i$; $b_i \in [0, 1)^n$, $\underline{p}_i \in [0, 25\,\|b_i\|)$ were chosen randomly for each $i = 1, 2, \ldots, m$; $\overline{p}_i := \infty$ for all $i = 1, 2, \ldots, m$; and $M := +\infty$.

The experimental results are shown in Table 3.1. The proposed algorithm (Algorithm 3.3.1) can iterate the computation more times than the existing one within the same computational time. Algorithm 3.3.1 does not require any subproblem to be solved, while Algorithm 3.4.1 requires one to be solved in order to find a metric projection onto the constraint set. Therefore, the required time for computing an iteration of Algorithm 3.3.1 is much less than that of Algorithm 3.4.1. According to the values of D, both Algorithms 3.3.1 and 3.4.1 for any step-size (and no matter whether a constant or diminishing step-size rule was used) can obtain the solution belonging to the constraint set. Indeed, our experimental environment (NumPy) used the `float64`

Table 3.1: Results of unbounded constraint case.

	k	V_{func}	V_{dist}
Alg. 3.3.1 ($v_k := 10^{-1}$)	11495.5	-0.00092189	$3.18571477 \times 10^{-13}$
Alg. 3.3.1 ($v_k := 10^{-2}$)	11539.5	-0.00061939	$3.17169706 \times 10^{-13}$
Alg. 3.3.1 ($v_k := 10^{-3}$)	11459.2	-0.00049238	$3.10991757 \times 10^{-13}$
Alg. 3.3.1 ($v_k := 10^{-1}/k$)	11474.5	-0.00046070	$3.05626418 \times 10^{-13}$
Alg. 3.3.1 ($v_k := 10^{-2}/k$)	11264.5	-0.00045498	$3.16790812 \times 10^{-13}$
Alg. 3.3.1 ($v_k := 10^{-3}/k$)	11318.4	-0.00045426	$3.28061131 \times 10^{-13}$
Alg. 3.4.1 ($v_k := 10^{-1}$)	160.4	-0.00050596	$4.04768150 \times 10^{-13}$
Alg. 3.4.1 ($v_k := 10^{-2}$)	39.6	-0.00045707	$4.12640954 \times 10^{-13}$
Alg. 3.4.1 ($v_k := 10^{-3}$)	25.1	-0.00045306	$4.00174170 \times 10^{-13}$
Alg. 3.4.1 ($v_k := 10^{-1}/k$)	36.1	-0.00045750	$4.23263890 \times 10^{-13}$
Alg. 3.4.1 ($v_k := 10^{-2}/k$)	26.6	-0.00045353	$3.99258739 \times 10^{-13}$
Alg. 3.4.1 ($v_k := 10^{-3}/k$)	20.5	-0.00045275	$4.10328799 \times 10^{-13}$

data type (double precision float: sign bit, 11-bit exponent, and 52-bit mantissa) to express a real number, and its resolution is 10^{-15}. By considering the number of dimensions as well, we can regard all values of D to be almost zero. Let us examine the functional values of the obtained solutions. When we applied Algorithms 3.3.1 and 3.4.1 to the problem with the same step-size, we found that the function value of the solution obtained by Algorithm 3.3.1 is better than that of Algorithm 3.4.1. In particular, the function value obtained by Algorithm 3.3.1 with $v_k := 10^{-1}$ is nearly twice as good as Algorithm 3.4.1 with the same step-size. Since Algorithm 3.3.1 can iterate the main loop more times than Algorithm 3.4.1, it can reduce the functional value sufficiently.

3.4.2 Bounded constraint case

Next, we evaluated Algorithms 3.3.1 and 3.4.1 when they were run with the following settings: $n := 100$; $m := 100$; $a_0, c_0 \in (0, 10]$, $\tilde{a} \in (0, 1]^n$, $c \in (0, 10]^n$ were chosen randomly; $a := \tilde{a} / \sum_{i=1}^{n} a_i$; $b_i \in [0, 1)^n$, $\underline{p}_i \in [0, 25 \|b_i\|)$, $\overline{p}_i \in (75 \|b_i\|, 100 \|b_i\|]$ were chosen randomly for each $i = 1, 2, \ldots, m$; and $M := 100$. As shown in Example 3.3.3, this case satisfies Assumption 3.3.1. Therefore, the sequence generated by Algorithm 3.3.1 is guaranteed to converge to some optimum.

The experimental results are shown in Table 3.2. The existing algorithm (Algorithm 3.4.1) with all step-size rules could not compute even one iteration within the time limit, 10 seconds. Algorithm 3.4.1 required about 15 seconds to compute the first iteration. In this case, Algorithm 3.4.1 must compute the metric projection onto the intersection of two hundred halfspaces and a box, but this intersection is too complex in shape to compute quickly. Therefore, it could not deal with this instance. In contrast, Algorithm 3.3.1 solved this instance. In particular, Algorithm 3.3.1 with $v_k := 10^{-1}$ found the solution having the best function value and belonging to the constraint set. Therefore, it can solve problems even if their constraint sets have complex shapes.

Table 3.2: Results of bounded constraint case.

	k	V_{func}	V_{dist}
Alg. 3.3.1 ($v_k := 10^{-1}$)	6254.0	-0.00092536	$1.21925957 \times 10^{-13}$
Alg. 3.3.1 ($v_k := 10^{-2}$)	6208.9	-0.00050503	$5.22534774 \times 10^{-3}$
Alg. 3.3.1 ($v_k := 10^{-3}$)	6276.1	-0.00019717	$6.42589842 \times 10^{-4}$
Alg. 3.3.1 ($v_k := 10^{-1}/k$)	6293.8	-0.00014214	$1.28281187 \times 10^{-5}$
Alg. 3.3.1 ($v_k := 10^{-2}/k$)	6245.6	-0.00014163	$8.56673942 \times 10^{-6}$
Alg. 3.3.1 ($v_k := 10^{-3}/k$)	6294.0	-0.00014162	$8.33884360 \times 10^{-6}$
All Results of Alg. 3.4.1	0.0	—	—

3.4.3 Optimization over generalized convex feasible sets

Finally, let us consider the case in which conflicts of constraints exist, i.e., the intersection of the constraint sets may be empty. Even in this case, the proposed algorithm can be used if the constraints are extended to generalized convex feasible sets, such as described in Section 3.2.

In the two previous subsections, we computed the metric projection with the constrained smooth optimization solver provided by the SciPy library. However, it is difficult to find the metric projection onto the set of minimizers of the functional (3.1) due to the discontinuity of its Hessian matrix and the complexity of the problem. Therefore, in this subsection, we will examine only the performance of Algorithm 3.3.1.

The settings of this experiment were as follows: $n := 100$; $m := 100$; $a_0, c_0 \in (0, 10]$, $\tilde{a} \in (0, 1]^n$, $c \in (0, 10]^n$ were chosen randomly; $a := \tilde{a} / \sum_{i=1}^{n} a_i$; $b_i \in [0, 1)^n$, $\underline{p}_i, \overline{p}_i \in [0, 100\,\|b_i\|)$ were chosen randomly for each $i = 1, 2, \ldots, m$; and $M := +\infty$. The existence of a constraint $i \in \{1, 2, \ldots, m\}$ which satisfies $\overline{p}_i < \underline{p}_i$ was guaranteed. This implies that the intersection of at least one pair of constraints is empty. We used [45, Definition (9)] for constructing a firmly nonexpansive mapping whose fixed point set coincides with the constraint set. As in the previous subsection, this case also satisfies Assumption 3.3.1.

The experimental results are shown in Table 3.3. Algorithm 3.3.1 solved the problem similarly for each step-size rule; the results scarcely depended on the step-size rule. With a constant step-size $v_k = 10^{-1}$ for all $k \in \mathbb{N}$, it gave the best score in terms of V_{func} and V_{dist}. Although the time required for computing one iteration exceeded those of the two previous experiments, it approximated the solution of this complicated problem in 10 seconds.

Table 3.3: Results of bounded constraint case.

	k	V_{func}	V_{dist}
Alg. 3.3.1 ($v_k := 10^{-1}$)	4988.9	-0.00015650	$2.48927983 \times 10^{-1}$
Alg. 3.3.1 ($v_k := 10^{-2}$)	4914.4	-0.00012585	$2.58656058 \times 10^{-1}$
Alg. 3.3.1 ($v_k := 10^{-3}$)	4833.6	-0.00012377	$2.59531605 \times 10^{-1}$
Alg. 3.3.1 ($v_k := 10^{-1}/k$)	4823.4	-0.00012363	$2.59665629 \times 10^{-1}$
Alg. 3.3.1 ($v_k := 10^{-2}/k$)	4818.4	-0.00012378	$2.59656373 \times 10^{-1}$
Alg. 3.3.1 ($v_k := 10^{-3}/k$)	4773.6	-0.00012380	$2.59626252 \times 10^{-1}$

Section 3.5. Conclusion

We proposed the novel algorithm for solving the constrained quasiconvex optimization problem even if the metric projection onto its constraint set cannot be computed easily. We showed its convergence for constant and diminishing step-size rules. When the step-size is constant, the limit inferiors of the functional value and the degree of

approximation to the fixed point are guaranteed to be optimal and tolerate errors proportioned to the step-size. When the step-size is diminishing, the existence of a subsequence of the generated sequence such that it converges to the solution of the problem is ensured. Furthermore, when the problem satisfies certain conditions, the whole generated sequence converges to the solution.

The numerical experiments showed that our algorithm runs stably and lightly even if the constraint set is too complex for the existing method to run quickly. Therefore, the proposed algorithm is useful for solving complicated constrained quasiconvex optimization problems.

Chapter 4

Convergence Rate Analysis of Fixed Point Quasiconvex Subgradient Method

This chapter investigates the rate of convergence of the sequence generated by the fixed point quasiconvex subgradient method. As we saw in the previous chapter, the fixed point quasiconvex subgradient method solves the constraint quasiconvex optimization problem, whose task is to minimize a given quasiconvex functional over the fixed point set of a given nonexpansive mapping. In this chapter, we evaluate its efficiency by discussing the rate of convergence of the method in terms of both the value of the objective functional and the distance to the constraint set. Furthermore, we prove a theorem which provides a sufficient condition for ensuring that the generated sequence converges to the optimal solution in a finite number of iterations.

The contents of this chapter are based on

[29] K. Hishinuma and H. Iiduka. Convergence rate analyses of fixed point quasiconvex subgradient method. Joint Conference NACA-ICOTA2019: International Conference on Nonlinear Analysis and Convex Analysis, International Conference on Optimization: Techniques and Applications (Oral), 2019;

[31] K. Hishinuma and H. Iiduka. Fixed point quasiconvex subgradient method. *European Journal of Operational Research*, 282(2):428–437, 2020;

[32] K. Hishinuma and H. Iiduka. Supplementary data S1 for the article entitled "fixed point quasiconvex subgradient method". `https://doi.org/10.1016/j.ejor.2019.09.037`, 2020.

Section 4.1. Introduction

Continuing from the previous chapter, this chapter considers the constrained quasiconvex optimization problem and the properties of the fixed point quasiconvex subgradient method which can be used to solve it. The previous chapter presented

convergence theorems of the proposed method and an experimental efficiency evaluation. This chapter unravels the (theoretical) efficiency of the fixed point quasiconvex subgradient method on the basis of several diffferent convergence rate analyses.

Many applications in economics, engineering, and management science can be framed as constrained quasiconvex optimization problems [33, 35]. In particular, the fractional programming problem is an important instance of constrained quasiconvex optimization, because it can be used as a model for optimizing ratio indicators such as the debt/equity ratio in financial and corporate planning, inventory/sales and output/employee ratios in production planning, and cost/patient and nurse/patient ratios in health care and hospital planning [86]. The previous chapter proposed the fixed point quasiconvex subgradient method for solving constrained quasiconvex optimization problems. This method optimizes a quasiconvex objective functional over the fixed point set of a given nonexpansive mapping. The convergence theorem guarantees that the generated sequence converges to a solution under certain conditions. However, in practical use, we cannot run the algorithm forever. Hence, in addition to showing whether the generated sequence converges, we need to show how fast the generated sequence converges; i.e., we have to analyze its convergence rate.

Reference [53] analyses the rate of convergence of the sequence generated by the quasiconvex subgradient method which uses the metric projection onto the constraint set. Reference [33] further analyses it under consideration of noise and errors in the generated sequence. However, both references consider a quasiconvex subgradient method that uses the metric projection onto the constraint set. Hence, they both assume that the metric projection is computable. In contrast to these studies, the previous chapter describes an algorithm which does not need a computable metric projection onto the constraint set. Instead of the metric projection, that algorithm uses a nonexpansive mapping whose fixed point set expresses the constraint set. However, the previous chapter evaluated its efficiency only in numerical experiments.

In this chapter, we analyze the rate of convergence of the algorithm presented in the previous chapter from three viewpoints. The first viewpoint is the behavior of the level sets made by the generated sequence. This shows the efficiency of minimizing the objective functional. Next, we analyze the rate of convergence in terms of the value of the objective functional. In this analysis, we use the result from the first viewpoint. Finally, we analyze the rate of convergence in terms of the distance to the constraint set. The existing studies [33, 53] do not consider this, because the metric projection obviously ensures the generated sequence is in the constraint set. But, the nonexpansive mappings used in the proposed algorithm cannot guarantee that the generated sequence will be in the constraint set. Hence, this chapter evaluates the efficiency of the proposed algorithm when the generated sequence approaches the constraint set. In addition, it proves a finite convergence theorem for the fixed point quasiconvex subgradient method. This theorem provides the condition under which the generated sequence reaches the solution in a finite number of iterations.

The convergence rate analyses of the fixed point quasiconvex subgradient method turns out to be similar to the analyses of the existing methods. This shows that the fixed point quasiconvex subgradient method can be regarded as an extension of the

existing quasiconvex subgradient method. To improve the usability of the algorithm, the convergence rate analyses, including proof of the finite convergence theorem, give a barometer of how many times we have to iterate the algorithm before it obtains a good enough approximation.

This chapter is organized as follows. Section 4.2 gives the mathematical preliminaries. Section 4.3 discusses the rate of convergence of the fixed point quasiconvex subgradient method. Section 4.4 concludes this chapter.

Section 4.2. Mathematical preliminaries

4.2.1 Notation and definitions

We use the following notation in this chapter. Let H be a real Hilbert space with inner product $\langle\cdot,\cdot\rangle : H \times H \to \mathbb{R}$ and its induced norm $\|\cdot\| : H \to \mathbb{R}$. $\mathbb{N}$ is the set of natural numbers without zero, and $\mathbb{R}$ is the set of real numbers. $\mathbf{B} := \{x \in H : \|x\| \leq 1\}$ is the unit ball in this Hilbert space, and $\mathbf{S} := \{x \in H : \|x\| = 1\}$ is the unit sphere in that space. Id is the identity mapping of H onto itself. The boundary of a set $C \subset H$ is denoted by $\operatorname{bd} C$; the closure of this set is denoted by $\operatorname{cl} C$.

The metric projection onto a closed, convex set $C \subset H$ is denoted by P_C and defined as $P_C(x) \in C$ and $\|x - P_C(x)\| = \inf_{y \in C} \|x - y\|$ for any $x \in H$. For any $\alpha \in \mathbb{R}$, the α-slice of a functional $f : H \to \mathbb{R}$ is denoted by $\operatorname{lev}_{<\alpha} f := \{x \in H : f(x) < \alpha\}$. A functional $f : H \to \mathbb{R}$ is called *quasiconvex* if $f(\alpha x + (1-\alpha)y) \leq \max\{f(x), f(y)\}$ for every $x, y \in H$ and $\alpha \in [0, 1]$ [1, Definition 5.1], [19, Definition (4.4)]. The effective domain of a functional $f : H \to \mathbb{R}$ is denoted by $\operatorname{dom}(f) := \{x \in H : f(x) < \infty\}$. A mapping $T : H \to H$ is said to be nonexpansive if $\|T(x) - T(y)\| \leq \|x - y\|$ for any $x, y \in H$, and it is said to be firmly nonexpansive if $\|T(x) - T(y)\|^2 + \|(\mathrm{Id} - T)x - (\mathrm{Id} - T)y\|^2 \leq \|x - y\|^2$ for any $x, y \in H$. Obviously, a firmly nonexpansive mapping is also a nonexpansive mapping [2, Subchapter 4.1]. The properties of these nonexpansivities are described in detail in [2, Chapter 4], [87, Chapter 6]. The fixed point set of a mapping $T : H \to H$ is denoted by $\operatorname{Fix}(T) := \{x \in H : T(x) = x\}$.

For given a point $x \in H$, we call the set $\partial^\star f(x) := \{g \in H : \langle g, y - x\rangle \leq 0 \ (y \in \operatorname{lev}_{<f(x)} f)\}$ the subdifferential of the quasiconvex functional f at a point $x \in H$ [33, Definition 2.3], [35, Definition 2.1], [53, Definition (9)], [54, Section 1]. We also call its element a *subgradient*.

4.2.2 Main problem and propositions

Let us recall the constrained quasiconvex optimization problem: given a quasiconvex continuous functional $f : H \to \mathbb{R}$ and two nonempty, closed, convex sets $X, D \subset H$ whose intersection is also nonempty, we would like to

$$\text{minimize } f(x) \text{ subject to } x \in X \cap D. \tag{4.1}$$

We define the *set of minima* and the *minimum value* of Problem 4.1 by $X^\star := \operatorname{argmin}_{x\in X\cap D} f(x)$ and $f_\star := \inf_{x\in X\cap D} f(x)$, respectively.

We list the conditions assumed throughout in this chapter.

Assumption 4.2.1. We suppose that

(A1) the effective domain $\operatorname{dom}(f)$ coincides with the whole space H;
(A2) there exists some firmly nonexpansive mapping $T : H \to H$ whose fixed point set $\operatorname{Fix}(T)$ coincides with the constraint set X;
(A3) the constraint set $X = \operatorname{Fix}(T)$ and the feasible set $X \cap D$ are nonempty and there exists at least one minima, i.e. $X^\star \neq \emptyset$.

We will use the following propositions.

Proposition 4.2.1. *Let $C \subset H$ be a nonempty, convex set, and suppose that $x \in C$, $y \notin C$. Then, there exists $\alpha \in [0,1]$ such that $x + \alpha(y-x) \in \operatorname{bd}(C)$.*

Proof. Define $\alpha := \sup\{\hat{\alpha} \in [0,1] : x + \hat{\alpha}(y-x) \in C\}$ and fix $\epsilon > 0$ arbitrarily. From the properties of the supremum, there exists $\beta > \alpha - \epsilon/\|y-x\|$ such that $x + \beta(y-x) \in C$. Thus, we have

$$\|(x+\alpha(y-x)) - (x+\beta(y-x))\| = (\alpha-\beta)\|y-x\| < \epsilon.$$

Since $\epsilon > 0$ was chosen arbitrarily, the above inequality implies that the point $(x + \alpha(y-x))$ is an adherent point of C. Furthermore, there exists $\gamma > \alpha + \epsilon/\|y-x\|$ such that $x + \gamma(y-x) \notin C$ due to the properties of the supremum. Hence, we also have

$$\|(x+\alpha(y-x)) - (x+\gamma(y-x))\| = (\gamma-\alpha)\|y-x\| < \epsilon.$$

Since $\epsilon > 0$ was chosen arbitrarily, the above inequality implies that the point $(x + \alpha(y-x))$ is an adherent point of the complement of C. Therefore, $x+\alpha(y-x)$ belongs to the boundary of C. This completes the proof. □

Proposition 4.2.2 ([2, Corollary 2.15])**.** *Let $x, y \in H$, and let $\alpha \in \mathbb{R}$. Then,*

$$\|\alpha x + (1-\alpha)y\|^2 = \alpha\|x\|^2 + (1-\alpha)\|y\|^2 - \alpha(1-\alpha)\|x-y\|^2$$

holds.

Section 4.3. Proposed method and its efficiency

4.3.1 Proposed method

In this chapter, we discuss the efficiency of the fixed point quasiconvex subgradient method for solving Problem 4.1. First, let us recall the fixed point quasiconvex subgradient method presented in the previous chapter. For the problem defined by

Algorithm 4.3.1 Fixed point quasiconvex subgradient method

Require:
$f\colon H \to \mathbb{R}$, $T\colon H \to H$, $D \subset H$;
$\{v_k\} \subset (0, \infty)$, $\{\alpha_k\} \subset (0, 1]$.
Ensure:
$\{x_k\} \subset D$.
1: $x_1 \in D$.
2: **for** $k = 1, 2, \ldots$ **do**
3: $g_k \in \partial^\star f(x_k) \cap \mathbf{S}$.
4: $x_{k+1} := P_D(\alpha_k x_k + (1 - \alpha_k)T(x_k - v_k g_k))$.
5: **end for**

(f, T, D) and for the given parameters $\{v_k\}$ and $\{\alpha_k\}$, the algorithm generates the sequence $\{x_k\}$. Step 4 yields each element in $\{x_k\}$ by using the generator composed of the subgradient method iterator $x_k - v_k g_k$ to improve the approximations with respect to the functional value and the Krasnosel'skiĭ-Mann iterator [55, 63] $\alpha_k \operatorname{Id} + (1 - \alpha_k)T$ to improve approximations with respect to the distance to the fixed point set $\operatorname{Fix}(T)$.

The previous chapter examined the convergence of Algorithm 4.3.1 with two different step-size rules: constant and diminishing. A constant step size $\{v_k\}$ is to a constant value $v \in \mathbb{R}$ if we adopt the constant step-size rule, while it is decreasing to satisfy certain conditions if we adopt the diminishing step-size rule. Here, let us review the results.

Assumption 4.3.1. (A4) For any $k \in \mathbb{N}$ such that $f_\star < f(x_k)$ and for all $x^\star \in X^\star$, the functional f satisfies the Hölder condition with degree $\beta > 0$ at the point $x^\star$ on the set $\operatorname{cl}(\operatorname{lev}_{<f(x_k)} f)$.
(A5) The generated sequence $\{x_k\}$ is bounded.
(A6) The real sequence $\{\alpha_k\} \subset (0, 1]$ satisfies $0 < \liminf_{k\to\infty} \alpha_k \leq \limsup_{k\to\infty} \alpha_k < 1$.

Lemma 4.3.1. *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 4.3.1. Suppose that Assumptions 4.2.1 and (A4) hold. Then, for any $k \in \mathbb{N}$ that satisfies $f_\star < f(x_k)$, the following inequalities hold.*

$$\begin{aligned} \|x_{k+1} - x^\star\|^2 &\leq \|x_k - x^\star\|^2 - 2v_k(1 - \alpha_k)\langle g_k, x_k - x^\star\rangle + (1 - \alpha_k)v_k^2 \qquad (4.2) \\ &\leq \|x_k - x^\star\|^2 - 2v_k(1 - \alpha_k)\left(\frac{f(x_k) - f_\star}{L}\right)^{\frac{1}{\beta}} + (1 - \alpha_k)v_k^2. \end{aligned}$$

Theorem 4.3.1. *Let $v > 0$ and $v_k := v$ for all $k \in \mathbb{N}$ and $\{x_k\} \subset H$ be a sequence generated by Algorithm 4.3.1. Suppose that Assumptions 4.2.1 and 4.3.1 hold. Then, the sequence $\{x_k\}$ satisfies*

$$\liminf_{k\to\infty} f(x_k) \leq f_\star + L\left(\frac{v}{2}\right)^\beta, \text{ and } \liminf_{k\to\infty} \|x_k - T(x_k)\|^2 \leq Mv$$

for some $M \geq 0$.

Theorem 4.3.2. *Let* $\{x_k\} \subset H$ *be a sequence generated by Algorithm 4.3.1. Suppose that*

(i) Assumptions 4.2.1 and 4.3.1 hold,
(ii) and the real sequence $\{v_k\} \subset (0, \infty)$ *satisfies*

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

Then, there exists a subsequence of the generated sequence $\{x_k\}$ *which converges weakly to a point in* $X^\star$. *In addition, if*

(iii) the whole space H *is an* N*-dimensional Euclidean space* $\mathbb{R}^N$,
(iv) and the solution $x^\star \in X^\star$ *is unique,*

then, the whole sequence $\{x_k\}$ *converges to this unique solution* $x^\star$.

4.3.2 Value of the objective functional

We discuss the rate of convergence of Algorithm 4.3.1 in terms of the value of the objective functional and the distance to the fixed point set. Furthermore, we discuss a sufficient condition to obtain finite convergence to some solution.

Let us start with the convergence rate analysis in terms of the objective function. Here, we will use the following concepts originally introduced in [33, 53].

Definition 4.3.1 ([33, Section 5], [53, Section 6])**.** Let $x^\star \in X^\star$. Define the following notations:

(i) $x_k^\star :\in \operatorname{argmin}_{x\in\{x_1,x_2,\ldots,x_k\}} f(x)$,
(ii) $r_k := \sup\{r > 0 : x^\star + r\mathbf{B} \subset \operatorname{lev}_{<f(x_k^\star)} f\}$.

$x_k^\star$ expresses the best solution acquired until the k-th iteration, and r_k expresses the distance between the level set of $x_k^\star$ and the optimal solution. The difference between the above definitions and the original ones is in considering the possibility that the generated sequence may be out of the fixed point set, in other words, the constraint set.

We start by proving the following lemma which leads us to the convergence rate of Algorithm 4.3.1.

Lemma 4.3.2. *Let* $\{x_k\} \subset H$ *be a sequence generated by Algorithm 4.3.1, and suppose that Assumptions 4.3.1 and 4.2.1 hold. Assume that the sequence* $\{v_k\}$ *is bounded. Then,*

$$r_k \leq \frac{\|x_i - x^\star\|^2 + \sum_{j=i}^{k}(1-\alpha_j)v_j^2}{2\sum_{j=i}^{k} v_j(1-\alpha_j)}$$

for any $x^\star \in X^\star$, $k \in \mathbb{N}$, *and* $i \in \{1, 2, \ldots, k\}$.

Proof. Fix $x^\star \in X^\star$, $k \in \mathbb{N}$, and $i \in \{1, 2, \ldots, k\}$ arbitrarily. If r_k is nonpositive, the statement obviously holds. Therefore, let us consider the case where r_k is positive in the following. Fix $\delta \in (0, r_k)$ arbitrarily. The definition of r_k and monotonicity of the sequence $\{r_k\}$ imply that $x^\star - \delta g_j$ belongs to the level set $\mathrm{lev}_{<f(x_j^\star)} f \subset \mathrm{lev}_{<f(x_j)} f$ for any $j = 1, 2, \ldots, k$. Therefore, $\langle g_j, (x^\star - \delta g_j) - x_j \rangle \leq 0$ holds for all $j = 1, 2, \ldots, k$. Rearranging this inequality with the property $\|g_j\| = 1$, we have

$$\langle g_j, x^\star - x_j \rangle \leq \delta$$

for all $j = 1, 2, \ldots, k$. Here, the assumption $r_k > 0$ implies that $f_\star < f(x_j)$ for all $j = 1, 2, \ldots, k$. Hence, all assumptions of Lemma 4.3.1 are satisfied and

$$\begin{aligned} \|x_{j+1} - j^\star\|^2 &\leq \|x_j - x^\star\|^2 - 2v_j(1-\alpha_j)\langle g_j, x_j - x^\star \rangle + (1-\alpha_j)v_j^2 \\ &\leq \|x_j - x^\star\|^2 - 2\delta v_j(1-\alpha_j) + (1-\alpha_j)v_j^2 \end{aligned}$$

is guaranteed by inequality (4.2) for all $j = 1, 2, \ldots, k$. Summing the above inequalities from $j = i$ to $j = k$ yields

$$0 \leq \|x_i - x^\star\|^2 - 2\delta \sum_{j=i}^{k} v_j(1-\alpha_j) + \sum_{j=i}^{k}(1-\alpha_j)v_j^2.$$

Transposing the term of δ, we get

$$\delta \leq \frac{\|x_i - x^\star\|^2 + \sum_{j=i}^{k}(1-\alpha_j)v_j^2}{2\sum_{j=i}^{k} v_j(1-\alpha_j)}.$$

The arbitrariness of $\delta \in (0, r_k)$ implies that

$$r_k \leq \frac{\|x_i - x^\star\|^2 + \sum_{j=i}^{k}(1-\alpha_j)v_j^2}{2\sum_{j=i}^{k} v_j(1-\alpha_j)}.$$

This completes the proof. □

In general settings, we can use this lemma to analyze the convergence rate in terms of r_k. The following two propositions present such an analysis for constant and diminishing step-size rules.

Proposition 4.3.1 (Convergence rate analysis for constant step-size rule)**.** *Let* $\{x_k\} \subset H$ *be a sequence generated by Algorithm 4.3.1, and suppose that the assumptions in Theorem 4.3.1 hold. Then, there exists a number* $k_0 \in \mathbb{N}$ *such that*

$$r_k \leq \frac{1}{1 - \limsup_{j\to\infty} \alpha_j} \left(\frac{\|x_{k_0} - x^\star\|^2}{(k - k_0 + 1)v} + \left(1 - \frac{1}{2}\liminf_{j\to\infty} \alpha_j\right) v \right)$$

holds for all $k \geq k_0$, in other words,

$$r_k = \mathcal{O}(1/k + v).$$

Furthermore,

$$r_k \leq \frac{\|x_1 - x^\star\|^2}{2(1-\alpha)kv} + \frac{1}{2}v$$

holds for all $k \in \mathbb{N}$ when the sequence $\{\alpha_k\}$ satisfies $\alpha_k = \alpha \in (0,1)$ for all $k \in \mathbb{N}$.

Proof. Assumption (A6) implies that there exists a number $k_0 \in \mathbb{N}$ such that

$$0 < \frac{1}{2}\left(1 - \limsup_{k\to\infty} \alpha_k\right) \leq 1 - \alpha_k \leq 1 - \frac{1}{2}\liminf_{k\to\infty} \alpha_k < 1$$

for all $k \geq k_0$. Therefore, Lemma 4.3.2 with $i := k_0$ leads to the finding that

$$\begin{aligned} r_k &\leq \frac{\|x_{k_0} - x^\star\|^2 + \sum_{j=k_0}^{k}(1-\alpha_j)v_j^2}{2\sum_{j=k_0}^{k} v_j(1-\alpha_j)} \\ &\leq \frac{\|x_{k_0} - x^\star\|^2 + (1 - \liminf_{j\to\infty}\alpha_j/2)(k-k_0+1)v^2}{(1-\limsup_{j\to\infty}\alpha_j)(k-k_0+1)v} \\ &= \frac{1}{1-\limsup_{j\to\infty}\alpha_j}\left(\frac{\|x_{k_0} - x^\star\|^2}{(k-k_0+1)v} + \left(1 - \frac{1}{2}\liminf_{j\to\infty}\alpha_j\right)v\right) \end{aligned}$$

for all $k \geq k_0$.

Furthermore, if $\{\alpha_k\}$ satisfies $\alpha_k = \alpha \in (0,1)$ for all $k \in \mathbb{N}$, Lemma 4.3.2 with $i := 1$ leads to

$$\begin{aligned} r_k &\leq \frac{\|x_1 - x^\star\|^2 + \sum_{j=i}^{k}(1-\alpha_j)v_j^2}{2\sum_{j=i}^{k} v_j(1-\alpha_j)} \\ &= \frac{\|x_1 - x^\star\|^2 + (1-\alpha)kv^2}{2(1-\alpha)kv} \\ &= \frac{\|x_1 - x^\star\|^2}{2(1-\alpha)kv} + \frac{1}{2}v \end{aligned}$$

for all $k \in \mathbb{N}$. This completes the proof. □

Proposition 4.3.2 (Convergence rate analysis for diminishing step-size rule)**.** *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 4.3.1, and suppose that the assumptions in Theorem 4.3.2 hold. Let c be a positive real number and assume that $v_k = c/k$ for all $k \in \mathbb{N}$. Then, there exists a number $k_0 \in \mathbb{N}$ such that*

$$r_k \leq \frac{\|x_{k_0} - x^\star\|^2 + 2c^2(1 - \liminf_{j\to\infty}\alpha_j/2)}{2c(1-\limsup_{j\to\infty}\alpha_j)(\log(k+1) - \log(k_0))}$$

holds for all $k \geq k_0$, in other words,

$$r_k = \mathcal{O}(1/\log(k+1)).$$

In addition,

$$r_k \leq \frac{\|x_1 - x^\star\|^2 + 2c^2(1-\alpha)}{2c(1-\alpha)\log(k+1)}$$

holds for all $k \in \mathbb{N}$ when the sequence $\{\alpha_k\}$ satisfies $\alpha_k = \alpha \in (0,1)$ for all $k \in \mathbb{N}$.

Proof. Assumption (A6) implies that there exists a number $k_0 \in \mathbb{N}$ such that

$$0 < \frac{1}{2}\left(1 - \limsup_{k\to\infty} \alpha_k\right) \leq 1 - \alpha_k \leq 1 - \frac{1}{2}\liminf_{k\to\infty} \alpha_k < 1$$

holds for all $k \geq k_0$. Therefore, Lemma 4.3.2 with $i := k_0$ leads to the finding that

$$\begin{aligned} r_k &\leq \frac{\|x_{k_0} - x^\star\|^2 + \sum_{j=k_0}^k (1-\alpha_j)v_j^2}{2\sum_{j=k_0}^k v_j(1-\alpha_j)} \\ &\leq \frac{\|x_{k_0} - x^\star\|^2 + c^2(1 - \liminf_{j\to\infty}\alpha_j/2)\sum_{j=k_0}^k 1/j^2}{2c(1-\limsup_{j\to\infty}\alpha_j)\sum_{j=k_0}^k 1/j} \end{aligned}$$

holds for all $k \geq k_0$. Using inequalities $\sum_{j=1}^\infty 1/j^2 \leq 2$, $\log(k+1) - \log(k_0) \leq \sum_{j=k_0}^k 1/j$, we have

$$r_k \leq \frac{\|x_{k_0} - x^\star\|^2 + 2c^2(1 - \liminf_{j\to\infty}\alpha_j/2)}{2c(1-\limsup_{j\to\infty}\alpha_j)(\log(k+1) - \log(k_0))}$$

for all $k \geq k_0$.

In addition, if $\{\alpha_k\}$ satisfies $\alpha_k = \alpha \in (0,1)$ for all $k \in \mathbb{N}$, Lemma 4.3.2 with $i := 1$ leads to

$$\begin{aligned} r_k &\leq \frac{\|x_1 - x^\star\|^2 + \sum_{j=i}^k (1-\alpha_j)v_j^2}{2\sum_{j=i}^k v_j(1-\alpha_j)} \\ &= \frac{\|x_1 - x^\star\|^2 + c^2(1-\alpha)\sum_{j=i}^k 1/j^2}{2(1-\alpha)\sum_{j=i}^k 1/j} \\ &\leq \frac{\|x_1 - x^\star\|^2 + 2c^2(1-\alpha)}{2(1-\alpha)\log(k+1)} \end{aligned}$$

for all $k \in \mathbb{N}$. This completes the proof. □

The following theorem guarantees that the convergence rate with respect to the functional value can be bounded from above by one with respect to r_k under certain assumptions. This implies that, under the assumptions, we can deduce the convergence rate with respect to the functional value from the result of analyzing r_k. We give this result after proving the following theorem.

Theorem 4.3.3. *Suppose that the whole space H is an N-dimensional Euclidean space $\mathbb{R}^N$. Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 4.3.1 and suppose that Assumptions 4.3.1 and 4.2.1 hold. Assume that $f_\star < f(x_k^\star)$ holds. Then,*

$$f(x_k^\star) - f_\star \leq L r_k^\beta$$

holds for all $k \in \mathbb{N}$.

Proof. Fix $k \in \mathbb{N}$ and $x^\star \in X^\star$ arbitrarily. Furthermore, fix $j \in \mathbb{N}$ arbitrarily. The complement of the slice $\mathrm{lev}_{<f(x_k^\star)} f$, i.e., the set $\{x \in \mathbb{R}^N : f(x_k^\star) \leq f(x)\}$, is nonempty because $x_k^\star$ obviously belongs to it. Therefore, from the definition of r_k, there exists a point $v_j \in \mathbb{R}^N$ such that

$$v_j \notin \mathrm{lev}_{<f(x_k^\star)} f \text{ and } \|x^\star - v_j\| \leq r_k + 1/j$$

hold. The assumption $f_\star < f(x_k^\star)$ implies that $x^\star \in \mathrm{lev}_{<f(x_k^\star)} f$, and the above discussion obtained the property $v_j \notin \mathrm{lev}_{<f(x_k^\star)} f$. Therefore, by proposition 4.2.1, a number $\alpha_j \in [0,1]$ exists such that $w_j := x^\star + \alpha_j(v_j - x^\star) \in \mathrm{bd}(\mathrm{lev}_{<f(x_k^\star)} f)$. Let us consider the lower and upper bounds of the norm $\|x^\star - w_j\|$. Here, the fact $w_j \notin \mathrm{lev}_{<f(x_k^\star)} f$ from the continuity of the objective functional f implies that $r_k \leq \|x^\star - w_j\|$ holds. The upper bound of the norm $\|x^\star - w_j\|$ is thus

$$\begin{aligned}
\|x^\star - w_j\| &= \|x^\star - (x^\star + \alpha_j(v_j - x^\star))\| \\
&= \alpha_j \|x^\star - v_j\| \\
&\leq \|x^\star - v_j\| \\
&\leq r_k + 1/j.
\end{aligned}$$

The sequence $\{w_j\}$ is bounded, since $\|w_j\| \leq \|x^\star\| + \|x^\star - w_j\| \leq \|x^\star\| + r_k + 1$ for all $j \in \mathbb{N}$. Therefore, together with the closedness of the boundary $\mathrm{bd}(\mathrm{lev}_{<f(x_k^\star)} f)$, there exists a subsequence $\{w_{j_t}\} \subset \{w_j\}$ and a point $u_k \in \mathrm{bd}(\mathrm{lev}_{<f(x_k^\star)} f)$ such that w_{j_t} converges to u_k. The continuity of $\|\cdot\|$ leads us to the finding that

$$\begin{aligned}
\|x^\star - u_k\| &= \lim_{l \to \infty} \|x^\star - w_{j_l}\| \\
&= r_k
\end{aligned}$$

due to the previous confirmation of the boundedness of the norm $\|x^\star - w_{j_t}\|$; i.e., $r_k \leq \|x^\star - w_{j_t}\| \leq r_k + 1/j_t$ holds for any $t \in \mathbb{N}$. Furthermore, the continuity of the objective functional f ensures the coincidence $f(u_k) = f(x_k^\star)$. Now, Assumption (A4)

guarantees that the functional f satisfies the Hölder condition with degree $\beta > 0$ at the point $x^\star$ on the set $\mathrm{cl}(\mathrm{lev}_{<f(x_k)} f)$. Therefore, we have

$$\begin{aligned} f(x_k^\star) - f_\star &= f(u_k) - f_\star \\ &\leq L \left\| x^\star - u_k \right\|^\beta \\ &= L r_k^\beta. \end{aligned}$$

This completes the proof. □

This theorem directly induces the following corollary giving the convergence rate in terms of the objective functional when the diminishing step-size rule is adopted.

Corollary 4.3.1. *Suppose that the whole space H is an N-dimensional Euclidean space $\mathbb{R}^N$ and the assumptions in Theorem 4.3.2 hold. Let c be a positive real number and assume that $v_k = c/k$ for all $k \in \mathbb{N}$. Then, a number $k_0 \in \mathbb{N}$ exists such that*

$$f(x_k^\star) - f_\star \leq L \left(\frac{\left\| x_{k_0} - x^\star \right\|^2 + 2c^2(1 - \liminf_{j\to\infty} \alpha_j/2)}{2c(1 - \limsup_{j\to\infty} \alpha_j)(\log(k+1) - \log(k_0))} \right)^\beta$$

holds for all $k \geq k_0$, in other words,

$$f(x_k^\star) - f_\star = \mathcal{O}\left(\frac{1}{(\log(k+1))^\beta} \right).$$

In addition,

$$f(x_k^\star) - f_\star \leq L \left(\frac{\left\| x_1 - x^\star \right\|^2 + 2c^2(1-\alpha)}{2c(1-\alpha)\log(k+1)} \right)^\beta$$

holds for all $k \in \mathbb{N}$ when the sequence $\{\alpha_k\}$ satisfies $\alpha_k = \alpha \in (0,1)$ for all $k \in \mathbb{N}$.

Proof. This is an immediate consequence of Proposition 4.3.2 and Theorem 4.3.3. □

4.3.3 Distance to the fixed point set

We thus far have discussed the convergence rate of Algorithm 4.3.1 in terms of the value of the objective functional. This subsection shows another convergence rate analysis of Algorithm 4.3.1, namely in terms of the distance to the fixed point set. The following theorem gives the convergence rate in terms of the distance to the fixed point set with respect to the averaged norm.

Theorem 4.3.4. *Suppose that the assumptions in Theorem 4.3.2 hold. If $f_\star < f(x_k)$ for all $k \in \mathbb{N}$, then*

$$\frac{1}{k} \sum_{j=1}^{k} \left\| x_j - T(x_j) \right\| = \mathcal{O}(1/k).$$

Proof. Fix $k \in \mathbb{N}$ arbitrarily. Using Proposition 4.2.2 and the fact that P_D is a nonexpansive mapping and $x^\star$ is its fixed point, we have

$$\begin{aligned}\|x_{k+1} - x^\star\|^2 &= \|P_D(\alpha_k x_k + (1-\alpha_k)T(x_k - v_k g_k)) - P_D(x^\star)\|^2 \\ &\leq \|\alpha_k(x_k - x^\star) + (1-\alpha_k)(T(x_k - v_k g_k) - x^\star)\|^2 \\ &= \alpha_k \|x_k - x^\star\|^2 + (1-\alpha_k)\|T(x_k - v_k g_k) - x^\star\|^2 \\ &\quad - \alpha_k(1-\alpha_k)\|x_k - T(x_k - v_k g_k)\|^2 .\end{aligned}$$

Here, $x^\star$ is also a fixed point of the nonexpansive mapping T. Therefore, we can expand the second term of the above expression as follows:

$$\begin{aligned}\|T(x_k - v_k g_k) - x^\star\|^2 &= \|T(x_k - v_k g_k) - T(x^\star)\|^2 \\ &\leq \|x_k - x^\star - v_k g_k\|^2 \\ &= \|x_k - x^\star\|^2 - 2v_k \langle x_k - x^\star, g_k\rangle + v_k^2.\end{aligned}$$

Now, the assumption $f_\star < f(x_k)$ ensures that $-\langle x_k - x^\star, g_k\rangle \leq 0$ holds because g_k is a normal vector of the slice $\mathrm{lev}_{<f(x_k)} f$ at x_k. Hence, we have

$$\|T(x_k - v_k g_k) - x^\star\|^2 = \|x_k - x^\star\|^2 + v_k^2.$$

Overall, we have

$$\begin{aligned}\|x_{k+1} - x^\star\|^2 &\leq \alpha_k \|x_k - x^\star\|^2 + (1-\alpha_k)(\|x_k - x^\star\|^2 + v_k^2) \\ &\quad - \alpha_k(1-\alpha_k)\|x_k - T(x_k - v_k g_k)\|^2 \\ &= \|x_k - x^\star\|^2 - \alpha_k(1-\alpha_k)\|x_k - T(x_k - v_k g_k)\|^2 + (1-\alpha_k)v_k^2.\end{aligned}$$

Assumption (A6) guarantees that a number $k_0 \in \mathbb{N}$ exists such that $\liminf_{j\to\infty} \alpha_j/2 < \alpha_k < (1 + \limsup_{j\to\infty} \alpha_j)/2$ holds for all $k \geq k_0$. Note that Assumption (A6) also ensures that $0 < \liminf_{j\to\infty} \alpha_k/2$ and $(1+\limsup_{j\to\infty} \alpha_j)/2 < 1$. Therefore, together with the above inequality, we have

$$\begin{aligned}\|x_{k+1} - x^\star\|^2 &\leq \|x_k - x^\star\|^2 + \left(1 - \frac{1}{2}\liminf_{j\to\infty} \alpha_j\right) v_k^2 \\ &\quad - \frac{1}{4}\left(\liminf_{j\to\infty} \alpha_j\right)\left(1 - \limsup_{j\to\infty} \alpha_j\right)\|x_k - T(x_k - v_k g_k)\|^2 \\ &\leq \|x_{k_0} - x^\star\|^2 + \left(1 - \frac{1}{2}\liminf_{j\to\infty} \alpha_j\right)\sum_{j=k_0}^{k} v_j^2 \\ &\quad - \frac{1}{4}\left(\liminf_{j\to\infty} \alpha_j\right)\left(1 - \limsup_{j\to\infty} \alpha_j\right)\sum_{j=k_0}^{k}\|x_j - T(x_j - v_j g_j)\|^2\end{aligned}$$

if $k \geq k_0$. This inequality implies that

$$\begin{aligned}
&\sum_{j=1}^{k} \|x_j - T(x_j - v_j g_j)\|^2 \\
&\leq \sum_{j=1}^{k_0-1} \|x_j - T(x_j - v_j g_j)\|^2 \\
&\quad + 4 \left(\liminf_{j\to\infty} \alpha_j\right)^{-1} \left(1 - \limsup_{j\to\infty} \alpha_j\right)^{-1} \left(\|x_{k_0} - x^\star\|^2 + \left(1 - \frac{1}{2}\liminf_{j\to\infty} \alpha_j\right) \sum_{j=k_0}^{k} v_j^2\right)
\end{aligned}$$

holds[*1]. Here, we assume that $\sum_{j=1}^{\infty} v_j^2$ converges. Therefore, the right side of the above inequality is bounded from above with respect to k. This implies that the sequence $\{\sum_{j=1}^{k} \|x_j - T(x_j - v_j g_j)\|^2\}$ is also bounded from above. Let $M \in \mathbb{R}$ denote an upper bound of this sequence.

Let us estimate the distance before and after applying the nonexpansive mapping to each approximation x_k. Using the parallelogram law and the nonexpansivity of T, we obtain

$$\begin{aligned}
\|x_k - T(x_k)\|^2 &= \|x_k - T(x_k - v_k g_k) + T(x_k - v_k g_k) - T(x_k)\|^2 \\
&\leq 2\|x_k - T(x_k - v_k g_k)\|^2 + 2\|T(x_k - v_k g_k) - T(x_k)\|^2 \\
&\leq 2\|x_k - T(x_k - v_k g_k)\|^2 + 2v_k^2.
\end{aligned}$$

Summing the above inequalities with respect to k and dividing both sides by k, we get

$$\begin{aligned}
\frac{1}{k}\sum_{j=1}^{k} \|x_j - T(x_j)\|^2 &\leq \frac{1}{k}\left(2\sum_{j=1}^{k} \|x_j - T(x_j - v_j g_j)\|^2 + 2\sum_{j=1}^{k} v_j^2\right) \\
&\leq \frac{1}{k}\left(2M + 2\sum_{j=1}^{\infty} v_j^2\right).
\end{aligned}$$

This completes the proof. □

4.3.4 Finite convergence

We discussed two convergence analyses for Algorithm 4.3.1 with the diminishing step-size rule. By placing assumptions on the problem to be solved, we can also prove finite convergence. The following proposition describes the requirements to obtain an optimal solution in a finite number of iterations.

[*1] If k is less than k_0, we consider $\sum_{j=k_0}^{k} v_j^2 = 0$ here.

Proposition 4.3.3. *Let $\{x_k\} \subset H$ be a sequence generated by Algorithm 4.3.1, and suppose that the assumptions in Theorem 4.3.2 hold. Furthermore, assume that $X^\star$ has a nonempty interior and the sequence $\{x_k\}$ is contained inside* $\mathrm{Fix}(T)$*. Then, $x_k \in X^\star$ for some $k \in \mathbb{N}$.*

Proof. We will proceed by way of contradiction and suppose that the conclusion does not hold, that is, $f_\star < f(x_k)$ for all $k \in \mathbb{N}$. Fix $x^\star \in X^\star$ arbitrarily. We deduce the following from inequality (4.2):

$$\begin{aligned}\|x_{k+1} - x^\star\|^2 &\leq \|x_k - x^\star\|^2 - 2v_k(1-\alpha_k)\langle g_k, x_k - x^\star\rangle + (1-\alpha_k)v_k^2 \\ &\leq \|x_1 - x^\star\|^2 - 2\sum_{j=1}^{k} v_j(1-\alpha_j)\langle g_j, x_j - x^\star\rangle + \sum_{j=1}^{k}(1-\alpha_j)v_j^2 \\ &\leq \|x_1 - x^\star\|^2 - 2\left(\min_{j=1,2,\ldots,k}\langle g_j, x_j - x^\star\rangle\right)\sum_{j=1}^{k}(1-\alpha_j)v_j + \sum_{j=1}^{k} v_j^2\end{aligned}$$

for all $k \in \mathbb{N}$. The nonnegativity of the left side of the above inequality ensures that

$$\min_{j=1,2,\ldots,k}\langle g_j, x_j - x^\star\rangle \leq \frac{\|x_1 - x^\star\|^2}{2\sum_{j=1}^{k}(1-\alpha_j)v_j} + \frac{\sum_{j=1}^{k} v_j^2}{2\sum_{j=1}^{k}(1-\alpha_j)v_j} \tag{4.3}$$

for all $k \in \mathbb{N}$. Now, there exists a positive real $\delta > 0$ which satisfies $\delta\mathbf{B} + x^\star \subset X^\star$ because $X^\star$ has a nonempty interior. Therefore, $\delta \leq \delta + \langle g_k, x_k - (x^\star + \delta g_k)\rangle = \langle g_k, x_k - x^\star\rangle$ holds for any $k \in \mathbb{N}$. This property with inequality (4.3) implies

$$\begin{aligned}0 < \delta &\leq \min_{j=1,2,\ldots,k}\langle g_j, x_j - x^\star\rangle \\ &\leq \frac{\|x_1 - x^\star\|^2}{2\sum_{j=1}^{k}(1-\alpha_j)v_j} + \frac{\sum_{j=1}^{k} v_j^2}{2\sum_{j=1}^{k}(1-\alpha_j)v_j}\end{aligned}$$

for all $k \in \mathbb{N}$. However, both terms of the right side of the above inequality converge to zero since $\limsup_{k\to\infty}\alpha_k < 1$, $\sum_{j=1}^{\infty} v_j = \infty$ and $\sum_{j=1}^{\infty} v_j^2 < \infty$. Therefore, we arrive at a contradiction. This completes the proof. □

The nonemptiness of the interior of minima appears in many interesting applications, such as surrogate relaxation of discrete programming problems [16, 33]. When we construct a nonexpansive mapping that transforms a given point into a fixed point of itself (an example of such a mapping is a metric projection, but notice that the assumption of this sentence is not limited to it) and give a fixed point of the mapping as the initial point to the algorithm, the generated sequence is contained within the fixed point set of the mapping due to its convexity. Therefore, Proposition 4.3.3 can be applied to these situations.

Section 4.4. Conclusion

By introducing an indicator computed using the radius of the level set, the rate of convergence in terms of the value of the objective functional can be estimated. Not only the value but also the rate of convergence in terms of the distance to the constraint set can be guaranteed by the theorem proved in this chapter. The finite convergence theorem provides a sufficient condition for the algorithm to terminate in a finite number of iterations. These results determine the detailed behavior of the fixed point quasiconvex subgradient method.

Acknowledgments

The author would like to express his sincere appreciation to Professor Hideaki Iiduka and Professor Wataru Takahashi for their valuable advice and constant encouragement during the preparation of this thesis. The author also would like to thank Professor Yoichi Hayashi, Professor Hisao Tamaki and Professor Yukihiro Iguchi for their valuable advice and supports.

This work was supported by the Japan Society for the Promotion of Science (JSPS KAKENHI Grant Number JP17J09220). The author was trained in highly developed information technology at the Security and Programming Camp 2011 held by the Information-technology Promotion Agency Japan (IPA), the Ministry of Economy, Trade and Industry (METI), and the Ministry of Education, Culture, Sports, Science and Technology (MEXT). The author would like to thank these public organizations for their supports.

Bibliography

[1] D. Aussel. New developments in quasiconvex optimization. In S. A. R. Al-Mezel, F. R. M. Al-Solamy, and Q. H. Ansari, editors, *Fixed Point Theory, Variational Analysis, and Optimization*, chapter 5, pages 139–169. Chapman and Hall/CRC, 2014.

[2] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer International Publishing, second edition, 2017.

[3] C. Beltran and F. J. Heredia. An effective line search for the subgradient method. *Journal of Optimization Theory and Applications*, 125(1):1–18, 2005.

[4] V. Berinde. *Iterative Approximation of Fixed Points*, volume 1912 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin Heidelberg, 2007.

[5] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[6] L. Bottou. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*, Nimes, France, 1991. EC2.

[7] S. P. Bradley and S. C. Frey. Fractional programming with homogeneous functions. *Operations Research*, 22(2):350–357, 1974.

[8] Y. Censor and A. Segal. Algorithms for the quasiconvex feasibility problem. *Journal of Computational and Applied Mathematics*, 185(1):34–50, 2006.

[9] Y. Cheung and J. Lou. Proximal average approximated incremental gradient descent for composite penalty regularized empirical risk minimization. *Machine Learning*, 106(4):595–622, 2017.

[10] P. L. Combettes. A block-iterative surrogate constraint splitting method for quadratic signal recovery. *IEEE Transactions on Signal Processing*, 51(7):1771–1782, 2003.

[11] P. L. Combettes and P. Bondon. Hard-constrained inconsistent signal feasibility problems. *IEEE Transactions on Signal Processing*, 47(9):2460–2468, 1999.

[12] P. L. Combettes and J. C. Pesquet. A douglas–rachford splitting approach to nonsmooth convex variational signal recovery. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):564–574, 2007.

[13] N. Cristianini, J. Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[14] J. Y. B. Cruz and W. D. Oliveira. On weak and strong convergence of the projected gradient method for convex optimization in real Hilbert spaces. *Numerical Functional Analysis and Optimization*, 37(2):129–144, 2016.

[15] D. Dheeru and E. K. Taniskidou. UCI machine learning repository, 2017.

[16] M. E. Dyer. Calculating surrogate constraints. *Mathematical Programming*, 19(1):255–278, 1980.

[17] K. Fujiwara, K. Hishinuma, and H. Iiduka. Evaluation of stochastic approximation algorithm and variants for learning support vector machines. *Linear and Nonlinear Analysis*, 4(1):29–61, 2018.

[18] H. Greenberg and W. Pierskalla. Quasi-conjugate functions and surrogate duality. *Cahiers Centre Études Recherche Opér*, 15:437–448, 1973.

[19] N. Hadjisavvas. Convexity, generalized convexity, and applications. In S. A. R. Al-Mezel, F. R. M. Al-Solamy, and Q. H. Ansari, editors, *Fixed Point Theory, Variational Analysis, and Optimization*, chapter 4, pages 139–169. Chapman and Hall/CRC, 2014.

[20] B. Halpern. Fixed points of nonexpansive maps. *Bulletin of the American Mathematical Society*, 73:957–961, 1967.

[21] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, second edition, 1988.

[22] W. L. Hare and Y. Lucet. Derivative-free optimization via proximal point methods. *Journal of Optimization Theory and Applications*, 160(1):204–220, 2014.

[23] Y. Hayashi and H. Iiduka. Optimality and convergence for convex ensemble learning with sparsity and diversity based on fixed point optimization. *Neurocomputing*, 273:367–372, 2018.

[24] K. Hishinuma and H. Iiduka. Parallel subgradient method for nonsmooth convex optimization with a simple constraint. *Linear and Nonlinear Analysis*, 1(1):67–77, 2015.

[25] K. Hishinuma and H. Iiduka. Convergence property, computational performance, and usability of fixed point quasiconvex subgradient method. the 6th Asian Conference on Nonlinear Analysis and Optimization (Oral), 2017.

[26] K. Hishinuma and H. Iiduka. Flexible stepsize selection of subgradient methods for constrained convex optimization. the 10th Anniversary Conference on Nonlinear Analysis and Convex Analysis (Oral), 2017.

[27] K. Hishinuma and H. Iiduka. Iterative method for solving constrained quasiconvex optimization problems based on the Krasnosel'skiĭ-Mann fixed point approximation method. RIMS Workshop on Nonlinear Analysis and Convex Analysis (Oral), 2017.

[28] K. Hishinuma and H. Iiduka. Convergence analysis of incremental and parallel line search subgradient methods in Hilbert space. *Journal of Nonlinear and Convex Analysis*, 20(9):1937–1947, 2019.

[29] K. Hishinuma and H. Iiduka. Convergence rate analyses of fixed point quasiconvex subgradient method. Joint Conference NACA-ICOTA2019: International Conference on Nonlinear Analysis and Convex Analysis, International Conference on Optimization: Techniques and Applications (Oral), 2019.

[30] K. Hishinuma and H. Iiduka. Incremental and parallel machine learning algorithms with automated learning rate adjustments. *Frontiers in Robotics and AI*, 6:77, 2019.

[31] K. Hishinuma and H. Iiduka. Fixed point quasiconvex subgradient method. *European Journal of Operational Research*, 282(2):428–437, 2020.

[32] K. Hishinuma and H. Iiduka. Supplementary data S1 for the article entitled "fixed point quasiconvex subgradient method". `https://doi.org/10.1016/j.ejor.2019.09.037`, 2020.

[33] Y. Hu, X. Yang, and C.-K. Sim. Inexact subgradient methods for quasi-convex optimization problems. *European Journal of Operational Research*, 240(2):315–327, 2015.

[34] Y. Hu, C. K. W. Yu, and C. Li. Stochastic subgradient method for quasi-convex optimization problems. *Journal of Nonlinear and Convex Analysis*, 17(4):711–724, 2016.

[35] Y. Hu, C. K. W. Yu, C. Li, and X. Yang. Conditional subgradient methods for constrained quasi-convex optimization problems. *Journal of Nonlinear and Convex Analysis*, 17(10):2143–2158, 2016.

[36] H. Iiduka. Iterative algorithm for solving triple-hierarchical constrained optimization problem. *Journal of Optimization Theory and Applications*, 148(3):580–592, 2011.

[37] H. Iiduka. Fixed point optimization algorithm and its application to power control in CDMA data networks. *Mathematical Programming*, 133(1):227–242, 2012.

[38] H. Iiduka. Iterative algorithm for triple-hierarchical constrained nonconvex optimization problem and its application to network bandwidth allocation. *SIAM Journal on Optimization*, 22(3):862–878, 2012.

[39] H. Iiduka. Fixed point optimization algorithms for distributed optimization in networked systems. *SIAM Journal on Optimization*, 23(1):1–26, 2013.

[40] H. Iiduka. Acceleration method for convex optimization over the fixed point set of a nonexpansive mapping. *Mathematical Programming*, 149(1):131–165, 2015.

[41] H. Iiduka. Parallel computing subgradient method for nonsmooth convex optimization over the intersection of fixed point sets of nonexpansive mappings. *Fixed Point Theory and Applications*, 2015:72, 2015.

[42] H. Iiduka. Convergence analysis of iterative methods for nonsmooth convex optimization over fixed point sets of quasi-nonexpansive mappings. *Mathematical Programming*, 159(1):509–538, 2016.

[43] H. Iiduka. Incremental subgradient method for nonsmooth convex optimization with fixed point constraints. *Optimization Methods and Software*, 31(5):931–951, 2016.

[44] H. Iiduka. Line search fixed point algorithms based on nonlinear conjugate gradient directions: Application to constrained smooth convex optimization. *Fixed Point Theory and Applications*, 2016:77, 2016.

[45] H. Iiduka. Proximal point algorithms for nonsmooth convex optimization with fixed point constraints. *European Journal of Operational Research*, 253(2):503–513, 2016.

[46] H. Iiduka. Almost sure convergence of random projected proximal and subgradient algorithms for distributed nonsmooth convex optimization. *Optimization*, 66(1):35–59, 2017.

[47] H. Iiduka. Distributed optimization for network resource allocation with non-smooth utility functions. *IEEE Transactions on Control of Network Systems*, 6(4):1354–1365, 2018.

[48] H. Iiduka. Stochastic fixed point optimization algorithm for classifier ensemble. *IEEE Transactions on Cybernetics*, pages 1–11, 2019.

[49] H. Iiduka and K. Hishinuma. Acceleration method combining broadcast and incremental distributed optimization algorithms. *SIAM Journal on Optimization*, 24(4):1840–1863, 2014.

[50] H. Iiduka and M. Uchida. Fixed point optimization algorithms for network bandwidth allocation problems with compoundable constraints. *IEEE Communications Letters*, 15(6):596–598, 2011.

[51] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed Aug. 16, 2018].

[52] F. Kelly. Charging and rate control for elastic traffic. *European transactions on Telecommunications*, 8(1):33–37, 1997.

[53] K. C. Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical Programming*, 90(1):1–25, 2001.

[54] I. V. Konnov. On convergence properties of a subgradient method. *Optimization Methods and Software*, 18(1):53–62, 2003.

[55] M. A. Krasnosel'skiĭ. Two remarks on the method of successive approximations. *Uspekhi Matematicheskikh Nauk*, 10(1(63)):123–127, 1955.

[56] T. Larsson, M. Patriksson, and A.-B. Strömberg. Conditional subgradient optimization—theory and applications. *European Journal of Operational Research*, 88(2):382–403, 1996.

[57] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[58] Y. LeCun, C. Cortes, and C. J. C. Burges. The mnist database of handwritten digits, 1998.

[59] C.-Y. Lee, A. L. Johnson, E. Moreno-Centeno, and T. Kuosmanen. A more efficient algorithm for convex nonparametric least squares. *European Journal of Operational Research*, 227(2):391–400, 2013.

[60] E. Leopold and J. Kindermann. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1):423–444, 2002.

[61] C.-J. Lin. LIBSVM data: Classification, regression, and multi-label, 2017.

[62] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46(1):191–202, 2002.

[63] W. R. Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4:506–510, 1953.

[64] M. Meiss, F. Menczer, S. Fortunato, A. Flammini, and A. Vespignani. Ranking web sites with real user traffic. In *Proc. First ACM International Conference on Web Search and Data Mining (WSDM)*, pages 65–75, 2008.

[65] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard, Version 3.1.* High-Performance Computing Center Stuttgart, 2015.

[66] A. Nedić and D. Bertsekas. *Convergence Rate of Incremental Subgradient Algorithms*, pages 223–264. Springer US, Boston, MA, 2001.

[67] A. Nedić and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

[68] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag New York, second edition, 2006.

[69] T. Oliphant. *A guide to NumPy*. Trelgol Publishing, USA, 2006.

[70] Z. Opial. Weak convergence of the sequence of successive approximations for nonexpansive mappings. *Bulletin of the American Mathematical Society*, 73(4):591–597, 1967.

[71] P. S. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, 1996.

[72] C. Pan, C. Yin, N. C. Beaulieu, and J. Yu. Distributed resource allocation in sdcn-based heterogeneous networks utilizing licensed and unlicensed bands. *IEEE Transactions on Wireless Communications*, 17(2):711–721, 2018.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[74] J.-P. Penot. Are generalized derivatives useful for generalized convex functions? In J.-P. Crouzeix, J.-E. Martinez-Legaz, and M. Volle, editors, *Generalized Convexity, Generalized Monotonicity: Recent Results, Nonconvex Optimization and Its Applications*, volume 27, pages 3–59. Kluwer Academic Publishers, 1998.

[75] F. Plastria. Lower subdifferentiable functions and their minimization by cutting planes. *Journal of Optimization Theory and Applications*, 46(1):37–53, 1985.

[76] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, 1998.

[77] B. T. Polyak. Introduction to optimization. translation series in mathematics and engineering. *Optimization Software*, 1987.

[78] S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. Support vector learning for semantic argument classification. *Machine Learning*, 60(1):11–39, 2005.

[79] S. Raschka. *Python machine learning*. Packt Publishing Ltd, 2015.

[80] R. T. Rockafellar. Monotone operators associated with saddle-functions and minimax problems. *Nonlinear functional analysis*, 18(I):397–407, 1970.

[81] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[82] K. Sakurai and H. Iiduka. Acceleration of the Halpern algorithm to search for a fixed point of a nonexpansive mapping. *Fixed Point Theory and Applications*, 2014(202), 2014.

[83] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

[84] K. Shimizu, K. Hishinuma, and H. Iiduka. Parallel computing proximal method for nonsmooth convex optimization with fixed point constraints of quasi-

nonexpansive mappings. *Applied Set-Valued Analysis and Optimization (accepted)*.

[85] K. Slavakis and I. Yamada. Robust wideband beamforming by the hybrid steepest descent method. *IEEE Transactions on Signal Processing*, 55(9):4511–4522, 2007.

[86] I. M. Stancu-Minasian. *Fractional Programming: Theory, Methods and Applications*. Kluwer Academic Publishers, 1997.

[87] W. Takahashi. *Introduction to Nonlinear and Convex Analysis*. Yokohama Publishers, Inc., Yokohama, 2009.

[88] T. X. Tran and D. Pompili. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology*, 68(1):856–868, 2019.

[89] W. F. Trench. *Introduction to Real Analysis*. Pearson Education, 2003.

[90] O. Tutsoy and M. Brown. An analysis of value function learning with piecewise linear control. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(3):529–545, 2016.

[91] O. Tutsoy and M. Brown. Reinforcement learning analysis for a minimum time balance problem. *Transactions of the Institute of Measurement and Control*, 38(10):1186–1200, 2016.

[92] P. Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

[93] I. Yamada. The hybrid steepest descent method for the variational inequality problem over the intersection of fixed point sets of nonexpansive mappings. In D. Butnariu, Y. Censor, and S. Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*, volume 8 of *Studies in Computational Mathematics*, pages 473–504. Elsevier, 2001.

[94] I. Yamada and N. Ogura. Hybrid steepest descent method for variational inequality problem over the fixed point set of certain quasi-nonexpansive mappings. *Numerical Functional Analysis and Optimization*, 25(7-8):619–655, 2005.

[95] G. Yuan, Z. Meng, and Y. Li. A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations. *Journal of Optimization Theory and Applications*, 168(1):129–152, 2016.

[96] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11:1081–1107, 2010.

Manuscript received November 15th, 2019
revised February 7th, 2020