SCIENCE CHINA
Mathematics

Special Topic on AI for Math and Math for AI
• ARTICLES •

https://doi.org/10.1007/s11425-025-2516-2

Formalization of algorithms for optimization with block structures

Chenyi Li[1], Zichen Wang[1], Yifan Bai[2], Yunxi Duan[3], Yuqing Gao[4], Pengfei Hao[5] & Zaiwen Wen[6,*]

[1]*School of Mathematical Sciences, Peking University, Beijing* 100871*, China;*
[2]*School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[3]*School of Mathematics, Statistics and Mechanics, Beijing University of Technology, Beijing* 100124*, China;*
[4]*Institut für Numerische Simulation, Universität Bonn, Bonn* 53115*, Germany;*
[5]*School of Mathematical Sciences, Beijing Normal University, Beijing* 100875*, China;*
[6]*Beijing International Center for Mathematical Research, Peking University, Beijing* 100871*, China*

Email: lichenyi@stu.pku.edu.cn, princhernwang@gmail.com, yifanbai@sjtu.edu.cn, yunxiduan@emails.bjut.edu.cn, gao.yuqing@uni-bonn.de, 202111998147*@mail.bnu.edu.cn, wenzw@pku.edu.cn*

Received July 11, 2025; accepted November 20, 2025; published online April 14, 2026

Abstract Block-structured problems are central to advances in numerical optimization and machine learning. In this paper, we provide the formalization of convergence analysis for two pivotal algorithms in such settings: the block coordinate descent (BCD) method and the alternating direction method of multipliers (ADMM). Utilizing the type-theory-based proof assistant Lean 4, we develop a rigorous framework to formally represent these algorithms. Essential concepts in nonsmooth and nonconvex optimization are formalized, notably subdifferentials, which extend the classical differentiability to handle nonsmooth scenarios, and the Kurdyka-Łojasiewicz (KL) property, which provides essential tools to analyze convergence in nonconvex settings. Such definitions and properties are crucial for the corresponding convergence analyses. We formalize the convergence proofs of these algorithms, demonstrating that our definitions and structures are coherent and robust. These formalizations lay a basis for analyzing the convergence of more general optimization algorithms. Our implementation is available at https://github.com/optsuite/optlib.

Keywords numerical optimization, Lean, the block coordinate descent (BCD) method, the alternating direction method of multipliers (ADMM)

MSC(2020) 03B35, 68V20, 65K10, 90C26

Citation: Li C Y, Wang Z C, Bai Y F, et al. Formalization of algorithms for optimization with block structures. Sci China Math, 2026, 69, https://doi.org/10.1007/s11425-025-2516-2

1 Introduction

Large-scale optimization problems with block structures play a central role in numerous fields, including machine learning, signal processing, and image processing. Efficiently solving these problems is essential for theoretical advancement and practical applications. Two prominent algorithms addressing these challenges are the block coordinate descent (BCD) method and the alternating direction method of multipliers (ADMM). The BCD algorithm solves optimization problems with objective functions

* Corresponding author

© Science China Press 2026

comprising both separable and nonseparable components by updating the variable blocks by blocks. The ADMM algorithm primarily addresses optimization problems involving multi-blocks connected with coupling constraints, iteratively minimizing the augmented Lagrangian function for each block and updating the multipliers to enforce constraint satisfaction. Despite their methodological differences, both approaches share the fundamental strategy of decomposing complex optimization problems into simpler subproblems.

The convergence analysis of algorithms for optimization problems with block structures has been widely studied. For the BCD method, early foundational work dates back to Ortega and Rheinboldt [32]. Powell [34] demonstrated the challenges of iterative nonconvex optimization through examples of non-convergence in nonconvex objective functions. Tseng [38] proposed a robust convergence framework for nondifferentiable convex minimization problems. To address these complexities in nonsmooth and nonconvex analysis, additional conditions, such as the Kurdyka-Łojasiewicz (KL) property, become necessary. The KL property can be traced back to Łojasiewicz's work on gradient inequalities [28]. The convergence of the BCD algorithm in nonconvex settings with the KL property is given by [7]. The ADMM method also gains significant attention due to its applicability in high-dimensional data problems and distributed optimization [8]. The classical ADMM method is introduced in [16, 19]. Glowinski [18] provided an introduction to the historical development. The convergence analysis of the ADMM algorithm under certain conditions is established in [15–17]. Fankel et al. [14] gave the convergence proof of a computationally more attractive large step-length version of the ADMM method.

Formal verification of numerical algorithms has seen significant advances across the Coq [20], Isabelle/HOL [31], and Lean [11] proof assistants. Progress is evident across several algorithmic domains. In the verification of ordinary differential equation (ODE) solvers, recent Coq contributions include verified solvers [25] and formal Runge-Kutta error bounds [13]. Concurrently, the Isabelle/HOL-analysis ecosystem has underpinned extensive developments, including mechanized error analyses for first-order methods [23], verified reachability using adaptive Runge-Kutta and affine arithmetic [21], and a solver validated on the Lorenz attractor [22]; related theories are curated in the archive of formal proofs (AFP) entry on ODEs [24]. For linear solvers, formal work in Coq has verified the convergence of the Jacobi method [35] and explored the asymptotic convergence of stationary iterative methods such as Gauss-Seidel [36]. In the optimization domain, recent Lean developments provide mechanized convergence rates for a spectrum of first-order methods [27], introducing formal definitions of convexity, L-smoothness, and the proximal operator. These domain-specific achievements are supported by robust foundational libraries and tools for numerical reasoning. In Coq, this infrastructure includes the Coquelicot library for real analysis [5] and Flocq for floating-point arithmetic [6]. These are complemented by proof-producing tools such as Gappa [10] and interval-based automation via CoqInterval [29, 30]. Further strengthening this foundation, recent work has advanced mechanized round-off analysis through frameworks such as VCFloat2 [1] and other formalisms for reasoning about numerical errors [1, 26].

The Lean 4 language and its theorem library MATHLIB [37] offer a flexible formalization tool for mathematicians, enabling precise reasoning about mathematical concepts through a machine-verified framework. Lean is based on the dependent type theory, allowing for the expression of refined mathematical properties directly in the type system. In this setting, proofs are constructed as objects of a specific type, ensuring that every step is logically sound and verifiable. This approach enables the construction of rigorous formalizations across various fields of mathematics. For example, the Gagliardo-Nirenberg-Sobolev inequality was formalized by van Doorn and Macbeth [39]. Dupuis et al. [12] explored the formalization of functional analysis with semilinear maps. Moreover, a formalization of Doob's martingale convergence theorems was provided by Ying and Degenne [40].

We present a formalized convergence analysis of the BCD and ADMM algorithms within the Lean theorem proving environment, building upon the mathematical foundations provided by MATHLIB and [27]. All formalization components, including the definitions, supporting lemmas, and convergence theorems for optimization algorithms with block structures, are newly introduced and verified in this paper. Our main contributions are summarized as follows.

(1) Subdifferential theory in Hilbert spaces. The Fréchet and limiting subdifferentials are formalized,

extending the formalization of the differential structure of nonsmooth functions. Basic theorems are presented, including equivalent definitions, subdifferential calculus, and optimality conditions of nonsmooth unconstrained optimization.

(2) The KL property and KL functions. The formalization of the KL property and KL functions is established. The sufficient conditions of the KL property are formally proved. The uniformized KL property is derived through extensions on compact sets, enabling convergence analysis in nonsmooth and nonconvex optimization problems.

(3) Algorithm frameworks and convergence analysis. (i) A formal framework is provided for the proximal alternating linearized minimization in two-block problems [7]. Formalized convergence theorems such as critical point analysis through subdifferential characterizations and finite-length convergence under the KL property are provided. (ii) A type of the ADMM algorithm is formalized for composite convex optimization problems with a linear constraint connecting two distinct components according to [9, 14]. Essential assumptions, including lower semicontinuity, convexity, and solvability of subproblems, are formally established to ensure systematic convergence analysis. Moreover, the error reduction bounds are formally presented, and global convergence of the ADMM method to Karush-Kuhn-Tucker (KKT) points is proved.

The rest of this paper is organized as follows. The formalized definitions and properties of the subdifferential for general functions are presented in Section 2. The KL property and KL functions are formalized in Section 3. The formalization of the schemes and the corresponding convergence analyses for the BCD and ADMM algorithms are presented in Sections 4 and 5, respectively. Insights from formalization and design principles are discussed in Section 6. The paper is concluded with discussion in Section 7.

2 Subdifferential structure of general functions

2.1 Definitions

In this subsection, we review the definition and properties of the subdifferential and the first-order optimality condition for general functions. These are particularly useful for analyzing nondifferentiable and nonconvex functions. Recall that for a convex function f defined on a Hilbert space E, the subgradient u provides a global lower bound for f, i.e., for all $y \in E$, the inequality $f(y) \geqslant f(x) + \langle u, y - x\rangle$ holds at every point in the domain. However, when f is nonconvex, this property is only expected to hold in a limiting sense. We define the subdifferential as follows.

Definition 2.1. For a function $f : E \to \mathbb{R}$ defined on a Hilbert space E, and a given point $x \in E$, the *Fréchet subdifferential* is defined as

$$\hat{\partial} f(x) = \left\{ u \in \mathbb{R}^n \;\middle|\; \liminf_{y \to x, y \neq x} \frac{f(y) - f(x) - \langle u, y - x\rangle}{\|y - x\|} \geqslant 0 \right\}.$$

For convenience, we introduce the notion of *f-attentive convergence*. The term

$$x^n \underset{f}{\to} x$$

denotes that a sequence $\{x^n\}$ satisfies that $x^n \to x$ with $f(x^n) \to f(x)$ as $n \to \infty$. The limiting subdifferential is defined as follows.

Definition 2.2. The *limiting subdifferential* of f at $x \in E$ is defined as

$$\partial f(x) = \left\{ u \in \mathbb{R}^n \;\middle|\; \exists\, x^k \underset{f}{\to} x, u^k \in \hat{\partial} f(x^k) \to u, \text{ as } k \to \infty \right\}, \tag{2.1}$$

where u is called a (general) subgradient of f at x.

Additional assumptions on the domain space are required to formalize these definitions. We assume the following instances for the space variable from this subsection.

```
variable {E : Type*} [NormedAddCommGroup E] [InnerProductSpace ℝ E]
```

The assumptions in brackets give the properties of the space E, claiming that `E` is a normed additive commutative group and an inner product space on $\mathbb{R}$. These assumptions can be automatically checked when using the related theorems. For example, when working with functions on $\mathbb{R}^n$, Lean recognizes that $\mathbb{R}^n$ is both a normed additive commutative group and an inner product space over $\mathbb{R}$. Hence, the theorems proved for a general space E work well for a concrete space such as $\mathbb{R}^n$.

The Fréchet subdifferential and limiting subdifferential in Lean are defined as follows.

```
def differential_fun (x : E) (f : E → ℝ) (u : E) :=
    fun y ↦ Real.toEReal ((f y - f x - inner u (y - x)) / ‖ y - x ‖)

def f_subdifferential (f : E → ℝ) (x : E) : Set E :=
    {v : E | liminf (differential_fun x f v) (nhds[≠] x) ≥ 0}

def subdifferential (f : E → ℝ) (x : E) : Set E :=
    {v₀ : E | ∃ u : ℕ → E, Tendsto u atTop (nhds x) ∧ Tendsto (fun n ↦ f (u n)) atTop (nhds
    (f x)) ∧ (∃ v : ℕ → E, ∀ n, v n ∈ f_subdifferential f (u n) ∧ Tendsto v atTop (nhds v₀))}
```

The function `Real.toEReal` is introduced to convert real numbers into extended real numbers (`EReal`). This conversion is necessary for correctly applying the operator lim inf in the definition of `f_subdifferential`. Specifically, the limit infimum can attain infinite values only within extended real numbers. In contrast, within Lean's real number type ($\mathbb{R}$), the limit infimum defaults to zero when it would otherwise take an infinite value.

We can define the active domain of a function $f(x)$ using subdifferential, which consists of points whose subdifferential is not empty. The formalized definition is given as follows.

```
def active_domain (f : E → ℝ) : Set E := {x | subdifferential f x ≠ ∅}
```

In the above definitions, the regular subgradient uses the concept of `liminf` in Lean. We can get an equal description using little-o notation through mathematical reformulation:

$$f(y) \geqslant f(x) + \langle u, y - x\rangle + o(\|y - x\|), \quad \text{as } \|y - x\| \to 0.$$

We can reformulate the definition using the concept of a filter to avoid explicitly managing Taylor expansions or infinitesimals, which brings convenience in some cases. An equivalent expression for the Fréchet subdifferential is defined in the following statement.

Theorem 2.3. *For a given $x \in \operatorname{dom} f$, $u \in \hat{\partial} f(x)$ if and only if for any $\varepsilon > 0$ and every y in the neighborhood of x, it holds that*

$$f(y) - f(x) - \langle u, y - x\rangle \geqslant -\varepsilon\|y - x\|.$$

The above theorem is formalized as follows.

```
theorem has_f_subdiff_iff : u ∈ f_subdifferential f x ↔ ∀ ε > 0,
(nhds x).Eventually fun x => f y - f x - inner u (y - x) ≥ -ε * ‖ y - x ‖
```

To state the next theorem, we define

$$G(f, \hat{\partial} f) := \{(x, f(x), u) \mid x \in \mathbb{R}^n, u \in \hat{\partial} f(x)\}, \tag{2.2}$$

$$G(\partial f) := \{(x, u) \mid x \in \mathbb{R}^n, u \in \partial f(x)\}. \tag{2.3}$$

We next introduce a more streamlined description of the subdifferential by leveraging the closure of the subdifferential graph. This reformulation provides an intuitive geometric interpretation by directly relating the subdifferential to the closure of its graph. The following theorem states the closedness of the subdifferential graph. This property is also used in the proof of the KL property.

Theorem 2.4. *Let $(x_k, x_k^*) \in G(\partial f)$ be a sequence that converges to (x, x^*). If $f(x_k)$converges to $f(x)$, then $(x, x^*) \in G(\partial f)$, or equivalently,*

$$\partial f(x) = \{u \in \mathbb{R}^n \mid (x, f(x), u) \in \overline{G(f, \hat{\partial} f)}\}. \tag{2.4}$$

The theorem is formalized as follows.

```
def subdifferential_Graph (f : E → ℝ) :=
{(x, u) : E × E | u ∈ subdifferential f x}

theorem GraphOfSubgradientIsClosed {f : E → ℝ} {xn un : ℕ → E} {x u : E}
(hx : ∀ n , (xn n, un n) ∈ subdifferential_Graph f)
(hconv : Tendsto (fun n => (xn n , un n)) atTop (nhds (x, u)))
(hf : Tendsto (fun n => f (xn n)) atTop (nhds (f x))) :
(x, u) ∈ subdifferential_Graph f
```

2.2 Properties of subdifferentials

These definitions of subdifferentials enable the derivation of the first-order optimality condition of general functions. For general functions, the first-order condition is stated as follows.

Theorem 2.5. *If x^* is a local minimum point of f, then $0 \in \partial f(x^*)$.*

In Lean, we formalize this as follows.

```
theorem first_order_optimality_condition (f : E → ℝ) (x₀ : E)
(hx : IsLocalMin f x₀) : (0 : E) ∈ f_subdifferential f x₀
```

The point y satisfying $0 \in \partial f(y)$ is called a critical point of the function $f(x)$. The set of critical points of $f(x)$ is denoted by crit $f(x)$. This is formalized as follows.

```
def  critical_point (f : E → ℝ) : Set E := {x | 0 ∈ subdifferential f x}
```

It is evident that a local minimum must be a critical point of the objective function.

Notably, both the Fréchet subdifferential and the limiting subdifferential reduce to the gradient of f at points, where the function f is smooth. Several theorems will be introduced to state the equivalence between subdifferential and gradient for differentiable functions. Since the gradient of the function is considered, we additionally assume `[CompleteSpace E]` to align with the assumptions regarding the existence of the gradient. To begin with, we present the following theorem to show that the definition of Fréchet differential is the same as the gradient when the function is differentiable.

Theorem 2.6. *The following two statements are equivalent*:

(1) *f is differentiable at x with a gradient u.*

(2) *$u \in \hat{\partial} f(x)$ and $-u \in \hat{\partial}(-f)(x)$.*

The formalized version can be given as follows.

```
theorem HasGradientAt_iff_f_subdiff : HasGradientAt f u x ↔ u ∈ f_subdifferential f x ∧ -u ∈
    f_subdifferential (-f) x
```

The proof of this proposition mainly utilizes the equivalent transformation of Fréchet subdifferential provided by `has_f_subdiff_iff` in Theorem 2.3, thereby avoiding dealing with the lower limit in the definition of Fréchet subdifferential. The following property states the equivalence between gradient, Fréchet subdifferential, and limiting subdifferential.

Theorem 2.7. *Let $f(x)$ be a differentiable function with gradient $\nabla f(x)$ on a Hilbert space E. Then for every $z \in E$, $\hat{\partial} f(z)$ is a singleton set with the element $\nabla f(z)$.*

The formalization of the theorem is given as follows.

```
theorem f_subdiff_gradient (f' : E → E) (z : E) (hf : HasGradientAt f (f' z) z) :
    f_subdifferential f z = {f' z}
```

We prove this based on Theorem 2.6. The conclusion $\nabla f(z) \in \hat{\partial} f(z)$ is a direct result from the proposition `HasGradientAt_iff_f_subdiff`. The theorem `f_subdiff_neg_f_subdiff_unique` is introduced to establish the connection between the subdifferentials of functions that are negatively correlated. Theorem 2.7 can be deduced from these preparations.

The calculation rules of the subdifferential are also formalized. The following theorems show that under certain conditions, the Fréchet subdifferential of the sum of two functions can be converted into the sum of the Fréchet subdifferentials of the two functions separately.

Theorem 2.8. *Let $f(x)$ and $g(x)$ be two functions defined on E. If either $f(x)$ or $g(x)$ is differentiable, then it holds that $\hat{\partial}(f+g)(x) = \hat{\partial}f(x) + \hat{\partial}g(x)$.*

The formalized version of the theorem is as follows.

```
theorem f_subdiff_add (f g : E → ℝ) (g' : E → E) (x : E) (hg : ∀ (x₁ : E), HasGradientAt g
    (g' x₁) x₁) (z : E) :
    z ∈ f_subdifferential (f + g) x ↔ ∃ zf ∈ f_subdifferential f x, ∃ zg ∈ f_subdifferential
    g x, z = zf + zg
```

With an extra condition that g is continuously differentiable, Theorem 2.8 also holds for the limiting subdifferential.

Theorem 2.9. *Let f and g be two functions defined on E. If either $f(x)$ or $g(x)$ is continuously differentiable on E, it holds that $\partial(f+g) = \partial f + \partial g$.*

The formalized theorem is as follows.

```
theorem subdiff_add (f g : E → ℝ) (g' : E → E) (x : E) (hg : ∀ (x₁ : E), HasGradientAt g (g'
    x₁) x₁) (gradcon : Continuous g') (z : E) : z ∈ subdifferential (f + g) x ↔ ∃ zf ∈
    subdifferential f x, ∃ zg ∈ subdifferential g x, z = zf + zg
```

The proximal operator is also widely used in optimization algorithms, which addresses an optimization problem that balances maintaining proximity to a specified input point x, while also minimizing the function f. For general functions, the proximal operator maps a point to a set as follows.

Definition 2.10. The *proximal operator* for function f at point x is given as

$$\operatorname*{prox}_f(x) = \arg\min_u \left\{ f(u) + \frac{1}{2}\|u-x\|^2 \right\}. \tag{2.5}$$

The formalization of the proximal operator can be referred to [27]. The optimality condition of the optimization problem naturally links the proximal operator with the subdifferential. The connection is described in the following theorem.

Theorem 2.11. *If $f(x)$ is a function from a Hilbert space E to $\mathbb{R}$, it holds that*

$$u \in \operatorname*{prox}_f(x) \quad \textit{if and only if} \quad x - u \in \partial f(u). \tag{2.6}$$

The theorem is formalized as follows.

```
theorem f_subdiff_prox {f : E → ℝ} {x y : E} (h : prox_prop f y x) : y - x ∈
    f_subdifferential f x
```

A key step to prove the above propositions is to utilize the first-order optimality condition proven before to connect the minimality definition of the proximal operator and Fréchet subdifferential. The equivalence and operational properties of Fréchet subdifferential for certain functions also help to simplify the problem.

3 The KL property and KL function

In the past few decades, the KL property has emerged as a versatile analytical framework for investigating convergence rates of first-order optimization methods, particularly when dealing with objectives that exhibit both nonsmoothness and nonconvexity (see [3,4,7]). It is an extension to the Polyak-Łojasiewicz (PL) inequality developed by Polyak [33] and Łojasiewicz [28]: $\|\nabla f(x)\| \geqslant \mu\|f(x) - f(\bar{x})\|^{\frac{1}{2}}$, $\mu > 0$, where f is an L-smooth function, and $\bar{x}$ is a global minimum of f. The linear convergence of the gradient descent algorithm can be established by this well-known inequality. It illustrates that the sub-optimality

can be controlled by the vanishing of gradient norm and a special concave function $\phi(t) = \mu t^{\frac{1}{2}}$. Such functions are the so-called desingularizing functions in the context of the KL property.

Definition 3.1. For any $\eta \in (0, +\infty)$, if ϕ satisfies the following properties:
(1) $\phi : [0, \eta) \to [0, \infty)$ is continuous and concave;
(2) $\phi(0) = 0$;
(3) ϕ is continuously differentiable on $(0, \eta)$ with $\phi' > 0$ on $(0, \eta)$,
then we call ϕ a desingularizing function and denote it by $\phi \in \Phi_\eta$.

The corresponding formalization of this family of functions is as follows.

```
def desingularizing_function (η : ℝ) := {ϕ : ℝ → ℝ | (ConcaveOn ℝ (Ico 0 η) ϕ) ∧ (ϕ 0 = 0) ∧
    (ContDiffOn ℝ 1 ϕ (Ioo 0 η)) ∧ (ContinuousAt ϕ 0) ∧ (∀ x ∈ Ioo 0 η, deriv ϕ x > 0)}
```

These desingularizing functions are essential to quantify the behavior of the objective f near its critical points, enabling the analysis of the relationship between the subdifferential and the function values. We can define the KL property utilizing such functions. Still, all variables are considered in the Hilbert space E.

Definition 3.2. Let $f : E \to \mathbb{R}$ be a lower semicontinuous function. We say that f satisfies the KL property at $u \in \mathrm{dom}\partial f := \{u \in E : \partial f(u) \neq \emptyset\}$ if there exist $\eta \in (0, \infty)$, a neighborhood S of u and $\phi \in \Phi_\eta$ such that for any $x \in S \cap [f(u) < f < f(u) + \eta]$, the inequality $\phi'(f(x) - f(u)) \cdot \mathrm{dist}(0, \partial f(x)) \geqslant 1$ holds. The point u is called a KL point of f. If f satisfies the KL property at every point of $\mathrm{dom}\partial f$, we say that f is a KL function.

We generalize the assumption of lower semicontinuity in Definition 3.2 in formalization. Note that the KL property holds naturally if $\partial f(\bar{u}) = \emptyset$. Hence, the non-emptiness of subdifferential can also be omitted in formalization. These two assumptions need not be included in the formalization of the KL point and KL function. Consequently, $\mathrm{dist}(0, \partial f(u))$ might be infinity when $\partial f(u)$ is empty. We use `EMetric.infEdist` to keep the inequality on extended non-negative real numbers `ENNReal`, and `ENNReal.ofReal` to convert real numbers to `ENNReal`. The formalization of the KL point and KL function is as follows.

```
def KL_point (f : E → ℝ) (u : E) : Prop := ∃ η ∈ Ioi 0, ∃ s ∈ nhds u, ∃ ϕ ∈
    desingularizing_function η, ∀ x ∈ s ∩ {y | f u < f y ∧ f y < f u + η}, (ENNReal.ofReal
    (deriv ϕ (f x - f u))) * (EMetric.infEdist 0 (subdifferential f x)) ≥ ENNReal.ofReal 1

def KL_function (f : E → ℝ) : Prop := ∀ u ∈ active_domain f, KL_point f u
```

It is shown in [2] that the KL property holds at noncritical points of f.

Theorem 3.3. *Suppose that $f : E \to \mathbb{R}$ is a lower semicontinuous function. The KL property holds at x, if it holds that $0 \notin \partial f(x)$.*

This theorem is formalized as follows.

```
theorem KL_property_at_noncritical_point (h_noncrit : 0 ∉ subdifferential f x) : KL_point f x
```

The uniformized KL property [7] is incorporated to deal with the case where the KL property holds at many different points.

Theorem 3.4. *Suppose that Ω is a compact set and $f : E \to \mathbb{R}$ is a lower semicontinuous function. Suppose that f is constant on Ω and satisfies the KL property at every point of Ω. Then there exist $\epsilon > 0, \eta > 0$ and a function $\phi \in \Phi_\eta$, such that for all $u \in \Omega$ and all x satisfying $\mathrm{dist}(x, \Omega) < \epsilon$ and $f(u) < f(x) < f(u) + \eta$, it holds that $\phi'(f(x) - f(u)) \cdot \mathrm{dist}(0, \partial f(x)) \geqslant 1$.*

```
theorem uniformized_KL_property {f : E → ℝ} {Ω : Set E} (h_compact : IsCompact Ω) (h_Ω1 : ∀
    x ∈ Ω, KL_point f x) (h_Ω2 : is_constant_on f Ω) : ∃ ε ∈ Ioi 0, ∃ η ∈ Ioi 0, ∃ ϕ ∈
    desingularizing_function η, ∀ u ∈ Ω , ∀ x ∈ {y : E | (EMetric.infEdist y Ω).toReal < ε} ∩
    {y | f u < f y ∧ f y < f u + η}, (ENNReal.ofReal (deriv ϕ (f x - f u))) * EMetric.infEdist
    0 (subdifferential f x) ≥ 1
```

To prove this theorem, a finite open cover can be chosen based on the neighborhoods of the KL points using `IsCompact.elim_finite_subcover_image`. Then there exists an open ball in this finite cover

from `IsCompact.exists_thickening_subset_open`. Furthermore, one can prove that the sum of the finite desingularizing functions on the finite cover will be in Φ_η for some $\eta > 0$ by checking Definition 3.1. Finally, a uniformized neighborhood and a desingularizing function are constructed for the inequality.

4 The block coordinate descent algorithm

In this section, we consider a special form of the BCD algorithm with proximal alternating linearized minimization following [7]. The BCD algorithm is applied to two-block optimization problems as follows:

$$\min_{x,y} \Psi(x,y) = f(x) + g(y) + H(x,y), \tag{4.1}$$

where $f(x) : \mathbb{R}^n \to \mathbb{R}$ and $g(y) : \mathbb{R}^m \to \mathbb{R}$ are separated parts for x and y, respectively. $H(x,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the composite part. The basic assumptions for this problem are listed as follows.

Assumption 4.1. (1) *The separated parts $f(x)$ and $g(y)$ are lower semicontinuous functions. $f(x)$, $g(y)$ and $\Psi(x,y)$ are all bounded below.*

(2) *$H(x,y)$ is a differentiable function with a Lipschitz continuous gradient. The Lipschitz constant of the gradient is l, i.e., $\|(\nabla_x H(x_1,y_1) - \nabla_x H(x_2,y_2), \nabla_y H(x_1,y_1) - \nabla_y H(x_2,y_2))\| \leqslant l\|(x_1 - x_2, y_1 - y_2)\|$.*

The BCD algorithm utilizes an alternating optimization method to update x and y, where each update of y integrates the latest value of x. In each iteration, the linear approximation of the smooth part $H(x,y)$ is used to approximate the local property of $H(x,y)$. Another penalty term with step size is added to balance the error of approximation. For the k-th iteration, we solve the following subproblems:

$$\begin{aligned} x^{k+1} &\in \arg\min_x \left\{ f(x) + \langle \nabla_x H(x^k, y^k), x - x^k \rangle + \frac{c_k}{2}\|x - x^k\|^2 \right\}, \\ y^{k+1} &\in \arg\min_y \left\{ g(y) + \langle \nabla_y H(x^{k+1}, y^k), y - y^k \rangle + \frac{d_k}{2}\|y - y^k\|^2 \right\}. \end{aligned} \tag{4.2}$$

The set of minimizers of the subproblem may not be a singleton. We select an arbitrary element to continue the iterations. The parameters c_k and d_k control the step sizes. Through the definition of the proximal operator (2.5), the subproblems can be rewritten as

$$x^{k+1} \in \mathrm{prox}_{c_k f}(x^k - c_k \nabla_x H(x^k, y^k)), \quad y^{k+1} \in \mathrm{prox}_{d_k g}(y^k - d_k \nabla_y H(x^{k+1}, y^k)). \tag{4.3}$$

4.1 Formalization of the update scheme

For any numerical algorithm, the formalization can typically be decomposed into two primary components: the problem data and the update mechanism. The problem data encapsulate well-founded assumptions about the problem, whereas the update mechanism encompasses the initial point, sequence of updates, and the specific rules applied at each iteration.

While the algorithm is described in natural language using coordinates in $\mathbb{R}^n$ and $\mathbb{R}^m$ following standard conventions in the literature, our formalization is conducted in general Hilbert spaces. This abstraction reflects that the specific dimensionality or coordinate representation is not essential to the algorithm. It also enables a cleaner and more reusable formal expression. Furthermore, the Hilbert space setting naturally supports extensions to matrix-valued problems, which are common in numerical optimization, thereby enhancing the practicality and generality of the formalization. The formalized problem data are presented below:

```
variable [NormedAddCommGroup E] [InnerProductSpace ℝ E] [CompleteSpace E]
variable [NormedAddCommGroup F] [InnerProductSpace ℝ F] [CompleteSpace F]
structure ProblemData (f : E → ℝ) (g : F → ℝ) (H : (WithLp 2 (E × F)) → ℝ) (l : NNReal) :
    Prop where
  lbdf : BddBelow (f '' univ)     lbdg : BddBelow (g '' univ)
  hf : LowerSemicontinuous f      hg : LowerSemicontinuous g
  conf : ContDiff ℝ 1 H           lpos : l > (0 : ℝ)
  lip : LipschitzWith l (gradient H)
```

All assumptions are bundled into the `ProblemData` structure. This design ensures that all fundamental assumptions are explicitly available at once, facilitating both the identification and consistent use of problem conditions throughout the formalization. The space $\mathbb{R}^n$ is replaced by a Hilbert space `E`. This not only makes the analysis clearer in formalization, but also generalizes the current results on $\mathbb{R}^n$ to a general Hilbert space. It is essential to note that the function H is defined on the domain `WithLp 2 (E × F)`. In MATHLIB, product spaces can be modeled in multiple ways. In particular, both `E × F` and `WithLp 2 (E × F)` represent elements as pairs (x, y), but their norms differ. For `(x,y)` of type `E × F`, the norm is defined as $\|(x, y)\| = \max\{\|x\|, \|y\|\}$, yielding a normed space without an inner product. In contrast, `WithLp 2 (E × F)` equips the product with the norm $\|(x, y)\| = \sqrt{\|x\|^2 + \|y\|^2}$, making it a Hilbert space. These norms are equivalent up to absolute constants. Since gradients are taken jointly with respect to (x, y) in $H(x, y)$, `WithLp 2 (E × F)` provides the appropriate Hilbert space structure, ensuring that the joint gradient is well-defined within an inner product space.

We give the formalized algorithm towards (4.3) using the `structure` type extending over `ProblemData`. The update scheme, predicated on the problem data, is structured as follows:

```
structure BCD (f : E → ℝ) (g : F → ℝ) (H : (WithLp 2 (E × F)) → ℝ) (l : NNReal)(x0 : E)
    (y0 : F) extends ProblemData f g H l where
  (x : ℕ → E)  (y : ℕ → F)
  (x0 : x 0 = x0) (y0 : y 0 = y0) (c d : ℕ → ℝ)
  s₁ : ∀ k, prox_prop (c k · f) (x k - c k · (grad_fst H (y k) (x k))) (x (k + 1))
  s₂ : ∀ k, prox_prop (d k · g) (y k - d k · (grad_snd H (x (k + 1)) (y k))) (y (k + 1))
```

The definitions for f, g and H are directly inherited from the problem data. The term `prox_prop f x y` means that $y \in \mathrm{prox}_f(x)$. We introduce additional notations using the same namespace BCD as follows:

```
def BCD.z {self : BCD f g H l x0 y0} : ℕ → WithLp 2 (E × F) :=
    fun n ↦ (WithLp.equiv 2 (E × F)).symm (self.x n, self.y n)

def BCD.ψ {_ : BCD f g H l x0 y0} :=
    fun z : WithLp 2 (E × F) ↦ (f z.1 + g z.2 + H z)
```

These definitions are given in the local namespace as beginning with "`BCD.`". Hence, if we use `alg : BCD f g H l x0 y0` to define an algorithm, `alg.z` automatically turns to this definition, which gives convenience for complicated calculation.

4.2 Convergence analysis

For the analysis of the fixed step size version of BCD, we mainly follow [7], and the detailed proof is omitted in this paper. We set the step sizes c_k and d_k consistently as $\frac{1}{\gamma l}$, where γ serves as a critical hyper-parameter. The parameter l represents the Lipschitz constant of ∇H. The conditions on the step sizes are given in `variable` as follows, and for simplicity, we omit to claim them for each theorem in this subsection.

```
variable (γ : ℝ) (hγ : γ > 1) (ck: ∀ k, alg.c k = 1 / (γ * l)) (dk: ∀ k, alg.d k = 1 / (γ *
    l))
```

To demonstrate the convergence of the algorithm, the analysis unfolds in several stages. Initially, we establish a sufficient descent condition showing that the function values decremented by the algorithm are monotonically decreasing.

Theorem 4.2. *Suppose that Assumption* 4.1 *holds. Under the condition that the iterative sequence* $\{z^k = (x^k, y^k)\}$ *generated by updates* (4.3) *is bounded and the step sizes satisfy* $c_k = d_k = \frac{1}{\gamma l}$ *for* $\gamma > 1$, *the sequence exhibits the following characteristics*:

(1) *At each iteration, the following decrement property holds*:

$$(\gamma - 1)l\|z^{k+1} - z^k\|^2 \leqslant H(z^k) - H(z^{k+1}), \quad \forall\, k \geqslant 0.$$

(2) *The sequence* $\{\|z^{k+1} - z^k\|\}_{k=1}^{\infty}$ *is square summable, i.e.,* $\sum_{k=1}^{\infty} \|z^{k+1} - z^k\|^2 < \infty$, *ensuring that* $\lim_{k\to\infty} \|z^{k+1} - z^k\| = 0$.

The following two statements formalize Theorem 4.2.

```
theorem Sufficient_Descent1 {alg : BCD f g H l x0 y0} :
    ∃ ρ₁ > 0, ρ₁ = (γ - 1) * l ∧ ∀ k, ρ₁ / 2 * ‖ alg.z (k+1) - alg.z k ‖ ^ 2 ≤ alg.ψ (alg.z
    k) - alg.ψ (alg.z (k + 1))

theorem Sufficient_Descent2 {alg : BCD f g H l x0 y0} (lbdψ : BddBelow (alg.ψ '' univ)):
    Tendsto (fun k ↦ ‖ alg.z (k + 1) - alg.z k ‖) atTop (nhds 0)
```

Theorem `Sufficient_Descent1` establishes the non-increasing nature of the objective function. Theorem `Sufficient_Descent2` guarantees that the distances between successive updates converge to zero, ensuring update stabilization.

To gain a deeper understanding of the analysis, it is crucial to examine the behavior of the subdifferential at the update points. We can establish that the subdifferential at each iteration is upper-bounded by the norm of the difference between two consecutive points, which implies that the norm of the subdifferential tends to zero as the iterations progress.

Lemma 4.3. *Suppose that Assumption* 4.1 *holds. Suppose that* $\{z^k\}$ *generated by the scheme* (4.3) *is bounded and the step sizes satisfy* $c_k = d_k = \frac{1}{\gamma l}$ *for* $\gamma > 1$. *For each positive integer* k, *define*

$$A_x^k := c_{k-1}(x^{k-1} - x^k) + \nabla_x H(x^k, y^k) - \nabla_x H(x^{k-1}, y^k),$$
$$A_y^k := d_{k-1}(y^{k-1} - y^k) + \nabla_y H(x^k, y^k) - \nabla_y H(x^k, y^{k-1}).$$

Then $(A_x^k, A_y^k) \in \hat{\partial}\Psi(x^k, y^k)$ *and there exists* $M = (2\gamma + 2)l > 0$ *such that*

$$\|(A_x^k, A_y^k)\| \leqslant \|A_x^k\| + \|A_y^k\| \leqslant M\|z^k - z^{k-1}\|, \quad \forall\, k \geqslant 1.$$

The specific values of A_x^k and A_y^k for the Fréchet subdifferentials are not essential to the continuation of the proof. What matters is their existence and norm bound. Therefore, for simplicity, our formalized version of Lemma 4.3 omits the explicit values of the subdifferentials and focuses solely on their existence and bounds. In our proof, the values of A_x^k and A_y^k are used, and the membership relation is established by the calculation rule in Theorem 2.8. One of the most difficult problems in formalization is to deal with the subdifferential in the product space. We establish calculation rules for block-wise subdifferential. The formalized version is provided below.

```
theorem Ψ_subdiff_bound {alg : BCD f g H l x0 y0} :
    ∃ ρ > 0, ∀ k, ∃ dΨ ∈ f_subdifferential alg.ψ (alg.z (k + 1)), ‖dΨ‖ ≤ ρ * ‖ alg.z (k + 1)
    - alg.z k ‖
```

From this boundedness of the subdifferential, we obtain that the limit points of the update sequence $\{z^k\}$ constitute critical points of the objective function.

Lemma 4.4. *Suppose that Assumption* 4.1 *holds. Let* $\{z^k\}$ *generated by the algorithm* (4.3) *be bounded and the step sizes satisfy* $c_k = d_k = \frac{1}{\gamma l}$ *for* $\gamma > 1$. *Denote the limit set of* $\{z^k\}$ *by* $\omega(z^0)$. *It holds that* $\emptyset \neq \omega(z^0) \subset \operatorname{crit}\Psi(z)$ *and the objective function* $\Psi(z)$ *is finite and constant on* $\omega(z^0)$.

We add a definition of limit set in Lean using the function `MapClusterPt` in MATHLIB4, which denotes the cluster point of a mapping. The formalized theorem is given as follows.

```
def limit_set (z : ℕ → E) := {x | MapClusterPt x atTop z}
theorem limitset_property {alg : BCD f g H l x0 y0} (bd : Bornology.IsBounded (alg.z '' univ))
    (lbdψ : BddBelow (alg.ψ '' univ)):
    (limit_set alg.z).Nonempty ∧ (limit_set alg.z) ⊆ critical_point alg.ψ ∧ ∃ (c : ℝ) , ∀ x ∈
     (limit_set alg.z) , alg.ψ x = c
```

Theorem 4.2 does not address the properties of the limit point of z^k. To ensure convergence to a critical point of the objective function $\Psi(x, y)$, it is assumed that the objective function $\Psi(x, y)$ exhibits the KL property near these critical points, strengthening conditions around the critical points of $H(x, y)$. We can obtain a stronger version of the convergence theorem.

Theorem 4.5. *Suppose that Assumption* 4.1 *holds. If* $\Psi(x,y)$ *is a KL function and the iterative sequence* $\{z^k\}$ *adheres to the updates* (4.3)*, the following theorems hold*:

(1) *The sequence* $\{z^k\}$ *possesses a finite total variation*: $\sum_{k=1}^{\infty}\|z^{k+1}-z^k\|<+\infty$,

(2) $\{z^k\}$ *converges to a critical point* $z^*=(x^*,y^*)$ *of* Ψ.

We can prove the formalized version of Theorem 4.5.

```
theorem Limited_length {alg : BCD f g H l x0 y0} (bd : Bornology.IsBounded (alg.z '' univ)) (hψ
    : KL_function alg.ψ) (lbdψ : BddBelow (alg.ψ '' univ)): ∃ M : ℝ, ∀ n,
    Σ k in Finset.range n, ‖ alg.z (k + 1) - alg.z k ‖ ⩽ M

theorem Convergence_to_critpt {alg : BCD f g H l x0 y0} (bd : Bornology.IsBounded (alg.z ''
    univ)) (hψ : KL_function alg.ψ) (lbdψ : BddBelow (alg.ψ '' univ)) :
    ∃ z_ : (WithLp 2 (E × F)), z_ ∈ (critical_point alg.ψ) ∧ Tendsto alg.z atTop (nhds z_)
```

The proofs of the aforementioned theorems rely on algebraic manipulations and basic analytical tools, including inequalities and continuity arguments, to establish the sufficiency and boundedness conditions. This rigorous framework supports a clear iterative convergence analysis and enables precise customization of the BCD algorithm for specific problem settings. Furthermore, the foundational structure naturally extends to a range of optimization challenges.

5 The alternating direction method of multipliers

The ADMM method is frequently used to solve optimization problems comprising multiple components coupled by a linear constraint. In this paper, we focus on a certain kind of the ADMM method:

$$\begin{aligned}\min_{x_1,x_2}\quad & f_1(x_1)+f_2(x_2),\\ \text{s.t.}\quad & A_1x_1+A_2x_2=b,\end{aligned}\tag{5.1}$$

where $f_1(x_1):\mathbb{R}^n\to\mathbb{R}$ and $f_2(x_2):\mathbb{R}^m\to\mathbb{R}$ are two separated parts of the objective function. The matrices $A_1\in\mathbb{R}^{p\times n}$ and $A_2\in\mathbb{R}^{p\times m}$ link the variables via linear constraints, with $b\in\mathbb{R}^p$ representing a given vector. Our analysis of this algorithm mainly follows [9, 14]. The assumptions of this problem are given below.

Assumption 5.1. (1) *The separated parts* $f_1(x_1)$ *and* $f_2(x_2)$ *are lower semicontinuous convex functions.*

(2) *The solution set of the problem* (5.1) *is nonempty and satisfies Slater's condition, i.e., there exists a feasible solution in the interior of the domain*:

$$\exists\,(\tilde{x}_1,\tilde{x}_2)\in\mathrm{int}(\mathrm{dom}f_1\times\mathrm{dom}f_2)\quad s.t.\ A_1\tilde{x}_1+A_2\tilde{x}_2=b.$$

The augmented Lagrangian function of the problem (5.1) is given as

$$L_\rho(x_1,x_2,y)=f_1(x_1)+f_2(x_2)+\langle y,A_1x_1+A_2x_2-b\rangle+\frac{\rho}{2}\|A_1x_1+A_2x_2-b\|^2,\tag{5.2}$$

where ρ is the penalty parameter. To solve the constraint optimization problem (5.1), the ADMM method sequentially minimizes the augmented Lagrangian function with respect to each variable, then updates the Lagrange multipliers. Based on (5.2), the iterative scheme of the ADMM is as follows:

$$x_1^{k+1}=\arg\min_{x_1}L_\rho(x_1,x_2^k,y^k),\tag{5.3}$$

$$x_2^{k+1}=\arg\min_{x_2}L_\rho(x_1^{k+1},x_2,y^k),\tag{5.4}$$

$$y^{k+1}=y^k+\tau\rho(A_1x_1^{k+1}+A_2x_2^{k+1}-b),\tag{5.5}$$

where τ is a step size. The alternating minimization steps for x_1 and x_2 are followed by the iteration for the dual variable y. We have an additional assumption on the schemes (5.3)–(5.5) as below.

Assumption 5.2. *The subproblems* (5.3) *and* (5.4) *have a unique solution, respectively.*

The first item in Assumption 5.1 guarantees the convexity of the subproblems. Assumption 5.2 ensures the uniqueness of the solutions to the subproblems, which is crucial for well-defined iterations. This assumption, though seemingly strong, is necessary to ensure the convergence of iteration points to a KKT point of the problem (5.1). Under Slater's condition, the solution to the problem (5.1) can be expressed as a KKT pair, enabling us to replace the solution with its corresponding KKT pair and simplify the analysis.

5.1 Formalization of the update scheme

This subsection provides a detailed formalization of the ADMM method. First, we define the fundamental structure of the optimization problem, given by (5.1). While the ADMM algorithm is typically presented in finite-dimensional Euclidean spaces using matrix representations, our formalization adopts a more abstract framework based on Hilbert spaces. This design follows the same principle as in the BCD formalization, emphasizing that the specific choice of coordinates or dimensions is not intrinsic to the logic of the algorithm. By lifting the formulation to general Hilbert spaces, we achieve a cleaner and more modular formalization that can naturally accommodate extensions to infinite-dimensional or function spaces. In particular, the linear operators A_1 and A_2, which are commonly given as matrices in the Euclidean setting, are represented as bounded linear maps between finite-dimensional Hilbert spaces. This abstraction not only weakens the reliance on coordinate representations but also provides a more general and reusable formal model. This structure is formally represented using a `class` type in Lean as follows.

```
variable (E₁ E₂ F : Type*)
variable [NormedAddCommGroup E₁] [InnerProductSpace ℝ E₁] [FiniteDimensional ℝ E₁]
variable [NormedAddCommGroup E₂] [InnerProductSpace ℝ E₂] [FiniteDimensional ℝ E₂]
variable [NormedAddCommGroup F ] [InnerProductSpace ℝ F ] [FiniteDimensional ℝ F]
class Opt_problem where
  f₁ : E₁ → ℝ            f₂ : E₂ → ℝ
  A₁ : E₁ → L[ℝ] F       A₂ : E₂ → L[ℝ] F
  b  : F
  lscf₁ : LowerSemicontinuous f₁        lscf₂ : LowerSemicontinuous f₂
  cf₁ : ConvexOn ℝ univ f₁              cf₂ : ConvexOn ℝ univ f₂
  nonempty : ∃ x₁ x₂, (A₁ x₁) + (A₂ x₂) - b = 0 ∧ IsMinOn (fun (x₁, x₂) ↦ f₁ x₁ + f₂ x₂) univ
    (x₁, x₂)
```

The functions f_1 and f_2 are real-valued, lower semicontinuous (`lscf`$_1$, `lscf`$_2$), and convex (`cf`$_1$, `cf`$_2$). The domains of the functions are taken as the universe for simplicity. The property `nonempty` guarantees the second item in Assumption 5.1. The continuous linear maps A_1 and A_2, corresponding to matrices in the original formulation, are implemented in this form to be consistent with the abstract Hilbert space framework. The following definition captures the augmented Lagrangian function (5.2).

```
def Augmented_Lagrangian_Function (opt : Opt_problem E₁ E₂ F) (ρ : ℝ) : E₁ × E₂ × F → ℝ :=
    fun (x₁, x₂, y) => (opt.f₁ x₁) + (opt.f₂ x₂) + inner y ((opt.A₁ x₁) + (opt.A₂ x₂) - opt.b) +
     ρ / 2 * ‖(opt.A₁ x₁) + (opt.A₂ x₂) - opt.b ‖ ^ 2
```

ADMM alternately updates x_1 and x_2, minimizing one variable at a time while keeping the others fixed. To support this iterative process and prove convergence, we define the iteration structure and introduce auxiliary variables as follows.

```
class ADMM extends (Opt_problem E₁ E₂ F) where
  x₁ : ℕ → E₁   x₂ : ℕ → E₂  y  : ℕ → F
  ρ  : ℝ        τ  : ℝ       hrho : ρ > 0
  htau : 0 < τ ∧ τ < (1 + sqrt 5) / 2
  iterx₁  : ∀ k, IsMinOn (fun x₁ ↦ (Augmented_Lagrangian_Function E₁ E₂ F toOpt_problem ρ)
    (x₁, x₂ k, y k)) univ (x₁ (k + 1))
  uiterx₁ : ∀ k, Admm_sub_Isunique (fun x₁ ↦ (Augmented_Lagrangian_Function E₁ E₂ F
    toOpt_problem ρ) (x₁, x₂ k, y k)) (x₁ (k + 1)) (iterx₁ k)
```

```
iterx₂  : ∀ k, IsMinOn (fun x₂ ↦ (Augmented_Lagrangian_Function E₁ E₂ F toOpt_problem ρ) (x₁
  (k + 1), x₂, y k)) univ (x₂ (k + 1))
uiterx₂ : ∀ k, Admm_sub_Isunique (fun x₂ ↦ (Augmented_Lagrangian_Function E₁ E₂ F
  toOpt_problem ρ) (x₁ (k + 1), x₂, y k)) (x₂ (k + 1)) (iterx₂ k)
itery   : ∀ k, y (k + 1) = y k + (τ * ρ) • ((A₁ $ x₁ (k + 1)) + (A₂ $ x₂ (k + 1)) - b)
```

The parameters ρ and τ range within the intervals $(0,\infty)$ and $(0,\frac{1+\sqrt{5}}{2})$, respectively. The property `iterx`$_1$ ensures that for any iteration index k, the point $x_1^{(k+1)}$ is a global minimizer of the function $L(x_1,x_2^{(k)},y^{(k)})$. The uniqueness of this minimizer is established by `uiterx`$_1$. Similarly, `iterx`$_2$ ensures that $x_2^{(k+1)}$ achieves global minimization for the function $L(x_1^{(k+1)},x_2,y^{(k)})$, with its uniqueness established by `uiterx`$_2$. The iteration of the dual variable y is defined by `itery`, which specifies the update formula for $y^{(k)}$ at each iteration.

The KKT conditions are necessary for optimality, given certain regularity assumptions. They are essential for determining the termination of the iterative process in the ADMM algorithm. For the problem (5.1), these conditions have been adapted to align with the algorithm's framework and are expressed as follows:

$$\begin{cases} -A_1^*y \in \partial f_1(x_1), \\ -A_2^*y \in \partial f_2(x_2), \\ A_1x_1+A_2x_2=b. \end{cases} \tag{5.6}$$

The formalization of the KKT conditions adheres to the structure of natural language and is presented below using the type `class`.

```
class Convex_KKT (x₁ : E₁) (x₂ : E₂) (y : F) (opt : Opt_problem E₁ E₂ F) : Prop where
  subgrad₁ : -(ContinuousLinearMap.adjoint opt.A₁) y ∈ SubderivAt opt.f₁ x₁
  subgrad₂ : -(ContinuousLinearMap.adjoint opt.A₂) y ∈ SubderivAt opt.f₂ x₂
  eq : (opt.A₁ x₁) + (opt.A₂ x₂) = opt.b
```

5.2 Convergence analysis

This subsection describes the convergence of the ADMM, with the convergence proof primarily based on [14]. The proof is divided into two components: a lemma that verifies the descending property of an auxiliary sequence, and a theorem that establishes the convergence of the ADMM iterative sequence. We denote by (x_1^*,x_2^*,y^*) the KKT pair for the original problem and define the error vectors and auxiliary functions as follows:

$$\begin{aligned} &(e_1^k,e_2^k,e_y^k)=(x_1^k,x_2^k,y^k)-(x_1^*,x_2^*,y^*),\\ &u^k=-A_1^*[y^k+(1-\tau)\rho(A_1e_1^k+A_2e_2^k)+\rho A_2(x_2^{k-1}-x_2^k)],\\ &v^k=-A_2^*[y^k+(1-\tau)\rho(A_1e_1^k+A_2e_2^k)],\\ &\Psi_k=\frac{1}{\tau\rho}\|e_y^k\|^2+\rho\|A_2e_2^k\|^2,\\ &\Phi_k=\Psi_k+\max(1-\tau,1-\tau^{-1})\rho\|A_1e_1^k+A_2e_2^k\|^2. \end{aligned}$$

The functions u^k and v^k are related to the optimality conditions of each subproblem; Ψ_k and Φ_k measure the errors (e_1^k,e_2^k,y^k). All of these auxiliary variables are formalized using the `ADMM` namespace.

Our first step is to prove the descent of Φ_k, implying that the limits of the errors (e_1^k,e_2^k,y^k) are zero.

Lemma 5.3. *Suppose that Assumptions* 5.1 *and* 5.2 *hold. Let* $\{(x_1^k,x_2^k,y^k)\}$ *be a sequence generated by the ADMM iterations* (5.3)–(5.5). *For each* $k\geqslant 1$, *it holds that* $u^k\in\partial f_1(x_1^k)$ *and* $v^k\in\partial f_2(x_2^k)$. *The error obtains a decrease as*

$$\begin{aligned} \Phi_k-\Phi_{k+1}\geqslant{}&\min(\tau,1+\tau-\tau^2)\rho\|A_2(x_2^k-x_2^{k+1})\|^2\\ &+\min(1,1+\tau^{-1}-\tau)\rho\|A_1e_1^{k+1}+A_2e_2^{k+1}\|^2. \end{aligned} \tag{5.7}$$

This lemma is formalized in Lean as follows.

```
lemma Φ_isdescending {E₁ : Type*} {E₂ : Type*} {F : Type*}
  [NormedAddCommGroup E₁][InnerProductSpace ℝ E₁][FiniteDimensional ℝ E₁]
  [NormedAddCommGroup E₂][InnerProductSpace ℝ E₂][FiniteDimensional ℝ E₂]
  [NormedAddCommGroup F][InnerProductSpace ℝ F][FiniteDimensional ℝ F]
  {admm : ADMM E₁ E₂ F} {admm_kkt : Existence_of_kkt admm} [Setting E₁ E₂ F admm admm_kkt] : ∀
    n : ℕ+, (Φ n) - (Φ (n + 1)) ≥ (min τ (1 + τ - τ ^ 2)) * ρ * ‖A₂ (x₂ n - x₂ (n + 1))‖ ^ 2 +
     (min 1 (1 + 1 / τ - τ)) * ρ * ‖A₁ (e₁ (n + 1)) + A₂ (e₂ (n + 1))‖ ^ 2
```

Role of the `Setting` instance. The entire analysis proceeds under a set of assumptions, mentioned only once in the natural language. However, Lean requires every lemma to list its assumptions explicitly. We therefore encapsulate *all* global hypotheses—space structure, Slater condition, full-rank constraints, uniqueness of sub-problem solutions, and bounds on ρ and τ—in the following formalized class.

```
class Setting (E₁ E₂ F : outParam Type*)
      [NormedAddCommGroup E₁] [InnerProductSpace ℝ E₁] [FiniteDimensional ℝ E₁]
      [NormedAddCommGroup E₂] [InnerProductSpace ℝ E₂] [FiniteDimensional ℝ E₂]
      [NormedAddCommGroup F ] [InnerProductSpace ℝ F ] [FiniteDimensional ℝ F ]
      (admm : outParam (ADMM E₁ E₂ F))
      (admm_kkt : outParam (Existence_of_kkt admm)) where
```

Including `[Setting ⋯]` in a theorem enables Lean's type-class resolver to automatically supply the entire assumption bundle. This keeps theorem statements concise and enhances portability to new spaces or ADMM variants by instantiating a different `Setting`. Declaring the algorithm object `admm` and its KKT certificate `admm_kkt` as `outParam`s turns the class `Setting` into an auto-completion switch. Once an instance of `[Setting E₁ E₂ F ⋯]` is available in the local environment, Lean can apply any theorem whose context contains `[Setting E₁ E₂ F ⋯]`, while silently inferring all remaining arguments. As a result, the user needs only to provide the minimal visible data, and the system fills in `admm`, `admm_kkt` and every other global assumption automatically. This approach follows the principle: "*write less, apply more*", without compromising formal rigor.

We then outline the proof of this lemma, structured as follows:

(1) We prove that u^k and v^k belong to the subdifferentials of f_1 and f_2, respectively: $u^{k+1} \in \partial f_1(x_1^{k+1})$ and $v^{k+1} \in \partial f_2(x_2^{k+1})$. This establishes that u^k and v^k approximate the KKT pair and converge to it, due to the closed property of subgradients. The relevant proofs are provided in `uvBelongsToSubgradient`, `u_inthesubgradient`, and `v_inthesubgradient`.

(2) We establish an inequality that describes the lower bound of the difference in Φ_k:

$$\begin{aligned}&\frac{1}{\tau\rho}(\|e_y^k\|^2 - \|e_y^{k+1}\|^2) - (2-\tau)\rho\|A_1x_1^{k+1} + A_2x_2^{k+1} - b\|^2\\ &\quad + 2M_{k+1} - \rho\|A_2(x_2^{k+1} - x_2^k)\|^2 - \rho\|A_2e_2^{k+1}\|^2 + \rho\|A_2e_2^k\|^2 \geqslant 0.\end{aligned}$$

(3) For the cases $0 < \tau < 1$ and $\tau > 1$, we categorize them and analyze their respective properties. Finally, we can get the desired inequality (5.7).

This lemma provides the foundation for proving convergence. It establishes that the iterations generated by the ADMM remain bounded throughout the process.

With the lemmas established above, we need only demonstrate the convergence of the series $\{(x_1^k, x_2^k, y^k)\}$. This can be achieved by proving the convergence of subsequences and using the monotonicity of the sequence.

Theorem 5.4. *Suppose that Assumptions* 5.1 *and* 5.2 *hold. Assuming that A_1 and A_2 have full rank and $\tau \in (0, \frac{1+\sqrt{5}}{2})$, the series $\{(x_1^k, x_2^k, y^k)\}$ generated by the ADMM iterations* (5.3)–(5.5) *converge to a KKT pair of* (5.1).

This theorem is defined in Lean as follows.

```
theorem ADMM_convergence {E₁ : Type*} {E₂ : Type*} {F : Type*}
  [NormedAddCommGroup E₁][InnerProductSpace ℝ E₁][FiniteDimensional ℝ E₁]
  [NormedAddCommGroup E₂][InnerProductSpace ℝ E₂][FiniteDimensional ℝ E₂]
```

```
[NormedAddCommGroup F][InnerProductSpace ℝ F][FiniteDimensional ℝ F]
{admm : ADMM E₁ E₂ F} {admm_kkt : Existence_of_kkt admm}(fullrank₁ : Function.Injective
  (OptProblem.A₁ E₂)) (fullrank₂ : Function.Injective (OptProblem.A₂ E₁))
[Setting E₁ E₂ F admm (admm_kkt₁ fullrank₁ fullrank₂)] :
∃ _x₁ _x₂ _y, Convex_KKT _x₁ _x₂ _y ADMM.toOptProblem ∧
    Tendsto (ADMM.x₁ E₂ F) atTop (nhds _x₁) ∧ Tendsto (ADMM.x₂ E₁ F) atTop (nhds _x₂) ∧
  Tendsto (ADMM.y E₁ E₂) atTop (nhds _y)
```

We hereby outline the key aspects of the proof, structured as follows:

(1) By Lemma 5.3, the sequence $\{(x_1^k, x_2^k, y^k)\}$ is bounded. We can demonstrate that each of the sequences x_1^k, x_2^k, y^k, u^k and v^k admits a convergent subsequence satisfying $(x_1^{\phi(k)}, x_2^{\phi(k)}, y^{\phi(k)}, u^{\phi(k)}, v^{\phi(k)}) \to (x_1'', x_2'', y'', -A_1^* y'', -A_2^* y'')$, as $k \to \infty$. Using monotonicity, we then prove the convergence of the entire sequence.

(2) The KKT conditions for the problem (5.1) are given by (5.6). Based on the boundedness of the iterates and the diminishing residuals, we can conclude that the sequence $\{x_1^k, x_2^k, y^k\}$ converges to a point satisfying the KKT conditions

$$\begin{cases} -A_1^* y'' \in \partial f_1(x_1''), \\ -A_2^* y'' \in \partial f_2(x_2''), \\ A_1(x_1'') + A_2(x_2'') = b. \end{cases}$$

(3) We derive the convergence of the entire sequences x_1^k, x_2^k, y^k, u^k and v^k from the descending property of Φ^k.

Thus, under specified assumptions, the ADMM algorithm is guaranteed to converge to an optimal solution.

6 Insights from formalization and design principles

The transition from informal mathematics to a formal development is rarely a one-to-one translation. The formalization process itself tends to expose hidden assumptions, demand conceptually sharper definitions, and reveal underlying structural regularities that are easily overlooked in traditional exposition. The Lean 4 mechanization accompanying this work led us to adopt the following principles, which in turn shape the ambient setting, the precise form of our definitions, and the fundamental architecture of our proofs.

• **Generality and interface alignment.** We state our results at the abstract level of real Hilbert spaces, a methodological choice that rigorously distinguishes formal development from traditional, concrete reasoning. While conventional exposition often relies on the intuition of Euclidean space, formalization necessitates precisely using only the required abstract properties. In `MATHLIB`, this environment is captured by `[NormedAddCommGroup E]` + `[InnerProductSpace ℝ E]` + `[CompleteSpace E]`. We strictly avoid coordinate models to achieve interface alignment. This strategy yields significant methodological benefits: it ensures that all foundational theorems proven at the Hilbert space level (e.g., in linear algebra or topology) apply definitionally to our objects, maximizing reusability, eliminating type mismatch issues, and guaranteeing that core properties are directly inherited from existing library structures, thereby avoiding the locality of a coordinate-based proof.

• **Definitions at the right level.** The formalization process compels us to distill the essence of definitions and to prioritize simple and serviceable formulations. This methodological principle encourages replacing "messy" limit schemata with clean geometric or topological characterizations, clarifying the notion itself while resulting in a markedly cleaner and more reusable proof architecture. For example, in defining the subdifferential, we move beyond traditional, sequence-heavy limit schemata to adopt an index-free, geometric formulation uniformly applicable in real Hilbert spaces. We use the closure of the graph $G(\partial f) := \{(x, u) : u \in \partial f(x)\}$ to define the subdifferential: $\partial f(x) = \{u \in E : (x, u) \in \overline{G(\partial f)}\}$. This closure-based representation offers significant mathematical advantages: it constitutes a more fundamental and topologically superior description than sequential limits, and the value-convergence

hypothesis $f(x_k) \to f(x)$ is naturally subsumed by the resulting closed-graph implication. Furthermore, it removes the burden of index bookkeeping, interfaces cleanly with quantifiable expressions such as $\mathrm{dist}(0, \partial f(x))$ used in KL theory, and aligns perfectly with the topological and variational APIs in `MATHLIB`. Ultimately, this choice streamlines complex limit-passing steps, yielding a formulation that is both conceptually transparent and automation-friendly.

• **Packaging human-default hypotheses as instances.** Formalization compels us to restructure traditional mathematical exposition in a more insightful and generative way. While natural-language proofs often elide ambient assumptions, such as a space's completeness or inner-product structure, the formal process demands that these "human-default" hypotheses be externalized and packaged into the typeclass layer for implicit delivery. Specifically, the normed-additive structure, inner-product geometry, and completeness live as

$$\texttt{[NormedAddCommGroup E]} \ + \ \texttt{[InnerProductSpace } \mathbb{R}\, E\texttt{]} \ + \ \texttt{[CompleteSpace } E\texttt{]}.$$

This practice offers a crucial thinking paradigm for traditional mathematics: separation of concerns. Core statements focus purely on mathematical content, while routine ambient obligations are delegated to the inference engine. This systematic packaging ensures that the required structure automatically propagates through canonical constructions (products, closed subspaces) without manual bookkeeping. This methodological refinement not only guarantees locality of assumptions and robustness under refactoring but, more importantly, provides an extensible, encoding-agnostic interface that significantly promotes the discovery and integration of new results.

• **Decomposition as a first-class proof principle.** Formal proof design transcends mere mimicry of long book-length derivations; it elevates decomposition to a first-class principle. It advocates structuring proofs as compositions of small, conceptually orthogonal interfaces with stable contracts. This decompositional mindset has profound implications for natural-language mathematics: it encourages the construction of a logical calculus rather than a linear narrative. By precisely controlling the local hypotheses and scope of each lemma, the formal development clearly reveals which assumptions are genuinely active at each step of the reasoning. Beyond strengthening automation and predictability, this structure demonstrates its value as a mathematical discovery tool by maximizing cross-context reusability. In effect, this methodology replaces long bespoke scripts with a compact, modular calculus of lemmas that transparently captures the logical flow, offering a highly modular and verifiable template whose utility extends far beyond mere validation.

7 Conclusion

In this paper, the formalization of the two first-order optimization algorithms, BCD and ADMM, specifically targeting block-structured optimization problems, is provided in Lean. To address nonsmooth and nonconvex objectives, the subdifferential and the KL property are formalized, together with related theorems crucial for convergence analysis. We give the formal update scheme of the numerical algorithms using the structure type. Building on these concepts, we prove that the BCD algorithm converges to the critical points, and the ADMM algorithm converges to the KKT points of the problem.

Our formalization framework can be naturally extended to derive further convergence results under alternative assumptions on the problem. Furthermore, numerous variations of the BCD and ADMM documented in the literature can be formalized using our established methodology. For example, various variants of the BCD algorithm, distinguished by slight modifications in their update schemes, can be systematically analyzed within this framework. Similarly, multiple versions of the ADMM algorithm are designed to accommodate specific problem structures. The formalization of these algorithms enables a more comprehensive study of the convergence properties of the ADMM method.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2024YFA1012903) and National Natural Science Foundation of China (Grant Nos. 12331010 and 12288101).

References

1 Appel A, Kellison A. VCFloat2: Floating-point error analysis in Coq. In: Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs. New York: ACM, 2024, 14–29
2 Attouch H, Bolte J, Redont P, et al. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. Math Oper Res, 2010, 35: 438–457
3 Attouch H, Bolte J, Svaiter B F. Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. Math Program, 2013, 137: 91–129
4 Bento G C, Mordukhovich B S, Mota T S, et al. Convergence of descent methods under Kurdyka–Łojasiewicz properties. arXiv:2407.00812, 2024
5 Boldo S, Lelay C, Melquiond G. Coquelicot: A user-friendly library of real analysis for Coq. Math Comput Sci, 2015, 9: 41–62
6 Boldo S, Melquiond G. Flocq: A unified library for proving floating-point algorithms in Coq. In: Proceedings of the 20th IEEE Symposium on Computer Arithmetic. Los Alamitos: IEEE, 2011, 243–252
7 Bolte J, Sabach S, Teboulle M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math Program, 2014, 146: 459–494
8 Boyd S. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn, 2011, 3: 1–122
9 Chen L, Sun D, Toh K C. A note on the convergence of ADMM for linearly constrained convex optimization problems. Comput Optim Appl, 2017, 66: 327–343
10 de Dinechin F, Lauter C, Melquiond G. Certifying the floating-point implementation of an elementary function using Gappa. IEEE Trans Comput, 2011, 60: 242–253
11 de Moura L, Kong S, Avigad J, et al. The Lean theorem prover (system description). In: Automated Deduction. Lecture Notes in Computer Science, vol. 9195. New York: Springer, 2015, 378–388
12 Dupuis F, Lewis R Y, Macbeth H. Formalized functional analysis with semilinear maps. In: Proceedings of the 13th International Conference on Interactive Theorem Proving. Leibniz International Proceedings in Informatics (LIPIcs), vol. 237. Wadem: Schloss Dagstuhl, 2022, 10:1–10:19
13 Faissole F. Formally-verified round-off error analysis of Runge-Kutta methods. J Automat Reason, 2024, 68: 1
14 Fazel M, Pong T K, Sun D, et al. Hankel matrix rank minimization with applications to system identification and realization. SIAM J Matrix Anal Appl, 2013, 34: 946–977
15 Fortin M, Glowinski R. Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems. Studies in Mathematics and Its Applications, vol. 15. Amsterdam: Elsevier, 2000
16 Gabay D, Mercier B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Comput Math Appl, 1976, 2: 17–40
17 Glowinski R. Numerical Methods for Nonlinear Variational Problems. New York: Springer, 2013
18 Glowinski R. On alternating direction methods of multipliers: A historical perspective. In: Modeling, Simulation and Optimization for Science and Technology. Dordrecht: Springer, 2014, 59–82
19 Glowinski R, Marroco A. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite d'une classe de problemes de Dirichlet non lineaires. Revue française d'automatique, informatique, recherche operationnelle. Anal Numér, 1975, 9: 41–76
20 Huet G, Kahn G, Paulin-Mohring C. The Coq proof assistant: A tutorial. Rapport Technique, https://inria.hal.science/inria-00069967, 1997
21 Immler F. Verified reachability analysis of continuous systems. In: Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 9035. New York: Springer, 2015, 37–51
22 Immler F. A verified ODE solver and the Lorenz attractor. J Automat Reason, 2018, 61: 73–111
23 Immler F, Hölzl J. Numerical analysis of ordinary differential equations in Isabelle/HOL. In: Interactive Theorem Proving. Lecture Notes in Computer Science, vol. 7406. New York: Springer, 2012, 377–392
24 Immler F, Hölzl J. Ordinary differential equations. Archive of Formal Proofs, https://isa-afp.org/entries/Ordinary_Differential_Equations.html, 2012
25 Kellison A E, Appel A W. Verified numerical methods for ordinary differential equations. In: Proceedings of the 15th International Workshop on Numerical Software Verification. Lecture Notes in Computer Science, vol. 13466. Berlin: Springer-Verlag, 2022, 147–163
26 Kellison A E, Hsu J. Numerical Fuzz: A type system for rounding error analysis. In: Proceedings of the ACM on Programming Languages, vol. 8. New York: ACM, 2024, 1954–1978
27 Li C, Wang Z, He W, et al. Formalization of convergence rates of four first-order algorithms for convex optimization. J Automat Reason, 2025, 69: 28
28 Łojasiewicz S. Une propriete topologique des sous-ensembles analytiques reels. In: Les Équations aux Dérivées Partielles, vol. 117. Palaiseau: École Polytechnique, 1963, 87–89
29 Melquiond G. CoqInterval: A Coq library for proving bounds on real-valued expressions. https://coqinterval.

gitlabpages.inria.fr/
30 Melquiond G. Plotting in a formally verified way. arXiv:2108.03974, 2021
31 Nipkow T, Paulson L C, Wenzel M. A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science. Cham: Springer, 2002
32 Ortega J M, Rheinboldt W C. Iterative Solution of Nonlinear Equations in Several Variables. Philadelphia: SIAM, 1970
33 Polyak B T. Gradient methods for the minimisation of functionals. USSR Comput Math Math Phys, 1963, 3: 864–878
34 Powell M J D. On search directions for minimization algorithms. Math Program, 1973, 4: 193–201
35 Tekriwal M, Appel A W, Kellison A E, et al. Verified correctness, accuracy, and convergence of a stationary iterative linear solver: Jacobi method. In: Intelligent Computer Mathematics. New York: Springer, 2023, 206–221
36 Tekriwal M, Miller J, Jeannin J B. Formalization of asymptotic convergence for stationary iterative methods. In: NASA Formal Methods. New York: Springer, 2024, 37–56
37 The MATHLIB Community. The Lean mathematical library. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. New York: ACM, 2020, 367–381
38 Tseng P. Convergence of a block coordinate descent method for nondifferentiable minimization. J Optim Theory Appl, 2001, 109: 475–494
39 van Doorn F, Macbeth H. Integrals within integrals: A formalization of the Gagliardo-Nirenberg-Sobolev inequality. In: Proceedings of 15th International Conference on Interactive Theorem Proving. Leibniz International Proceedings in Informatics (LIPIcs), vol. 309. Wadern: Schloss Dagstuhl, 2024, 37:1–37:18
40 Ying K, Degenne R. A formalization of Doob's martingale convergence theorems in mathlib. In: Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. New York: ACM, 2023, 334–347