# PyBADS: Fast and robust black-box optimization in Python

Gurjeet Sangra Singh [1,3]¶ and **Luigi Acerbi** [2]¶

1 University of Geneva **2** University of Helsinki **3** University of Applied Sciences and Arts Western Switzerland (HES-SO) ¶ Corresponding author

Summary

PyBADS is a Python implementation of the Bayesian Adaptive Direct Search (BADS) algorithm for fast and robust *black-box* optimization (Acerbi & Ma, 2017). BADS is an optimization algorithm designed to efficiently solve difficult optimization problems where the objective function is rough (non-convex, non-smooth), mildly expensive (e.g., the function evaluation requires more than 0.1 seconds), possibly noisy, and gradient information is unavailable. With BADS, these issues are well addressed, making it an excellent choice for fitting computational models using methods such as maximum-likelihood estimation. The algorithm scales efficiently to black-box functions with up to $D \approx 20$ continuous input parameters and supports bounds or no constraints. PyBADS comes along with an easy-to-use Pythonic interface for running the algorithm and inspecting its results. PyBADS only requires the user to provide a Python function for evaluating the target function, and optionally other constraints.

Extensive benchmarks on both artificial test problems and large real model-fitting problems models drawn from cognitive, behavioral and computational neuroscience, show that BADS performs on par with or better than many other common and state-of-the-art optimizers (Acerbi & Ma, 2017), making it a general model-fitting tool which provides fast and robust solutions.

Statement of need

Many optimization problems in science and engineering involve complex and expensive simulations or numerical approximations such that the target function can only be evaluated at a point with moderate to high cost, possibly yielding stochastic outcomes, and gradients are unavailable (or exceedingly expensive) – the typical *black-box* scenario. There is a large landscape of derivative-free optimization algorithms for tackling black-box problems (Rios & Sahinidis, 2013), many of which follow variants of direct-search methods (Abramson et al., 2009; Audet et al., 2021; Audet & Dennis, 2006; Deng & Ferris, 2006). Despite their theoretical guarantees, direct search methods require a large number of function evaluations and have limited support for handling stochastic targets.

Conversely, Bayesian Optimization (BayesOpt) is a recently popular family of methods that has shown effectiveness in solving *very* costly black-box problems in machine learning and engineering with very few, possibly noisy, function evaluations (Agnihotri & Batra, 2020; Garnett, 2023; Shahriari et al., 2016). However, BayesOpt requires specific technical knowledge to be implemented or tuned beyond simple tasks, since vanilla BayesOpt applied to complex real-world problems can be strongly affected by deviations from the algorithm's assumptions (model misspecification), a problem rarely dealt with in current implementations. Moreover, traditional BayesOpt methods assume *highly expensive* target functions (e.g., with evaluation costs of hours or more), whereas many computational models might only have a *moderate*

42 evaluation cost (e.g., from a fraction of a second to a few seconds), meaning that the
43 optimization algorithm should add only a relatively small overhead.

44 PyBADS addresses all these problems as a fast hybrid algorithm that combines the strengths of
45 BayesOpt and the Mesh Adaptive Direct Search (Audet & Dennis, 2006) method. In contrast
46 to other black-box optimization algorithms, PyBADS is both *fast* in terms of wall-clock time
47 and *sample-efficient* in terms of the number of target evaluations (typically of the order of a
48 few hundred), with support for noisy targets. Moreover, PyBADS does not require any specific
49 tuning and runs off-the-shelf with its well-modularized Python API.

Method

51 PyBADS follows the Mesh Adaptive Direct Search (MADS, Audet & Dennis, 2006) schema
52 for minimizing the given objective function. The algorithm alternates between a series of fast
53 local Bayesian Optimization (BayesOpt) steps, referred to as *search stage*, and systematic
54 exploration of the mesh space in a neighborhood of the current point, known as *poll stage*,
55 based on the MADS poll method (Audet & Dennis, 2006); see Figure 1. Briefly:

56 ▪ In the poll stage, points are evaluated on a mesh by taking steps in one (non-orthogonal)
57   direction at a time, until an improvement is found or all directions have been tried. The
58   step size is doubled in case of success, halved otherwise.
59 ▪ In the search stage, a Gaussian process (GP) surrogate model (Rasmussen & Williams,
60   2006) of the target function is fit to a local subset of the points evaluated so far. New
61   points to evaluate are quickly chosen according to a *lower confidence bound* strategy
62   that trades off between exploration of uncertain regions (high GP uncertainty) and
63   exploitation of promising solutions (low GP mean). The search switches back to the poll
64   stage after repeated failures to find an improvement over the current point.
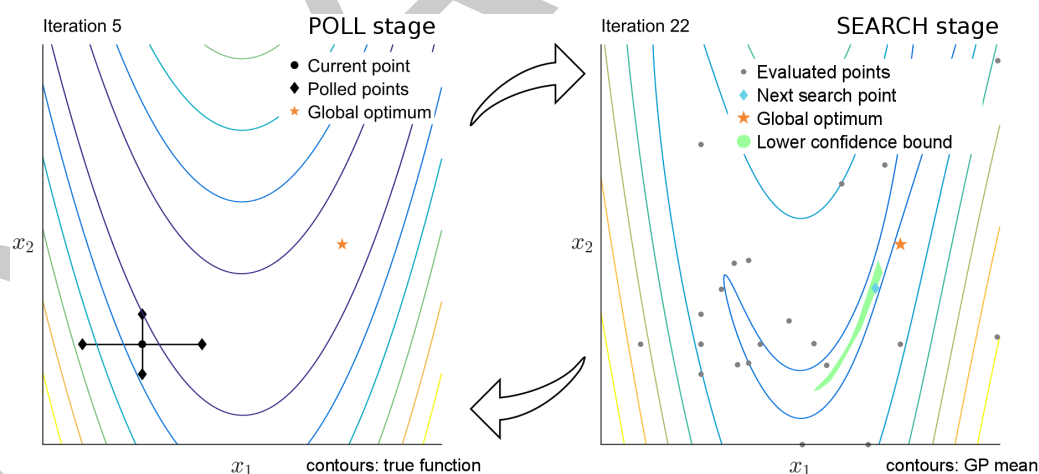
Figure 1: Contour plots and PyBADS exploration of a two-dimensional Rosenbrock function. Lines represent the contours of the true target (left) and of the GP surrogate model built during the search stage (right). The solid black diamonds indicate new points chosen by the poll method (here for simplicity a simple orthogonal poll), the grey circles represent the previously sampled points, and the orange solid star represents the global minimum of the function. When switching to the search stage, the blue diamond describes the point selected by the active sampling method based on the lower confidence bound obtained from the GP surrogate model (green region).

65 This alternation between the two stages makes BADS uniquely robust and effective. The
66 poll stage follows a slow but steady "model-free" optimization with theoretical guarantees.
67 Conversely, the search stage exploits a powerful "model-based" GP surrogate to propose
68 potentially large steps, which can be extremely effective if the surrogate is able to approximate

69 the target well. Notably, this strategy is fail-safe in that if the GP fails to locally model the
70 target, the search will fail and PyBADS will fall back to the safer poll method. The points
71 acquired during the poll will afford a construction of a better surrogate at the next search,
72 and so on. In addition, when the target is noisy, BADS follows some effective heuristics for
73 calibrating the surrogate model, checking the reliability of the predictions, and reassessing the
74 estimated value of the current point in light of the new points.

75 Thanks to these techniques, our algorithm has demonstrated high robustness and effectiveness
76 in solving optimization problems with noisy and complex non-convex objective functions.

Related work

78 Similarly to PyBADS, relevant libraries have been developed over the years in the area of
79 Bayesian Optimization, such as BoTorch (Balandat et al., 2020), GPflowOpt (Knudde et al.,
80 2017), Spearmint (Snoek et al., 2014), among others. Instead, NOMAD (Audet et al., 2022)
81 is the main reference library for *pattern search* algorithms, and it implements several variants
82 of MADS in C++, by providing Python and Julia interface bindings.

83 Differently to these algorithms, PyBADS comes with a unique hybrid, fast and robust combi-
84 nation of direct search (MADS) and Bayesian Optimization. This combination of strategies
85 protects against failures of the GP surrogate models – whereas vanilla BayesOpt does not have
86 such fail-safe mechanisms, and can be strongly affected by misspecification of the surrogate GP
87 model. PyBADS has also been designed to avoid problem-specific tuning, making it a generic
88 tool for model fitting. Compared to other approaches, PyBADS also has the advantage of
89 natively accommodating target functions with heteroskedastic (input-dependent) observation
90 noise. The results of our approach demonstrate that a hybrid Bayesian approach to optimization
91 can be beneficial beyond the domain of costly black-box functions. Finally, unlike most other
92 BayesOpt packages, targeted to an audience of machine learning researchers, PyBADS comes
93 with a neat API library and well-structured, user-friendly documentation.

94 PyBADS was developed in parallel to PyVBMC, a new software for sample-efficient Bayesian
95 inference (Acerbi, 2018, 2019, 2020; Huggins et al., 2023). PyBADS can be used in combination
96 with PyVBMC, by providing an effective way of initializing the inference algorithm at the
97 maximum-a-posteriori (MAP) solution.

Applications and usage

99 The BADS algorithm, in its MATLAB implementation, has already been applied in multiple
100 fields, especially in neuroscience where it finds a broad audience by efficiently solving difficult
101 model-fitting problems (Cao et al., 2019; Daube et al., 2019; J.-A. Li et al., 2020; Tajima et al.,
102 2019). Other fields in which BADS has been successfully applied include control engineering
103 (Stenger & Abel, 2022), electrical engineering (M. Li et al., 2022), material engineering (Ren
104 et al., 2021), robotics (Ren et al., 2020), petroleum science (Feng et al., 2022), environmental
105 economics (Nobel et al., 2020), and cognitive science (Stengård et al., 2022; van Opheusden
106 et al., 2023). Moreover, BADS has been shown to perform best in most settings of a black-box
107 optimization benchmark for control engineering (Stenger & Abel, 2022), highlighting the
108 effectiveness of our algorithm compared to other BayesOpt and direct-search approaches. With
109 PyBADS, we bring the same sample-efficient and robust optimization to the wider open-source
110 Python community, while improving the interface, test coverage, and documentation.

111 The package is available on both PyPI (`pip install pybads`) and `conda-forge`, and provides
112 an idiomatic and accessible interface, only depending on standard, widely available scientific
113 Python packages (Harris et al., 2020). The user only needs to give a few basic details about the
114 objective function and its parameter space, and PyBADS handles the rest of the optimization
115 task. PyBADS includes automatic handling of bounded variables, robust termination conditions,
116 sensible default settings, and does not need tunable parameters. At the same time, experienced
117 users can easily supply their own options. We have extensively tested the algorithm and

implementation details for correctness and performance. We provide detailed tutorials, so that
PyBADS may be accessible to those not already familiar with black-box optimization, and our
comprehensive documentation will aid not only new users but future contributors as well.

Acknowledgments

References

Abramson, M. A., Audet, C., & Digabel, S. L. (2009). OrthoMADS: A deterministic MADS
instance with orthogonal directions. *SIAM Journal on Optimization*, *20*(2), 948–966.

Acerbi, L. (2018). Variational Bayesian Monte Carlo. *Advances in Neural Information Processing
Systems*, *31*, 8222–8232.

Acerbi, L. (2019). An exploration of acquisition and mean functions in Variational Bayesian
Monte Carlo. *PMLR*, *96*, 1–10.

Acerbi, L. (2020). Variational Bayesian Monte Carlo with noisy likelihoods. *Advances in Neural
Information Processing Systems*, *33*, 8211–8222.

Acerbi, L., & Ma, W. J. (2017). Practical Bayesian optimization for model fitting with Bayesian
adaptive direct search. *Advances in Neural Information Processing Systems*, *30*, 1834–1844.

Agnihotri, A., & Batra, N. (2020). Exploring bayesian optimization. *Distill*. https://doi.org/
10.23915/distill.00026

Audet, C., & Dennis, J. (2006). Mesh adaptive direct search algorithms for constrained
optimization. *SIAM Journal on Optimization*, *17*, 188–217.

Audet, C., Dzahini, K. J., Kokkolaras, M., & Le Digabel, S. (2021). Stochastic mesh adaptive
direct search for blackbox optimization using probabilistic estimates. *Computational
Optimization and Applications*, *79*(1), 1–34.

Audet, C., Le Digabel, S., Montplaisir, V. R., & Tribes, C. (2022). Algorithm 1027: NOMAD
version 4: Nonlinear optimization with the MADS algorithm. *ACM Transactions on
Mathematical Software*, *48*(3). https://doi.org/10.1145/3544489

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E.
(2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. *Advances
in Neural Information Processing Systems 33*. http://arxiv.org/abs/1910.06403

Cao, Y., Summerfield, C., Park, H., Giordano, B. L., & Kayser, C. (2019). Causal inference in
the multisensory brain. *Neuron*, *102*(5), 1076–1087.

Daube, C., Ince, R. A. A., & Gross, J. (2019). Simple acoustic features can explain phoneme-
based predictions of cortical responses to speech. *Current Biology*, *29*(12), 1924–1937.e9.

Deng, G., & Ferris, M. C. (2006). Adaptation of the UOBYQA algorithm for noisy functions.
Proceedings of the 2006 Winter Simulation Conference, 312–319.

Feng, Q.-H., Li, S.-S., Zhang, X.-M., Gao, X.-F., & Ni, J.-H. (2022). Well production
optimization using streamline features-based objective function and bayesian adaptive direct
search algorithm. *Petroleum Science*, *19*(6), 2879–2894.

Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press.

159 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,
160 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M.
161 H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E.
162 (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

163 Huggins, B., Li, C., Tobaben, M., Aarnos, M. J., & Acerbi, L. (2023). PyVBMC: Efficient
164 bayesian inference in python. *arXiv*. https://doi.org/10.48550/ARXIV.2303.09519

165 Knudde, N., van der Herten, J., Dhaene, T., & Couckuyt, I. (2017). GPflowOpt: A Bayesian
166 Optimization Library using TensorFlow. *arXiv Preprint – arXiv:1711.03845*. https://arxiv.
167 org/abs/1711.03845

168 Li, J.-A., Dong, D., Wei, Z., Liu, Y., Pan, Y., Nori, F., & Zhang, X. (2020). Quantum
169 reinforcement learning during human decision-making. *Nature Human Behaviour*, *4*(3),
170 294–307.

171 Li, M., Yu, P., Wang, Y., Sun, Z., & Chen, Z. (2022). Topology comparison and sensitivity
172 analysis of fuel cell hybrid systems for electric vehicles. *IEEE Transactions on Transportation*
173 *Electrification*.

174 Nobel, A., Lizin, S., Witters, N., Rineau, F., & Malina, R. (2020). The impact of wildfires on
175 the recreational value of heathland: A discrete factor approach with adjustment for on-site
176 sampling. *Journal of Environmental Economics and Management*, *101*, 102317.

177 Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT
178 Press. ISBN: 978-0-262-18253-9

179 Ren, X., Chai, Z., Xu, J., Zhang, X., He, Y., Chen, H., & Chen, X. (2020). A new method to
180 achieve dynamic heat input monitoring in robotic belt grinding of inconel 718. *Journal of*
181 *Manufacturing Processes*, *57*, 575–588.

182 Ren, X., Huang, X., Feng, H., Chai, Z., He, Y., Chen, H., & Chen, X. (2021). A novel energy
183 partition model for belt grinding of inconel 718. *Journal of Manufacturing Processes*, *64*,
184 1296–1306.

185 Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: A review of algorithms
186 and comparison of software implementations. *Journal of Global Optimization*, *56*(3),
187 1247–1293.

188 Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & Freitas, N. de. (2016). Taking the
189 human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*,
190 *104*(1), 148–175.

191 Snoek, J., Swersky, K., Zemel, R., & Adams, R. (2014). Input warping for bayesian optimization
192 of non-stationary functions. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st*
193 *international conference on machine learning* (Vol. 32, pp. 1674–1682). PMLR.

194 Stengård, E., Juslin, P., Hahn, U., & van den Berg, R. (2022). On the generality and cognitive
195 basis of base-rate neglect. *Cognition*, *226*, 105160.

196 Stenger, D., & Abel, D. (2022). *Benchmark of bayesian optimization and metaheuristics for*
197 *control engineering tuning problems with crash constraints*. https://arxiv.org/abs/2211.
198 02571

199 Tajima, S., Drugowitsch, J., Patel, N., & Pouget, A. (2019). Optimal policy for multi-alternative
200 decisions. *Nature Neuroscience*, *22*(9), 1503–1511.

201 van Opheusden, B., Kuperwajs, I., Galbiati, G., Bnaya, Z., Li, Y., & Ma, W. J. (2023).
202 Expertise increases planning depth in human gameplay. *Nature*.