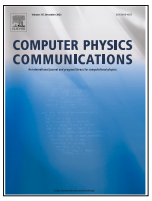
Computer Physics Communications 329 (2026) 110368



Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc



Computer Programs in Physics



Update of PHYSBO: Improving usability and portability of Bayesian optimization for physics and materials research

Yuichi Motoyama [a,*], Kazuyoshi Yoshimi [a,*], Tatsumi Aoyama [a], Kei Terayama [b], Koji Tsuda [c,d], Ryo Tamura [c,d,*]

[a] Institute for Solid State Physics, The University of Tokyo, Chiba, 277-8581, Japan
[b] Graduate School of Medical Life Science, Yokohama City University, Yokohama, 230-0045, Japan
[c] Graduate School of Frontier Sciences, The University of Tokyo, Chiba, 277-8561, Japan
[d] Center for Basic Research on Materials, National Institute for Materials Science, Ibaraki, 305-0044, Japan

ARTICLE INFO

The review of this paper was arranged by Prof. Martin Weigel

Keywords:
Bayesian optimization
Materials science
Physics simulations
Multi-objective optimization
Continuous optimization
Research software

ABSTRACT

Bayesian optimization (BO) is widely used to accelerate physics and materials research, where objective function evaluations are computationally or experimentally expensive. While many BO frameworks focus on algorithmic efficiency, practical usability and portability are equally critical for sustained use in real research environments. PHYSBO is a Bayesian optimization library designed to address these needs by enabling optimization over user-defined candidate pools and by supporting domain-specific problem settings.

This paper presents the major updates introduced in PHYSBO versions 2 and 3, with a focus on improvements in usability, portability, and practical deployment rather than on new optimization algorithms. In PHYSBO version 2, the software license was changed from GPL to MPL to improve compatibility with a wider range of research and software ecosystems. Building on this revision, PHYSBO version 3 introduces a set of implementation-oriented updates aimed at improving usability and portability, without modifying the core optimization algorithms. These updates include improvements in computational performance and scalability, extended support for multi-objective optimization, the introduction of range-based policies for continuous-variable optimization, the removal of environment-dependent components such as tightly coupled Cython modules, and compatibility with NumPy 2.

These improvements reduce the technical and organizational burden on users, enabling PHYSBO to be deployed across diverse computing environments and research workflows. By emphasizing portability and ease of integration while maintaining sufficient performance, PHYSBO version 3 is positioned as a sustainable research infrastructure for Bayesian optimization in physics and materials science.

Program summary/new version program summary
Program Title: PHYSBO
CPC Library link to program files: https://doi.org/10.17632/22d72yb6k6.2
Developer's repository link: https://github.com/issp-center-dev/PHYSBO
Licensing provisions: MPL-2.0
Programming language: Python
Journal reference of previous version: Y. Motoyama et al., Computer Physics Communications 278 (2022) 108405.
Does the new version supersede the previous version?: Yes
Reasons for the new version: PHYSBO version 3 was developed to improve usability, portability, and long-term sustainability in real research environments. While previous versions provided efficient Bayesian optimization for discrete candidate spaces, practical use revealed limitations related to software environment dependency, restricted support for continuous and multi-objective optimization, and licensing constraints that hindered in-

* Corresponding authors.
E-mail addresses: y-motoyama@issp.u-tokyo.ac.jp (Y. Motoyama), k-yoshimi@issp.u-tokyo.ac.jp (K. Yoshimi), tamura.ryo@nims.go.jp (R. Tamura).

https://doi.org/10.1016/j.cpc.2026.110368
Received 24 February 2026; Received in revised form 27 July 2026; Accepted 17 August 2026
Available online 29 August 2026
0010-4655/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

tegration into larger workflows. The new version addresses these issues to enable broader and more flexible deployment.

Summary of revisions: Major updates in PHYSBO version 3 include improvements in computational performance and scalability, enhanced support for multi-objective optimization, and the introduction of range-based policies for continuous-variable optimization. Environment-dependent components such as tightly coupled Cython modules have been removed, and compatibility with NumPy 2 has been added. In addition, the software license has been changed from GPL to MPL to facilitate easier integration into diverse research workflows.

Nature of problem: Optimization problems in physics and materials science often involve expensive objective function evaluations, such as first-principles calculations, large-scale simulations, or experimental measurements. Bayesian optimization is a powerful approach for reducing the number of evaluations required to identify optimal parameters. However, practical application of Bayesian optimization in research settings faces several challenges, including scalability to large search spaces, handling of multiple competing objectives, representation of continuous design variables, and software environment compatibility. In addition, researchers often need to incorporate domain-specific constraints and prior knowledge into the optimization process, which is not always supported by generic optimization frameworks.

Solution method: PHYSBO provides a Bayesian optimization framework tailored to physics and materials science applications. It supports optimization over user-defined candidate pools as well as range-based policies that enable direct optimization over continuous variables. PHYSBO version 3 extends support for multi-objective optimization using established approaches such as scalarization and non-dominated sorting. The software is designed to balance computational efficiency with practical usability by minimizing environment-dependent components and ensuring compatibility with modern numerical libraries. These design choices allow PHYSBO to be easily integrated into simulation and experimental workflows while maintaining sufficient performance for real-world optimization tasks.

Additional comments including restrictions and unusual features: PHYSBO is intended for use as a research library and can be imported into larger software frameworks and workflows. The adoption of the MPL license in version 3 reduces potential legal barriers associated with library usage and facilitates industry-academia collaboration. Performance-critical components are implemented to ensure practical efficiency, while prioritizing portability and ease of installation. PHYSBO does not impose restrictions on the form of objective functions, allowing users to integrate arbitrary external simulation or experimental codes.

1. Introduction

Bayesian optimization (BO) has become an essential tool for accelerating materials and physics research, where objective function evaluations are often expensive and time-consuming [1–3]. Typical applications include parameter optimization for first-principles calculations and large-scale simulations, as well as experimental design, where each evaluation entails substantial computational or experimental costs. In such settings, the practical usability of BO libraries is as important as their algorithmic performance.

PHYSBO is a BO library engineered for applications in physics and materials science [4]. A defining feature of PHYSBO is its ability to perform optimization over a predefined, discrete candidate pool, allowing researchers to seamlessly incorporate domain-specific knowledge, such as chemical intuition, physical constraints, and experimental limitations, into the exploration process. While this design is particularly advantageous for physics and materials science, the underlying framework is not restricted to these domains and can be applied more broadly to discrete and constrained optimization problems in various scientific and engineering fields, where evaluations are costly and prior knowledge is essential. This flexibility distinguishes it from generic BO frameworks that often struggle to integrate such nuanced priors. Furthermore, PHYSBO is designed to require minimal manual hyperparameter tuning, aiming for robust performance with little user intervention.

The library's utility is evidenced by its successful application in various areas, ranging from simulation-based atomic configuration exploration [5] and the discovery of next-generation magnetic materials [6] to the design of thermal metamaterials [7]. Its efficacy extends to a broad range of experimental domains as well, notably in the development of electrocatalysts [8], superconducting materials [9], polymer composites [10,11], drug discovery [12], antimicrobial peptides [13], quantum chemistry calculations [14], estimation of the exciton Hamiltonian [15], and fish feed [16]. Beyond these specific applications, PHYSBO has also been integrated into general-purpose data-driven frameworks such as 2DMAT [17], where it is employed for efficient parameter optimization in experimental and data-analysis workflows. More recently, as the paradigm of self-driving laboratories [18–20] gains momentum, PHYSBO has been adopted as the core optimization engine within the NIMO library [21], serving as the "algorithmic brain" for various autonomous experimental systems [22–25]. Furthermore, PHYSBO has been explored in community-driven events such as the AIMHack2024 hackathon, where researchers from materials science, information science, and related fields engaged in collaborative projects involving data-driven optimization workflows, highlighting its accessibility and practical utility beyond conventional research settings [26].

However, as PHYSBO has been increasingly used in real research environments, several limitations have become apparent. The current framework faces significant technical and strategic challenges, including a heavy reliance on specific environments like Cython and scalability bottlenecks during large-scale multi-objective optimization. While PHYSBO was originally centered on the paradigm of candidate pool-based optimization, there is an increasing demand for a unified tool capable of handling continuous design variables, thereby eliminating the operational overhead of switching to other software such as GPyOpt or Optuna. Furthermore, long-term sustainability and broader adoption have been constrained by compatibility issues with evolving numerical libraries and restrictive licensing conditions that limit its utility in industrial and collaborative environments. PHYSBO version 3 was developed to address these challenges, with a primary focus on improving usability, portability, and practical deployability, rather than introducing fundamentally new optimization algorithms. The updates include improvements in computational performance and scalability, enhanced support for multi-objective optimization, the introduction of range-based policies for continuous-variable optimization, the removal of environment-dependent components such as tightly coupled Cython modules, compatibility with NumPy 2, and a change of software license from GPL to MPL. The supporting evidence for these advantages is presented in the corresponding sections: the portability improvements are quantified in Section 3, and the scalability of the new multi-objective methods is benchmarked in Section 4 (Fig. 5).

In this paper, we present an overview of the major updates introduced in PHYSBO version 3 (3.2.1 released on 2026-02-06) and

discuss their impact on practical research workflows. Rather than focusing solely on algorithmic details, we emphasize how these changes reduce the burden on users and enable more flexible and sustainable use of BO in real-world physics and materials research.

2. License change: from GPL to MPL

Software licensing plays an important role in determining how research software can be used, integrated, and redistributed in practical research environments. Earlier versions of PHYSBO version 1 were distributed under the GNU General Public License (GPL), which ensures strong copyleft protection but can introduce limitations when the software is combined with other codes or libraries. In particular, the use of GPL-licensed software may cause concerns when PHYSBO is imported as a library into larger software frameworks or workflows. In some cases, simply importing a GPL-licensed package can raise legal or organizational issues, especially in collaborative projects involving industrial partners or mixed-license codebases. These concerns can hinder the adoption of otherwise useful research software, even when it is used purely as a dependency rather than as a standalone application.

To address these issues, PHYSBO version 2 adopted the Mozilla Public License (MPL). MPL provides a file-level copyleft mechanism that preserves openness of modifications to PHYSBO itself, while allowing the library to be imported and used within broader software ecosystems under different licenses. This change significantly improves compatibility with other scientific software and reduces potential legal barriers associated with library usage. The license change reflects a broader shift in the design philosophy of PHYSBO toward practical and sustainable deployment as a research infrastructure.

3. Software portability: removing environment dependency

In practical research environments, software portability and ease of installation are critical factors that strongly affect the usability of numerical optimization libraries. In earlier versions of PHYSBO, performance-critical components were implemented using Cython to accelerate Gaussian process–based computations. While this approach reduced computational overhead, it also introduced significant environment dependencies. In particular, building Cython extensions required specific compiler settings and library versions, which often caused installation difficulties on certain platforms, including Windows systems and heterogeneous high-performance computing environments. Such build- and environment-dependencies are a widely recognized obstacle to the portability and reproducibility of scientific software [27,28].

In PHYSBO version 3, this tight coupling between the core library and Cython-based implementations has been removed. The core functionality of PHYSBO is now provided as a pure Python package by default, eliminating the need for platform-specific compilation steps. As a result, installation is significantly simplified, and the library can be used more reliably across diverse research environments. Concretely, version 3 no longer requires a C/C++ compiler or the Cython build toolchain at installation: the core library installs as a pure-Python package whose only runtime dependencies are NumPy and SciPy, runs under Python 3.9 or later, and is compatible with both NumPy 1.x and NumPy 2.x. The number of non-Python, build-time dependencies is thereby reduced to zero, removing the platform-specific compilation step that was a frequent source of installation failures in earlier versions. The performance-critical parts previously implemented with Cython are further revised using vectorized NumPy arithmetic operations to keep the surrogate-model computations efficient. For typical Bayesian optimization workflows, in which the overall optimization cost is usually dominated by the objective-function evaluations rather than by the surrogate-model updates, the pure Python implementation is expected to remain sufficient for practical applications.

Another important update in PHYSBO version 3 is compatibility with NumPy 2.0 and later versions. As the Python scientific ecosystem evolves, maintaining compatibility with modern numerical libraries is essential for long-term sustainability. By supporting NumPy 2.0, PHYSBO can be seamlessly integrated into contemporary Python environments without relying on legacy configurations.

4. New multi-objective optimization methods

4.1. Multi-objective optimization problems and their scalarization

When multiple objective functions are to be optimized simultaneously, it is generally impossible to improve all objectives at once, and trade-offs among them inevitably arise. Such trade-offs are characterized by the concept of the Pareto front.

Let $\boldsymbol{y}_i = (y_{i,1}, \dots, y_{i,p})$ and $\boldsymbol{y}_j = (y_{j,1}, \dots, y_{j,p})$ denote objective vectors for two candidate points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, where all objectives are assumed to be maximized. We say that $\boldsymbol{y}_i$ dominates $\boldsymbol{y}_j$ if

$$\forall a\ y_{i,a} \geq y_{j,a} \quad \text{and} \quad \exists a\ y_{i,a} > y_{j,a}. \tag{1}$$

A candidate solution is said to be *Pareto-optimal* if it is not dominated by any other solution in the dataset. The set of all Pareto-optimal solutions forms the *Pareto front*, which represents the optimal trade-offs among competing objectives. Given a Pareto front $\{\boldsymbol{y}_i^{\mathrm{P}}\}$ and a reference point $\boldsymbol{y}^{\mathrm{ref}}$, the dominated region is defined as the union of hyperrectangles

$$\Omega_i = \prod_{a=1}^{p} \left[y_a^{\mathrm{ref}},\ y_{i,a}^{\mathrm{P}} \right]. \tag{2}$$

The hypervolume (HV) of the dominated region is commonly used as a performance metric to quantitatively assess the quality of a Pareto front.

In previous versions of PHYSBO, a separate surrogate model is constructed for each objective, and candidate points are selected by maximizing acquisition functions based on the hypervolume, such as the hypervolume probability of improvement (HVPI) and the expected hypervolume improvement (EHVI). However, the computational cost of hypervolume calculation grows exponentially with the number of objectives, which can become a significant bottleneck in practical applications. To address this issue, an alternative approach is to scalarize multiple objectives into a single objective function z_i and perform Bayesian optimization on this transformed objective. PHYSBO version 3 implements another `Policy` class based on this approach, `physbo.search.discrete_unified.Policy`. This class has a similar interface with the existing multi-objective policy, `physbo.search.discrete_multi.Policy`. The main difference is that the `discrete_unified.Policy.bayes_search` method takes an extra argument, `unify_method`, to specify the scalarization strategies for calculating z_i from $y_{i,a}$. In the present version of PHYSBO, we implement two such scalarization strategies: ParEGO and non-dominated sorting (NDS).

4.2. ParEGO

Pareto Efficient Global Optimization (ParEGO) is a scalarization-based approach for multi-objective Bayesian optimization, originally proposed by Knowles [29]. The key idea is to transform a p-objective optimization problem into a sequence of single-objective problems by means of randomized scalarization.

Let $\boldsymbol{y}_i = (y_{i,1}, y_{i,2}, \dots, y_{i,p})$ denote the vector of objective values evaluated at design point $\boldsymbol{x}_i$. ParEGO constructs a scalar objective function z_i from $\boldsymbol{y}_i$ as

$$z_i = \rho_{\mathrm{sum}} \sum_{a=1}^{p} w_a y_{i,a} + \rho_{\mathrm{min}} \min_a \left(w_a y_{i,a} \right), \tag{3}$$

where $\boldsymbol{w} = (w_1, \dots, w_p)$ is a weight vector satisfying $\sum_a w_a = 1$, and ρ_{sum} and ρ_{min} are hyperparameters that control the relative contributions of the weighted sum term and the weighted Chebyshev term, respectively[1].

[1] It should be noted that $\min_a$ is used in the weighted Chybyshev term here while $\max_a$ was used in the original form because PHYSBO maximizes z_i while Knowles minimized.

In PHYSBO, the weight vector $\boldsymbol{w}$ is either randomly sampled at each Bayesian optimization step or generated from a discretized simplex. Each objective $y_{i,a}$ is normalized using the minimum and maximum values observed in the current training data, so that all objectives contribute on a comparable scale. The resulting scalar objective z_i is then modeled using Gaussian process regression, and standard acquisition functions such as expected improvement (EI) are applied. By repeatedly changing the weight vector $\boldsymbol{w}$ across optimization steps, ParEGO implicitly explores different trade-offs among objectives and enables efficient approximation of the Pareto front, while avoiding the high computational cost associated with directly optimizing multi-objective acquisition functions such as hypervolume improvement.

In PHYSBO version 3, ParEGO is implemented as `physbo.search.unify.ParEGO`, and can be used through the unified multi-objective policy without modifying the underlying Bayesian optimization workflow.

4.3. NDS

Non-dominated sorting (NDS) is a fundamental technique in multi-objective optimization for classifying candidate solutions according to Pareto dominance relations. Unlike ParEGO, which relies on randomized scalarization of objective values, NDS does not introduce additional scalarization weights. Instead, it directly exploits dominance relations among objective vectors, making it particularly suitable for post-analysis and interpretation of multi-objective optimization results. NDS assigns a rank r_i to each solution i by recursively identifying Pareto fronts. All non-dominated points in the current dataset are assigned rank $r_i = 1$, corresponding to Pareto-optimal solutions. After removing these points, the non-dominated points of the remaining set are assigned rank $r_i = 2$. This procedure is iterated with increasing rank values $r_i = 3, 4, \ldots$ until the entire set of candidate solutions is exhaustively partitioned. In this manner, NDS decomposes the solution set into a sequence of Pareto fronts and provides an ordinal measure of the relative quality of candidate solutions. The rank obtained by NDS is converted into a scalar objective value as $z_i = 1/r_i$, which is maximized in the subsequent Bayesian optimization procedure. In this formulation, solutions on better Pareto fronts are assigned larger objective values, allowing multi-objective optimization to be reduced to a single-objective problem while preserving Pareto dominance information.

In PHYSBO version 3, NDS is implemented as `physbo.search.unify.NDS`, where the maximum rank used for scalarization can be controlled by the user. For points i with rank larger than this maximum, z_i is set to 0.

4.4. Numerical example

In the following, we present benchmark results comparing ParEGO and NDS with an existing method, HVPI, using three test functions: VLMOP2, KYMN, and Gaussians. [2] [3]

4.4.1. VLMOP2 benchmark

We first consider the VLMOP2 benchmark problem [31,32], which is widely used to evaluate multi-objective optimization algorithms. In this problem, a solution $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ is evaluated by two objective functions

$$y_1(\boldsymbol{x}) = 1 - \exp\left[-\sum_{i=1}^{2}\left(x_i - \frac{1}{\sqrt{2}}\right)^2\right], \tag{4}$$

$$y_2(\boldsymbol{x}) = 1 - \exp\left[-\sum_{i=1}^{2}\left(x_i + \frac{1}{\sqrt{2}}\right)^2\right]. \tag{5}$$

These objectives attain their minimum values of 0 at $\boldsymbol{x} = (\pm 1/\sqrt{2}, \pm 1/\sqrt{2})$, respectively, and are bounded above by 1. Since PHYSBO assumes a maximization setting, the optimization is performed on the negated objectives,

$$\tilde{\boldsymbol{y}}(\boldsymbol{x}) = \big(-y_1(\boldsymbol{x}), -y_2(\boldsymbol{x})\big),$$

so that larger values correspond to Pareto-improving solutions. Fig. 1 visualizes the two objective functions over the two-dimensional design space $[-2, 2] \times [-2, 2]$. Each objective exhibits a single smooth basin with its maximum located near $(\pm 1/\sqrt{2}, \pm 1/\sqrt{2})$, and the two optima are spatially separated. As a result, improving one objective necessarily degrades the other, giving rise to a smooth and continuous Pareto front.

The input space is discretized on a two-dimensional domain $[-2, 2] \times [-2, 2]$, and a Cartesian grid with 101 points per dimension is used, yielding 10,201 candidate solutions. Bayesian optimization is carried out with an initial 10 random sampling steps[4] followed by 40 optimization steps using either HVPI, NDS-based unification with $r_{\max} = 10$, or ParEGO-based scalarization with $\rho_{\max} = 1$ and $\rho_{\text{sum}} = 0.05$. Fig. 2 compares the Pareto fronts obtained by the three methods. HVPI achieves the largest dominated hypervolume[5] (HV = 0.3285), indicating the most extensive coverage of the smooth and continuous Pareto front. However, this performance comes at the cost of the longest computation time. In contrast, the NDS-based approach attains a comparable hypervolume (HV = 0.3092) while reducing the computational time by more than a factor of two. ParEGO exhibits the shortest computation time, but its Pareto front is sparse, resulting in a smaller hypervolume (HV = 0.3074). These results indicate that, for this benchmark with a smooth and continuous Pareto front, NDS provided an efficient alternative to HVPI by balancing solution quality and computational cost.

4.4.2. Kita–Yabumoto–Mori–Nishikawa benchmark

We next consider the Kita–Yabumoto–Mori–Nishikawa (KYMN) benchmark problem [33], which is a constrained bi-objective optimization problem originally proposed to test multi-objective algorithms under nonlinear feasibility conditions. In this problem, a solution $\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2$ is evaluated by two objective functions

$$y_1(x_1, x_2) = -x_1^2 + x_2^2, \tag{6}$$

$$y_2(x_1, x_2) = 0.5x_1 + x_2 + 1, \tag{7}$$

subject to the inequality constraints

$$g_1(x_1, x_2) = 6.5 - \frac{x_1}{6} - x_2 \geq 0, \tag{8}$$

$$g_2(x_1, x_2) = 7.5 - \frac{x_1}{2} - x_2 \geq 0, \tag{9}$$

$$g_3(x_1, x_2) = 30 - 5x_1 - x_2 \geq 0. \tag{10}$$

All objectives are maximized within the feasible region defined by these constraints. Fig. 3 visualizes the two objective functions over the design space. Unlike the VLMOP2 benchmark, both objective functions vary almost monotonically along different directions in the design space, and the feasible region is strongly restricted by the constraints. As a consequence, the set of feasible Pareto-optimal solutions occupies a narrow and irregular region in the objective space.

The candidate set is generated by discretizing the design space $[-7, 4] \times [-7, 4]$ on a two-dimensional grid, while excluding points that

[2] All the scripts used in this paper are available from ISSP data repository [30].

[3] These benchmarks were performed on Supercomputer System C (kugui) of the Institute for Solid State Physics, the University of Tokyo (AMD EPYC 7763, 2.45 GHz, 64 × 2 cores).

[4] It should be noted that common random samples are used for all three methods.

[5] The reference point of the dominated region is $\boldsymbol{y}^{\text{ref}} = (-1, -1)$.

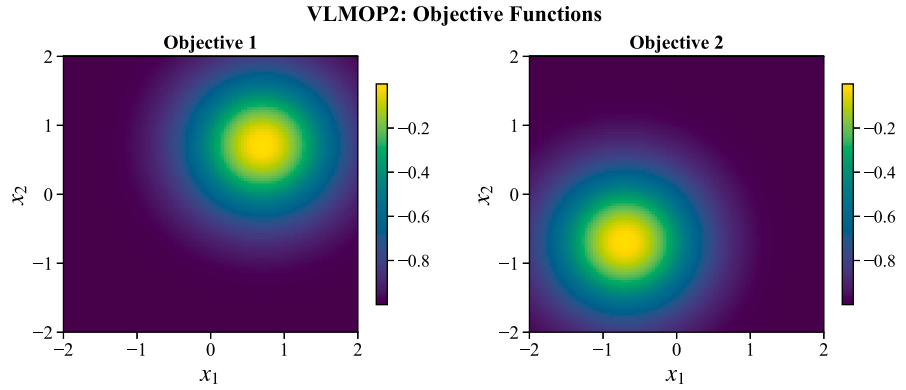


Fig. 1. Objective function landscapes of the VLMOP2 benchmark. Left: $-y_1(x_1, x_2)$. Right: $-y_2(x_1, x_2)$. The two objectives attain their maxima at different locations in the design space, inducing a trade-off between them.

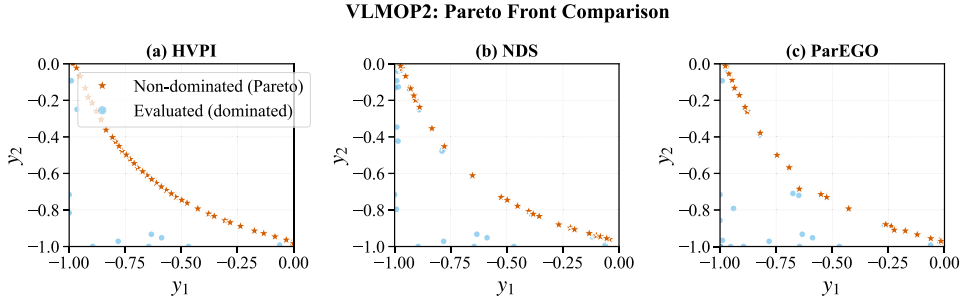


Fig. 2. Pareto front comparison for the VLMOP2 benchmark. Panels (a), (b), and (c) show the results obtained with HVPI, NDS, and ParEGO, respectively. Blue points denote evaluated solutions, while orange markers indicate the non-dominated solutions; a legend identifying the two marker types is shown in panel (a). HVPI yields the most complete coverage of the Pareto front, NDS provides a competitive approximation at a significantly lower computational cost, and ParEGO produces a sparse set of trade-off solutions.

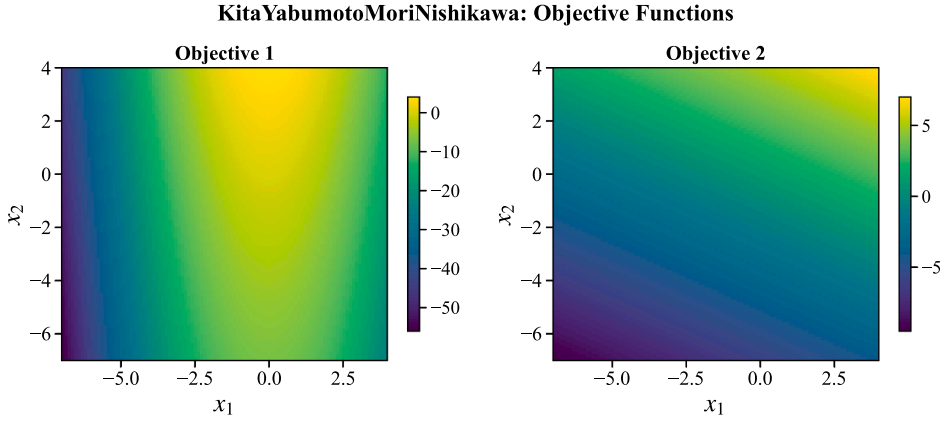


Fig. 3. Objective function landscapes of the Kita–Yabumoto–Mori–Nishikawa (KYMN) benchmark. Left: $y_1(x_1, x_2)$. Right: $y_2(x_1, x_2)$. The feasible region is implicitly restricted by nonlinear inequality constraints, leading to a highly constrained optimization landscape.

violate the constraints. Bayesian optimization is performed using the same protocol as in the VLMOP2 experiments, namely an initial 10 random sampling steps followed by 40 optimization steps based on either HVPI, NDS-based unification, or ParEGO-based scalarization. Fig. 4 compares the Pareto fronts obtained by the three methods. In contrast to the smooth Pareto front of VLMOP2, the KYMN benchmark yields a fragmented and highly nonconvex Pareto front. In this constrained setting, both NDS and ParEGO outperform HVPI in terms of the dominated hypervolume.[6] ParEGO achieves the largest hypervolume (HV = 977.4), closely followed by NDS (HV = 970.8), while HVPI yields a smaller value (HV = 932.1). From a computational perspective, both NDS and ParEGO require less than half the runtime of HVPI. These results demonstrate that, for this constrained benchmark with an irregular and nonconvex Pareto front, the unification-based approaches NDS and ParEGO provided more efficient and effective exploration than direct hypervolume-based optimization.

Together, the VLMOP2 and KYMN benchmarks highlight how the structure of the objective landscape and the presence of constraints critically affect the relative performance of multi-objective optimization strategies.

These observations, although based on two benchmarks, suggest the regime in which each approach may be preferable. HVPI directly maximizes the dominated hypervolume and tended to be most effective when

[6] The reference point of the dominated region is $\boldsymbol{y}^{\mathrm{ref}} = (-56.0, -9.5)$.

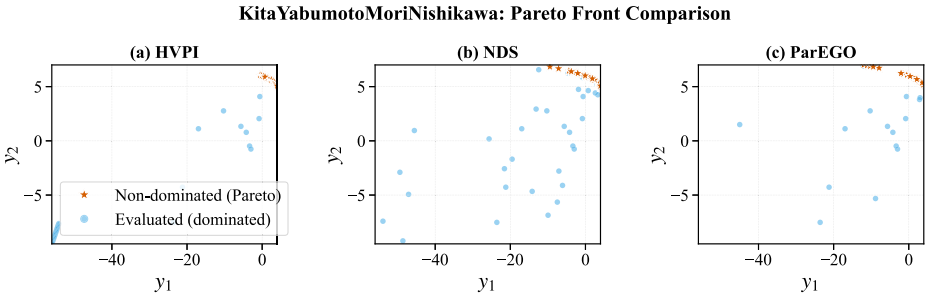


Fig. 4. Pareto front comparison for the Kita–Yabumoto–Mori–Nishikawa (KYMN) benchmark. Panels (a), (b), and (c) show the results obtained with HVPI, NDS, and ParEGO, respectively. Blue points denote evaluated solutions, and orange markers indicate the non-dominated solutions; a legend identifying the two marker types is shown in panel (a). ParEGO concentrates sampling on a narrow feasible Pareto-optimal region, whereas NDS yields a broader distribution of near-Pareto solutions. HVPI exhibits higher computational cost and reduced coverage in this constrained scenario.

the Pareto front is smooth, convex, and continuous, so that a single acquisition function captures the entire trade-off, as in the VLMOP2 benchmark. In this regime, and in particular when the number of objectives is small and the candidate set is not large, the hypervolume-based acquisition functions (HVPI, EHVI) remain affordable and are a natural choice. When, however, the feasible region is strongly constrained and the Pareto front is fragmented and nonconvex, as in the KYMN benchmark, the hypervolume-based acquisition becomes both more expensive and less effective at covering the disconnected feasible pieces, whereas the scalarization-based methods—ParEGO through its randomized weight vectors and NDS through its rank-preserving reduction—sample the feasible front more broadly and outperform HVPI. Moreover, as the number of objectives increases, the cost of the hypervolume computation grows exponentially while that of NDS and ParEGO remains essentially constant (Section 4.4.3), so that the scalarization-based methods become the more practical option for many-objective problems.

The scalarization-based methods do introduce additional hyperparameters, which could in principle increase the tuning burden. In our experiments, however, this burden appears limited. NDS introduces no scalarization weights and only a single, interpretable integer parameter, the maximum rank $r_{\max}$ (the number of Pareto layers retained), which we fixed to $r_{\max} = 10$ in all benchmarks. For ParEGO we adopt the values of $\rho_{\min}$ and ρ_{sum} recommended by Knowles [29], and the sensitivity analysis presented below (Section 4.4.4) shows that this default performs well on both benchmarks considered here: the hypervolume is essentially independent of the hyperparameters on the constrained KYMN problem, while on VLMOP2 the default is among the best-performing settings. Although the most suitable settings may ultimately depend on the specific problem, these observations suggest that the recommended default can be used as a sensible starting point without extensive problem-specific tuning, so that the additional hyperparameters introduce only a modest overhead relative to the gains in efficiency and coverage for constrained, nonconvex, or many-objective problems.

4.4.3. Computational time

Next, we consider a set of Gaussian objective functions to examine how the computational cost scales with the number of objectives. The optimization problem is defined as

$$y_a(\boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_a\|_2^2}{2\sigma_a^2}\right], \tag{11}$$

where $\boldsymbol{\mu}_a = \left[\cos\theta_a, \sin\theta_a\right]^{\mathrm{T}}$ denotes the center and σ_a the width of the ath objective function. In this benchmark, the centers and widths are randomly generated as

$$\theta_a \sim U(0, 2\pi), \tag{12}$$

$$\sigma_a \sim U(0.5, 1.0), \tag{13}$$

where $U(l, h)$ denotes the uniform distribution over the semi-closed interval $[l, h)$. The input space $[-2, 2] \times [-2, 2]$ is discretized into 101×101 grid points.

As in the previous benchmarks, we first perform 10 random sampling steps for initialization and then measure the computational time over 40 Bayesian optimization steps. To reduce the effect of the computational environment and random seeds, we conduct 10 independent trials, in which the centers and widths are regenerated. For a fair comparison among the three methods, the centers $\{\boldsymbol{\mu}_a\}$, widths $\{\sigma_a\}$, and the points $\{\boldsymbol{x}_i\}$ selected in the initial random sampling are identical for all methods within each trial.

Fig. 5 shows the mean and standard deviation of the computational time over 10 independent trials. As the number of objective functions increases, the computational costs of NDS and ParEGO remain almost constant, whereas that of HVPI grows exponentially. This exponential growth mainly originates from the computation of the hypervolume of the dominated region, which is required in the HVPI and EHVI methods.

4.4.4. Sensitivity to the ParEGO hyperparameters

Finally, to justify the choice of the ParEGO hyperparameters and to characterize their influence, we performed a sensitivity analysis on the VLMOP2 and KYMN benchmarks. Using the same protocol as above (10 random sampling steps followed by 40 Bayesian optimization steps), we varied $\rho_{\min} \in \{0.5, 1.0, 2.0\}$ and $\rho_{\text{sum}} \in \{0, 0.01, 0.05, 0.1, 0.25, 0.5, 1.0\}$. Because all settings share the same random seeds—and hence the same initial random samples—we ran 20 independent seeds and, in addition to the mean hypervolume, computed for each setting the *paired* per-seed difference relative to the default ($\rho_{\min} = 1$, $\rho_{\text{sum}} = 0.05$) together with its (pointwise, unadjusted) 95% confidence interval (CI); a setting whose CI excludes zero is taken as significantly different from the default at the pointwise level.

Fig. 6 shows the hypervolume *relative to the default* (the per-seed ratio to the default setting, averaged over the shared seeds, so that the default lies at 1) as a function of ρ_{sum} for each value of $\rho_{\max}$, with standard-error-of-the-mean error bars and a symmetric-log horizontal scale so that the small ρ_{sum} values are resolved; the dashed vertical line marks the default $\rho_{\text{sum}} = 0.05$. The two benchmarks behave differently. On VLMOP2 (left) the default is a near-optimal choice: no setting yields a significantly larger hypervolume, and settings with small $\rho_{\text{sum}} \lesssim 0.05$ are statistically indistinguishable from the default (their paired 95% CIs include zero), whereas settings with large $\rho_{\text{sum}}/\rho_{\min} \geq 0.25$ are significantly worse, by up to about 40% (the intermediate value $\rho_{\text{sum}} = 0.1$ is borderline). On the strongly constrained KYMN problem (right), by contrast, the hypervolume is practically insensitive to both hyperparameters: the grid-mean spread is only 0.07% of the default value, so that even the differences that are statistically resolvable are negligible in magnitude. This weak

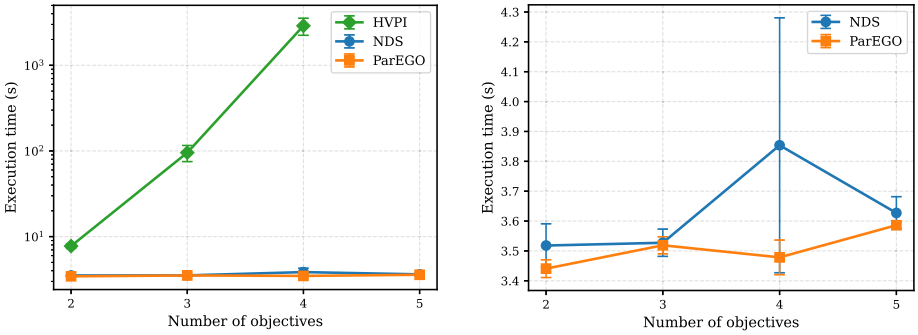


Fig. 5. Dependence of the computational time for 40 Bayesian optimization steps on the number of objective functions p for each method. The mean and standard deviation over 10 independent runs are shown. The left panel shows all three methods (HVPI, NDS, and ParEGO), while the right panel is an enlarged view of the left panel showing only the NDS and ParEGO results, whose execution time remains nearly constant as p increases.

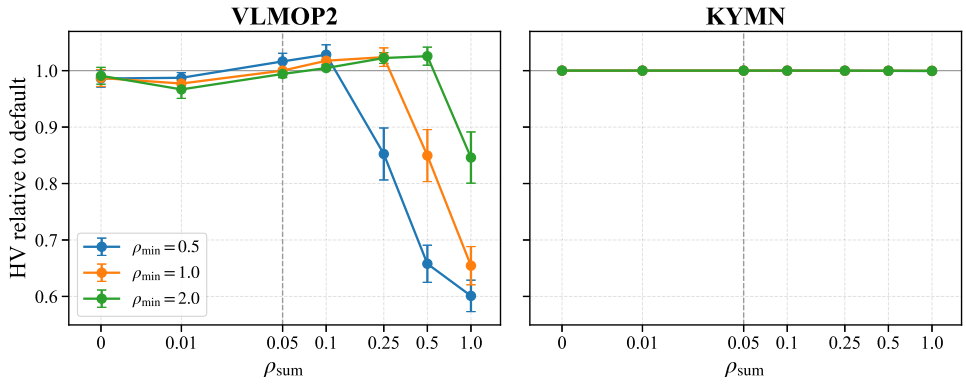


Fig. 6. Sensitivity of the ParEGO hypervolume to the scalarization hyperparameters $\rho_{\min}$ and ρ_{sum} for the VLMOP2 (left) and KYMN (right) benchmarks. The vertical axis is the hypervolume *relative to the default* ($\rho_{\min} = 1$, $\rho_{\text{sum}} = 0.05$), computed per shared random seed and averaged over 20 seeds, so that the default lies exactly at 1 (horizontal line); the error bars denote the standard error of this mean ratio, and the horizontal axis uses a symmetric-log scale so that the small ρ_{sum} values are resolved. The dashed vertical line marks the default $\rho_{\text{sum}} = 0.05$. With the two benchmarks on a common relative scale the contrast is clear: on VLMOP2 the default is near-optimal and the hypervolume degrades by up to about 40% for large ρ_{sum}, whereas on KYMN the hypervolume is practically independent of the hyperparameters (all settings within 0.07% of the default).

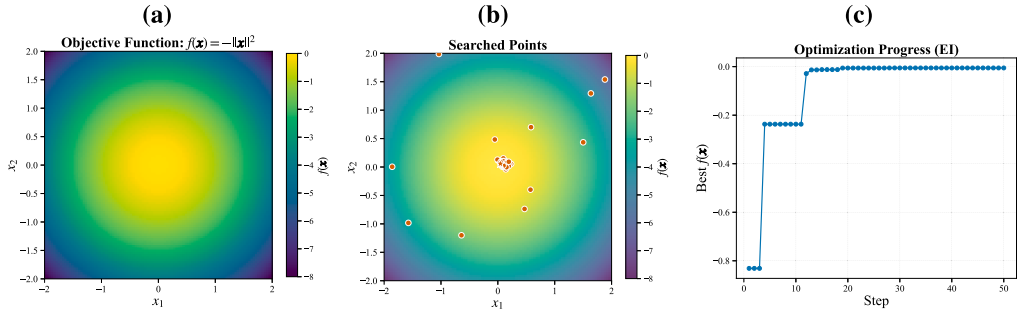


Fig. 7. Bayesian optimization in a continuous parameter space using the range policy of PHYSBO. (a) Objective function landscape $f(\boldsymbol{x}) = -\|\boldsymbol{x}\|^2$ over $[-2, 2] \times [-2, 2]$. (b) Locations of evaluated points during optimization, showing adaptive concentration near the optimum. (c) Optimization progress in terms of the best-so-far objective value using the expected improvement (EI) acquisition function.

dependence is likely a consequence of the problem structure: the feasible Pareto-optimal region is narrow and tightly constrained, so that different scalarizations converge to nearly the same few solutions, and the large hypervolume baseline set by the distant reference point further compresses the relative differences. Taken together, these results support the use of the recommended default as a robust, near-optimal starting point: on both benchmarks it is never outperformed by a practically meaningful margin, and the only clear guidance is to keep ρ_{sum} small, as originally recommended by Knowles [29], rather than to tune it precisely for each problem. This supports the reproducibility of the reported results.

5. Range policy

5.1. Motivation and design concept

In many optimization problems encountered in physics and materials science, decision variables are naturally expressed as continuous parameters rather than elements of a finite, predefined candidate pool. While earlier versions of PHYSBO focused on optimization over discrete candidate sets, such an approach can impose a significant burden on users when the search space is high-dimensional or inherently continuous.

To address this limitation, PHYSBO introduces the *range policy*, which provides a unified framework for exploring continuous parameter spaces defined by lower and upper bounds. The range policy is initialized by explicitly providing these bounds, for example, as

```
policy = physbo.search.range.Policy(min_X=min_X, max_X=max_X)
```

where `min_X` and `max_X` have the same dimensionality and collectively define a hyper-rectangular domain in the continuous parameter space. Instead of requiring users to discretize the search space in advance, the range policy allows optimization problems to be formulated directly in terms of parameter ranges. This design significantly reduces problem setup complexity and aligns more naturally with typical formulations of simulation and experimental parameters.

In Bayesian optimization, the next candidate point is proposed by maximizing an acquisition function, which itself constitutes an auxiliary maximization problem that must be solved at every iteration. In the case of a discrete search space, this problem can be solved straightforwardly by evaluating the acquisition function over all remaining points in the candidate pool and selecting the maximizer. In contrast, for a continuous search space, an additional numerical optimization solver is required to maximize the acquisition function.

The `range.Policy.bayes_search` function in PHYSBO is designed to accept a user-specified optimization routine for acquisition function maximization, allowing the optimization strategy to be flexibly chosen. The current version of PHYSBO provides some optimizers as described in the following subsections. For example, the optimizer by random sampling is initialized as

```
optimizer = physbo.search.optimize.random.Optimizer(min_X, max_X, nsamples)
```

and passed to `bayes_search` as

```
res = policy.bayes_search(optimizer=optimizer)
```

As well as the single-objective problem, the multi-objective versions of the range solver are implemented as `physbo.search.range_multi.Policy` and `physbo.search.range_unified.Policy`.

The range policy enhances the usability of PHYSBO by decoupling the definition of the search space from the choice of optimization algorithm. Users can define parameter ranges once and flexibly apply different search strategies without restructuring their workflows.

5.2. Random sampling

In PHYSBO, random search is employed as the default strategy for acquisition function optimization: a fixed number of candidate points is sampled uniformly at random within the predefined parameter ranges, and the point that maximizes the acquisition function value is selected. By default, the number of random samples used for acquisition optimization is set to `nsamples = 1000`, which provides a balance between robustness and computational cost. The value of `nsamples` can be adjusted by explicitly specifying a random optimizer via `physbo.search.optimize.random.Optimizer` and passing it to the `optimizer` keyword of the `bayes_search` function. This configurability allows users to tune the trade-off between exploration accuracy and computational overhead according to the problem size and available computational resources.

As an initialization step for optimization under the range policy, the default exploration strategy is random sampling within the specified parameter ranges. Users define the continuous search space by specifying the lower and upper bounds for each parameter through the vectors `min_X` and `max_X`. Each element of these vectors corresponds to one dimension of the search space and represents the minimum and maximum allowable values of the associated parameter.

Random sampling provides an efficient and unbiased initial exploration when little prior information about the objective function is available, while ensuring broad coverage of the search space with minimal user intervention. This choice reflects common practices in the design of experiments, where an initial random or space-filling sampling phase is used to gather information before applying more sophisticated optimization methods.

5.3. A wrapper of another optimization solver library, ODAT-SE

ODAT-SE (Open Data Analysis Tool for Science and Engineering) [17,34] is a Python framework for solving optimization problems with several algorithms:

- `exchange`: replica exchange Monte Carlo, effective for rugged or multimodal landscapes [35].
- `pamc`: population annealing Monte Carlo, enabling efficient exploration of complex parameter spaces [36].
- `minsearch`: the Nelder–Mead simplex method, suitable for smooth objective functions [37,38].
- `mapper`: grid-based search providing systematic coverage and high reproducibility.
- `bayes`: Bayesian optimization based on surrogate modeling and acquisition functions.

PHYSBO provides an optimizer class wrapping ODAT-SE as `physbo.search.optimize.odatse.Optimizer`. In practical workflows using ODAT-SE, the optimization algorithm is explicitly specified by the variable `algorithm_name`, and the algorithm's hyperparameters like the number of Monte Carlo samplings are described using a dictionary representation equivalent to the `[algorithm]` section of the ODAT-SE input file. PHYSBO provides a utility function, `physbo.search.optimize.odatse.default_alg_dict`, which generates a template dictionary with default hyperparameters for the selected algorithm. For example, a dictionary representing the default settings for the replica exchange Monte Carlo method can be obtained as

```
odatse_alg_dict = physbo.search.optimize.odatse.default_alg_dict(
    min_X=min_X, max_X=max_X, algorithm_name="exchange"
)
```

and users can modify parameters as they want. This design enables seamless switching between optimization strategies by modifying only the value of `algorithm_name`, while keeping the search space definition unchanged. Once the parameter dictionary is prepared, an optimizer is initialized as

```
optimizer = physbo.search.optimize.odatse.Optimizer(odatse_alg_dict)
```

and then passed to a range policy in the same way as the random sampling.

5.4. Numerical example

We demonstrate Bayesian optimization in a continuous parameter space using the range policy of PHYSBO, following the implementation shown in the accompanying sample code. As a simple yet illustrative example, we consider the optimization of a smooth two-dimensional objective function

$$f(\boldsymbol{x}) = -\|\boldsymbol{x}\|^2, \qquad \boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2, \tag{14}$$

which has a unique maximum at $\boldsymbol{x} = (0, 0)$ and represents a typical single-basin energy landscape encountered in physics and materials science. The objective function landscape over the domain $[-2, 2] \times [-2, 2]$ is shown in Fig. 7(a). The function value decreases smoothly away from the origin, providing a well-behaved test case for continuous Bayesian optimization.

The continuous search space is specified by lower and upper bounds for each parameter, and the range policy is initialized accordingly. An initial exploration phase is performed using random sampling within the

specified parameter ranges. Subsequently, Bayesian optimization is applied using the expected improvement (EI) acquisition function, which is optimized by random sampling in the continuous parameter space.

Fig. 7(c) shows the optimization progress in terms of the best objective value found so far. After an initial rapid improvement during the random sampling phase, the Bayesian optimization stage efficiently refines the search and quickly converges toward the global optimum at $f(\boldsymbol{x}) = 0$. This behavior illustrates the effectiveness of acquisition-driven search in continuous spaces without discretization. Fig. 7(b) visualizes the locations of the evaluated points in the parameter space. While the initial random samples are broadly distributed, subsequent evaluations are concentrated near the optimum. This demonstrates how Bayesian optimization adaptively focuses sampling in promising regions of the continuous search space. Together, these results highlight the ability of PHYSBO to perform efficient Bayesian optimization directly in continuous parameter spaces using the range policy, avoiding the need for explicit discretization and enabling scalable optimization in higher-dimensional settings.

6. Conclusion

In this paper, we have presented the major updates introduced in PHYSBO version 3, with a particular emphasis on usability, portability, and practical deployment in real-world physics and materials research. Rather than focusing on the introduction of fundamentally new optimization algorithms, PHYSBO version 3 represents a systematic refinement of the library to better support research workflows.

The updates described in this work address several long-standing challenges encountered in practical Bayesian optimization. Improvements in computational performance and scalability enable stable use in large-scale optimization problems. Enhanced support for multi-objective optimization allows users to handle realistic research scenarios involving competing objectives. The introduction of range-based policies extends PHYSBO to continuous-variable optimization, improving flexibility. In addition, the removal of environment-dependent components, compatibility with modern numerical libraries such as NumPy 2, and the transition to a more permissive software license collectively improve portability, sustainability, and ease of integration.

Taken together, these changes reposition PHYSBO from a specialized optimization tool to a more general research infrastructure for Bayesian optimization. By lowering technical and organizational barriers, PHYSBO version 3 enables researchers to more easily incorporate Bayesian optimization into computational and experimental studies, facilitating rapid iteration, reproducibility, and collaboration. Notably, the removal of the Cython dependency significantly enhances accessibility for Windows users. Given that many experimental instruments operate on Windows environments, this update facilitates the transition of Bayesian optimization from purely cyber-space simulations to physical-world experimental research. These improvements ensure that PHYSBO is more easily integrated into modern AI-robotic research, positioning it to support future developments in automated and data-driven scientific discovery.

CRediT authorship contribution statement

Yuichi Motoyama: Writing – review & editing, Writing – original draft, Software; **Kazuyoshi Yoshimi:** Writing – review & editing, Writing – original draft, Visualization, Software, Project administration; **Tatsumi Aoyama:** Writing – review & editing, Writing – original draft, Validation, Software; **Kei Terayama:** Writing – review & editing, Methodology; **Koji Tsuda:** Supervision, Methodology, Conceptualization; **Ryo Tamura:** Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Kazuyoshi Yoshimi reports financial support was provided by Government of Japan Ministry of Education Culture Sports Science and Technology. Kazuyoshi Yoshimi reports financial support was provided by Japan Society for the Promotion of Science. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We wish to thank Naoki Kawashima and Tsuyoshi Ueno for their highly useful discussions. PHYSBO was developed under the support of the “Project for advancement of software usability in materials science” [39] in fiscal years 2020 and 2025 by the Institute for Solid State Physics, University of Tokyo. This work is partially supported by JSPS KAKENHI Grant Nos. 25H01403 and 25K22013. Additional support was provided by the Ministry of Education, Culture, Sports, Science, and Technology (MEXT) through the Data Creation and Utilization Type Material Research and Development Project (JPJ010337, Grant Number: JPMXP1122683430 and JPMXP1121467561).

Data availability

All the scripts are available from ISSP Data repository (see the first footnote of the manuscript).

References

[1] T. Ueno, T.D. Rhone, Z. Hou, T. Mizoguchi, K. Tsuda, COMBO: an efficient Bayesian optimization library for materials science, Mater. Discov. 4 (2016) 18–21. https://doi.org/10.1016/j.md.2016.04.001

[2] S. Ju, T. Shiga, L. Feng, Z. Hou, K. Tsuda, J. Shiomi, Designing nanostructures for phonon transport via Bayesian optimization, Phys. Rev. X 7 (2017) 021024. https://doi.org/10.1103/PhysRevX.7.021024

[3] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, T. Oguchi, Crystal structure prediction accelerated by Bayesian optimization, Phys. Rev. Mater. 2 (2018) 013803. https://doi.org/10.1103/PhysRevMaterials.2.013803

[4] Y. Motoyama, R. Tamura, K. Yoshimi, K. Terayama, T. Ueno, K. Tsuda, Bayesian optimization package: PHYSBO, Comput. Phys. Commun. 278 (2022) 108405. https://doi.org/10.1016/j.cpc.2022.108405

[5] A. Kusaba, T. Kuboyama, K. Kawka, P. Kempisty, Y. Kangawa, PyAPX: Python toolkit for atomic configuration pattern exploration, 2025, arXiv:2511.17972

[6] K. Nakamura, D. Furuya, Y. Miura, T. Yamazaki, A.L. Foggiatto, Y. Iwasaki, M. Kotsugi, Efficient discovery of large magnetic anisotropy with thermodynamically stable materials via multi-objective Bayesian optimization, Sci. Technol. Adv. Mater.: Methods 5 (1) (2025). https://doi.org/10.1080/27660400.2025.2485025

[7] W. Xi, Y.-J. Lee, S. Yu, Z. Chen, J. Shiomi, S.-K. Kim, R. Hu, Ultrahigh-efficient material informatics inverse design of thermal metamaterials for visible-infrared-compatible camouflage, Nat. Commun. 14 (1) (2023). https://doi.org/10.1038/s41467-023-40350-6

[8] K. Sakaushi, W. Hoisang, R. Tamura, Human–machine collaboration for accelerated discovery of promising oxygen evolution electrocatalysts with on-demand elements, ACS Cent. Sci. 9 (12) (2023) 2216–2224. https://doi.org/10.1021/acscentsci.3c01009

[9] S. Yamazaki, T. Horide, Y. Yoshida, Accelerated process development of $YBa_2Cu_3O_7$ film with Bayesian optimization augmented by transferred data, ACS Appl. Electron. Mater. 7 (14) (2025) 6582–6591. https://doi.org/10.1021/acsaelm.5c00884

[10] B. Xu, T.A. Sultana, K. Kitai, J. Guo, T. Seki, R. Tamura, K. Tsuda, J. Shiomi, Experiment-in-loop interactive optimization of polymer composites for "5G-and-beyond" communication technologies, Mater. Horiz. 12 (10) (2025) 3332–3340. https://doi.org/10.1039/d4mh01606h

[11] Y. Amamoto, C. Koganemaru, K. Kojio, A. Takahara, S. Yamamoto, K. Okazawa, Y. Tsuji, T. Aritake, K. Terayama, A machine learning approach to designing and understanding tough, degradable polyamides, Npj Comput. Mater. 11 (1) (2025) 198. https://doi.org/10.1038/s41524-025-01696-1

[12] T. Yoshizawa, S. Ishida, T. Sato, M. Ohta, T. Honma, K. Terayama, A data-driven generative strategy to avoid reward hacking in multi-objective molecular design, Nat. Commun. 16 (1) (2025) 2409. https://doi.org/10.1038/s41467-025-57582-3

[13] Y. Murakami, S. Ishida, Y. Demizu, K. Terayama, Design of antimicrobial peptides containing non-proteinogenic amino acids using multi-objective Bayesian optimisation, Digit. Discov. 2 (5) (2023) 1347–1353. https://doi.org/10.1039/D3DD00090G

[14] K. Terayama, Y. Osaki, T. Fujita, R. Tamura, M. Naito, K. Tsuda, T. Matsui, M. Sumita, Koopmans' theorem-compliant long-range corrected (KTLC) density functional mediated by black-box optimization and data-driven prediction for organic molecules, J. Chem. Theory Comput. 19 (19) (2023) 6770–6781. https://doi.org/10.1021/acs.jctc.3c00764
[15] W. Shimooka, H. Kitoh-Nishioka, S. Itoh, A. Kimura, Improvement of exciton hamiltonians for photosynthetic reaction centers by Bayesian optimization, ACS Omega 10 (24) (2025) 26090–26097. https://doi.org/10.1021/acsomega.5c03026
[16] K. Terayama, S. Soma, H. Furuita, T. Shimizu, T. Iwasaki, K. Sakata, K.-i. Akagi, T. Asakura, T. Sanda, T. Yamaguchi, et al., Data-driven feed optimization for sustainable aquaculture through integrated omics analysis and two-stage Bayesian optimization, bioRxiv (2025) 2025–01. https://doi.org/10.1101/2025.01.27.635187
[17] Y. Motoyama, K. Yoshimi, I. Mochizuki, H. Iwamoto, H. Ichinose, T. Hoshi, Data-analysis software framework 2DMAT and its application to experimental measurements for two-dimensional material structures, Comput. Phys. Commun. 280 (2022) 108465. https://doi.org/10.1016/j.cpc.2022.108465
[18] B.P. MacLeod, F.G.L. Parlane, T.D. Morrissey, F. Häse, L.M. Roch, K.E. Dettelbach, R. Moreira, L.P.E. Yunker, M.B. Rooney, J.R. Deeth, V. Lai, G.J. Ng, H. Situ, R.H. Zhang, M.S. Elliott, T.H. Haley, D.J. Dvorak, A. Aspuru-Guzik, J.E. Hein, C.P. Berlinguette, Self-driving laboratory for accelerated discovery of thin-film materials, Sci. Adv. 6 (20) (2020) eaaz8867. https://doi.org/10.1126/sciadv.aaz8867
[19] G. Tom, S.P. Schmid, S.G. Baird, Y. Cao, K. Darvish, H. Hao, S. Lo, S. Pablo-García, E.M. Rajaonson, M. Skreta, N. Yoshikawa, S. Corapi, G.D. Akkoc, F. Strieth-Kalthoff, M. Seifrid, A. Aspuru-Guzik, Self-driving laboratories for chemistry and materials science, Chem. Rev. 124 (16) (2024) 9633–9732. https://doi.org/10.1021/acs.chemrev.4c00055
[20] N. Yoshikawa, Y. Asano, D.N. Futaba, K. Harada, T. Hitosugi, G.N. Kanda, S. Matsuda, Y. Nagata, K. Nagato, M. Naito, T. Natsume, K. Nishio, K. Ono, H. Ozaki, W. Shin, J. Shiomi, K. Shizume, K. Takahashi, S. Takeda, I. Takeuchi, R. Tamura, K. Tsuda, Y. Ushiku, Self-driving laboratories in Japan, Digit. Discov. 4 (6) (2025) 1384–1403. https://doi.org/10.1039/d4dd00387j
[21] R. Tamura, K. Tsuda, S. Matsuda, NIMS-OS: an automation software to implement a closed loop between artificial intelligence and robotic experiments in materials science, Sci. Technol. Adv. Mater.: Methods 3 (1) (2023). https://doi.org/10.1080/27660400.2023.2232297
[22] S. Matsuda, K. Nishioka, S. Nakanishi, High-throughput combinatorial screening of multi-component electrolyte additives to improve the performance of Li metal secondary batteries, Sci. Rep. 9 (1) (2019). https://doi.org/10.1038/s41598-019-42766-x
[23] S. Akiyama, H. Akitsu, R. Tamura, W. Matsuoka, S. Maeda, K. Tsuda, Y. Nagata, Autonomous optimization of discrete reaction parameters: mono-functionalization of a bifunctional substrate via a Suzuki-Miyaura reaction (2024). https://doi.org/10.26434/chemrxiv-2024-bnj6p-v2
[24] R. Tamura, H. Taketa, S. Murata, D. Ryuno, T. Yokota, K. Tsuda, S. Matsuda, Seamless integration of legacy robotic systems into a self-driving laboratory via NIMO: a case study on liquid handler automation, Sci. Technol. Adv. Mater.: Methods 5 (1) (2025). https://doi.org/10.1080/27660400.2025.2565144
[25] R. Toyama, R. Tamura, S. Matsuda, Y. Iwasaki, Y. Sakuraba, Autonomous closed-loop exploration of composition-spread films for the anomalous Hall effect, Npj Comput. Mater. 11 (1) (2025). https://doi.org/10.1038/s41524-025-01828-7
[26] T. Misawa, A. Koizumi, R. Tamura, K. Yoshimi, Exploring utilization of generative AI for research and education in data-driven materials science, Sci. Technol. Adv. Mater.: Methods 5 (1) (2025) 2535956. https://doi.org/10.1080/27660400.2025.2535956
[27] C. Collberg, T.A. Proebsting, Repeatability in computer systems research, Commun. ACM 59 (3) (2016) 62–69. https://doi.org/10.1145/2812803
[28] G.K. Sandve, A. Nekrutenko, J. Taylor, E. Hovig, Ten simple rules for reproducible computational research, PLoS Comput. Biol. 9 (10) (2013) e1003285. https://doi.org/10.1371/journal.pcbi.1003285
[29] J. Knowles, ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, IEEE Trans. Evol. Comput. 10 (1) (2006) 50–66. https://doi.org/10.1109/TEVC.2005.851274
[30] PHYSBO developers, PHYSBO gallery, 2026, https://isspns-gitlab.issp.u-tokyo.ac.jp/physbo-dev/physbo-gallery. Accessed: 2026-02.
[31] C.M. Fonseca, P.J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evol. Comput. 3 (1) (1995) 1–16. https://doi.org/10.1162/evco.1995.3.1.1
[32] D.A. van Veldhuizen, G.B. Lamont, Multiobjective evolutionary algorithm test suites, in: Proceedings of the 1999 ACM Symposium on Applied Computing, SAC99, ACM, 1999, p. 351–357. https://doi.org/10.1145/298151.298382
[33] H. Kita, Y. Yabumoto, N. Mori, Y. Nishikawa, Multi-objective optimization by means of the thermodynamical genetic algorithm, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature — PPSN IV, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp. 504–512. https://doi.org/10.1007/3-540-61723-x_1014
[34] ODAT-SE GitHub repository, https://github.com/issp-center-dev/ODAT-SE. Accessed: 2025-12-24.
[35] K. Hukushima, K. Nemoto, Exchange Monte Carlo method and application to spin glass simulations, J. Phys. Soc. Jpn. 65 (6) (1996) 1604–1608. https://doi.org/10.1143/JPSJ.65.1604
[36] K. Hukushima, Y. Iba, Population annealing and its application to a spin glass, in: AIP Conference Proceedings, vol. 690, American Institute of Physics, 2003, pp. 200–206. https://doi.org/10.1063/1.1632130
[37] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (4) (1965) 308–313. https://doi.org/10.1093/comjnl/7.4.308
[38] M.H. Wright, Direct search methods: once scorned, now respectable, in: D.F. Griffiths, G.A. Watson (Eds.), Numerical Analysis 1995, vol. 344 of *Pitman Research Notes in Mathematics Series*, Addison-Wesley, 1996, pp. 191–208.
[39] K. Yoshimi, Y. Motoyama, T. Aoyama, M. Kawamura, N. Kawashima, Project for advancement of software usability in materials science, Sci. Technol. Adv. Mater.: Methods 5 (1) (2025) 2564055. https://doi.org/10.1080/27660400.2025.2564055