UNIVERSITAT AUTÒNOMA DE BARCELONA

DOCTORAL THESIS

Calibration in Cost-Effectiveness modeling (research diary)

Author:
David GÓMEZ

Supervisors:
Dr. Josep Lluis ARCOS
Dr. Mireia DÍAZ

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Artificial Intelligence Research Institute (IIIA)
Escola d'Enginyeria de la UAB

August 8, 2022

Contents

Chapter 1

Background

1.1 Cost-Effectiveness Models in Healthcare

1.1.1 Objective

Compare different strategies for detection/treatment of a disease, from health and economic point of view.

1.1.2 Methodology

Simulation model to mimic the strategies and compare the outputs for each strategy to determine which strategies are worth considering. A special "strategy" called the natural history describes the progression of the disease without any planned interventions, and it is used to calibrate some the inputs that will be used in the rest of strategies (see section 3).

1.1.3 Types of model

Decision trees, markov model, microsimulation, ... depending on the needs of the domain and the degree of detail and granularity required.

1.1.4 Inputs

Parameters extracted from the scientific literature, studies, expert opinions, assumptions, ... An interesting intermediate input are the transition matrices that show the probabilities of transitioning between health states.

1.1.5 Outputs

For each strategy:

- Effectiveness measure (e.g. Quality-Adjusted Life Years, QALYs)
- Cost measure (e.g. euros, €)
- Other general measures of interest: incidence, mortality, ...
- Other domain-dependent measures: e.g. number of hysterectomies, number of high-grade lesions, ...

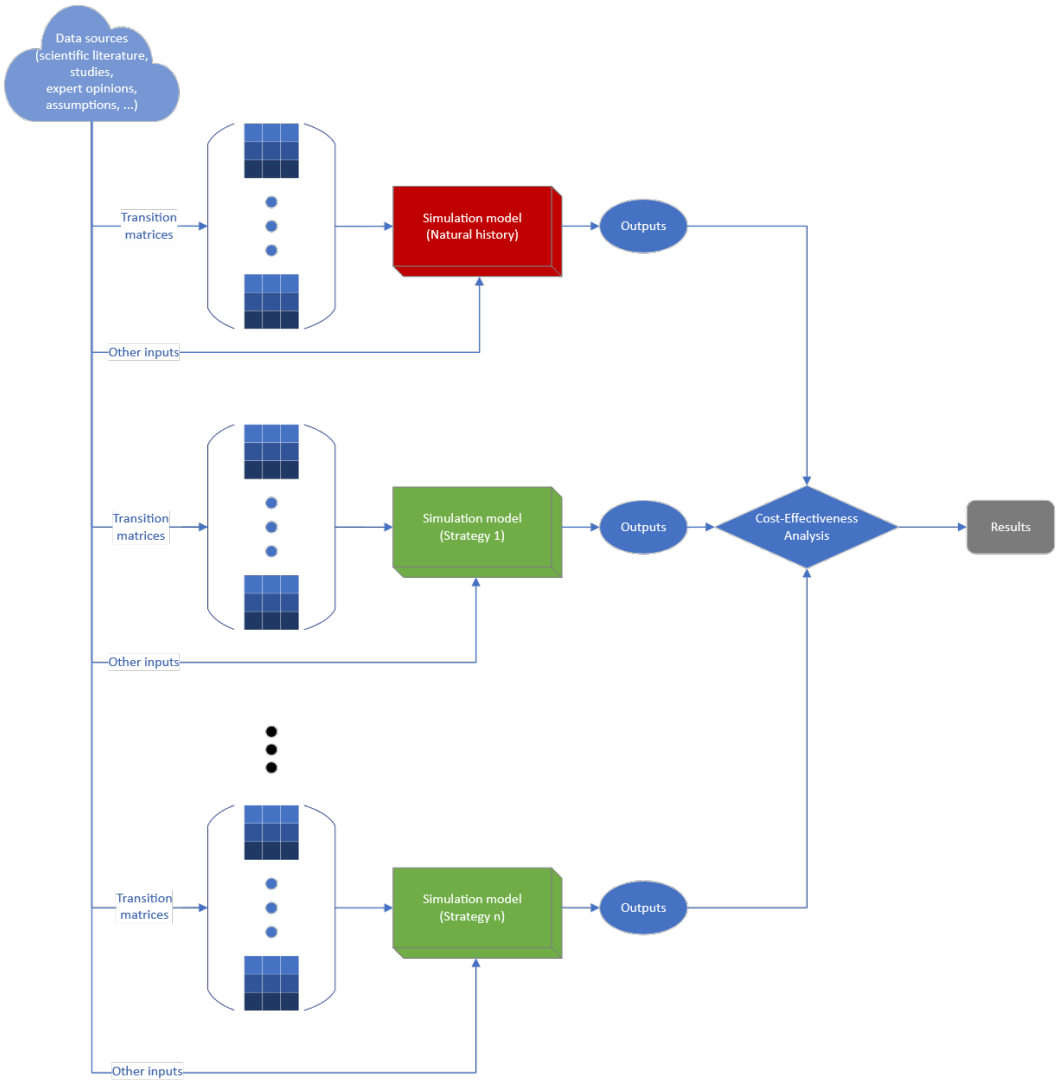

FIGURE 1.1: Overview of the cost-effectiveness analysis.

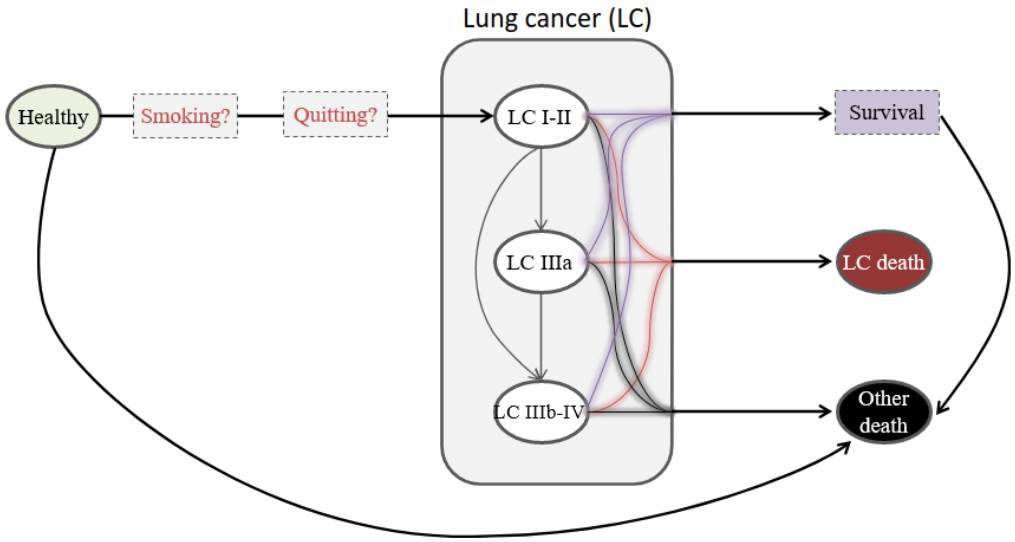

FIGURE 1.2: Markov state diagram of the lung cancer model.

Chapter 2

Cost-Effectiveness Analysis Methodology

2.1 Calibration

Before starting the base analysis, we calibrate the transition matrices in the natural history by slightly modifying the original probabilities so that the output of our model (e.g. incidence, mortality, ...) fits an observed value based on evidence. These calibrated probabilities can then be used by the rest of the strategies in the base analysis. See Calibration Workflow for more details.

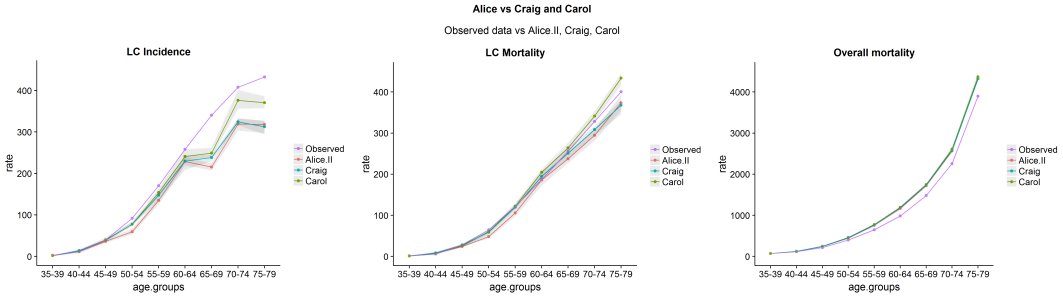

FIGURE 2.1: Example of three calibration curves for incidence, lung cancer mortality and mortality from other causes, along with the expected outcome.

2.2 Base analysis

Each strategy is plotted in the Cost and Effectiveness axes and the efficiency curve shows the strategies that are cost-effective, the rest are dominated by them and they are not considered cost-effective.

We can compare each strategy in relation to another calculating the Incremental Cost-Effectiveness Ratio as:

$$ICER = \frac{\Delta C}{\Delta E} = \frac{C_2 - C_1}{E_2 - E_1}$$

If the ICER is below the Willingness-To-Pay (WTP) threshold (i.e. the maximum amount of money a country/region is willing to pay per additional QALY) the second strategy is more cost-effective than the first. If the ICER is greater than the WTP the strategy's benefits are not considered cost-effective (i.e. the increased

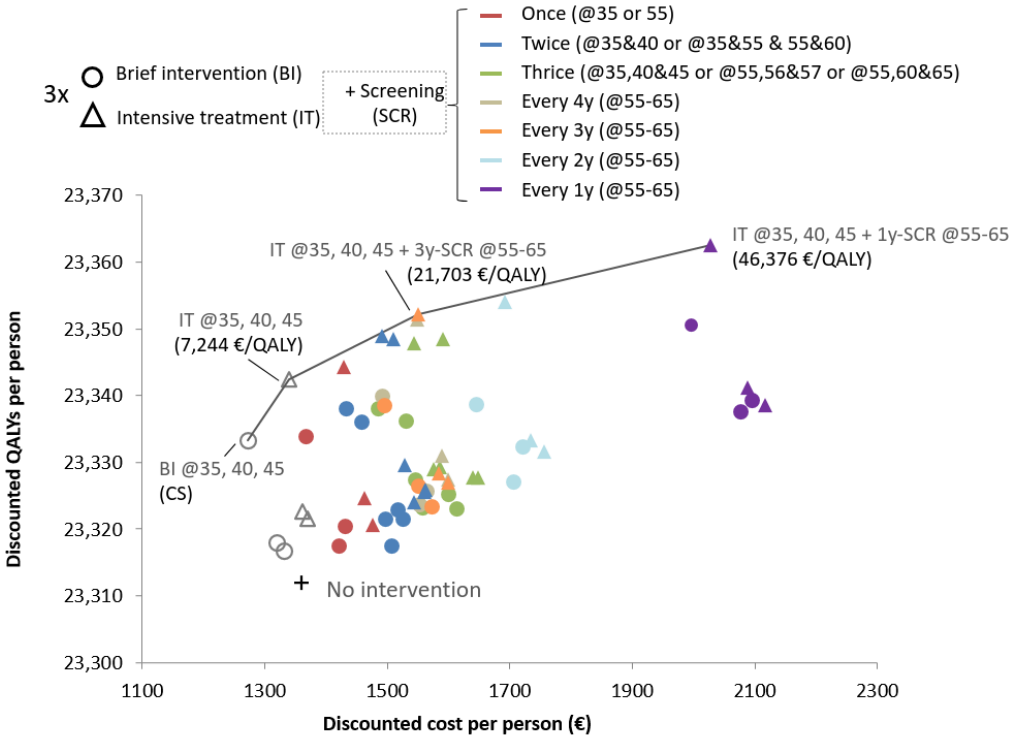

FIGURE 2.2: Example of an efficiency curve showing the cost (€) and effectiveness (QALYs) of each of the simulated strategies. The strategies on the curve represent the cost-effective strategies, those below it are not usually considered since they are always dominated by one of those in the curve.

health benefit does not justify the increment of cost). Negative ICERs imply that one strategy dominates the other one.

2.3 Sensitivity analysis

Once the base analysis is performed we evaluate the uncertainty of the used parameters to check the robustness of the results. We modify the values of the parameters of interest to see how they affect the output of the model.

2.3.1 Deterministic Sensitivity Analysis (DSA)

A sweep is performed over a range (e.g. +-15% of the base value) for the parameters of interest, to see how the ICER changes and whether the cost-effectiveness decision is different.

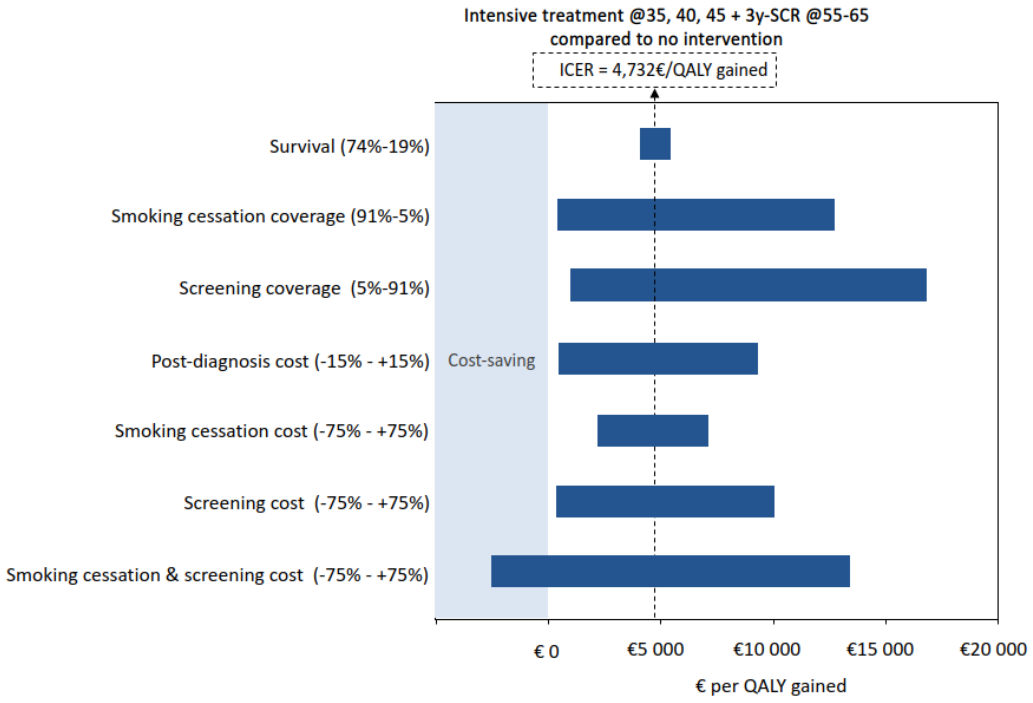

FIGURE 2.3: Example of a tornado diagram showing the impact on the ICER between two particular strategies when changing one single parameter over a predefined range.

2.3.2 Probabilistic Sensitivity Analysis (PSA)

Each parameter of interest is modeled as a probabilistic distribution (e.g. Beta for probabilities, Gamma/Lognormal for costs, ...) with the base value as the mean and a standard deviation dependent on the amount of uncertainty. We sample from these distributions (univariate or multivariate) to run a number of random simulations to check the percentage of simulations that show a cost-effective result (i.e. the percentage of simulations below the line ICER=WTP, see figure 2.4).

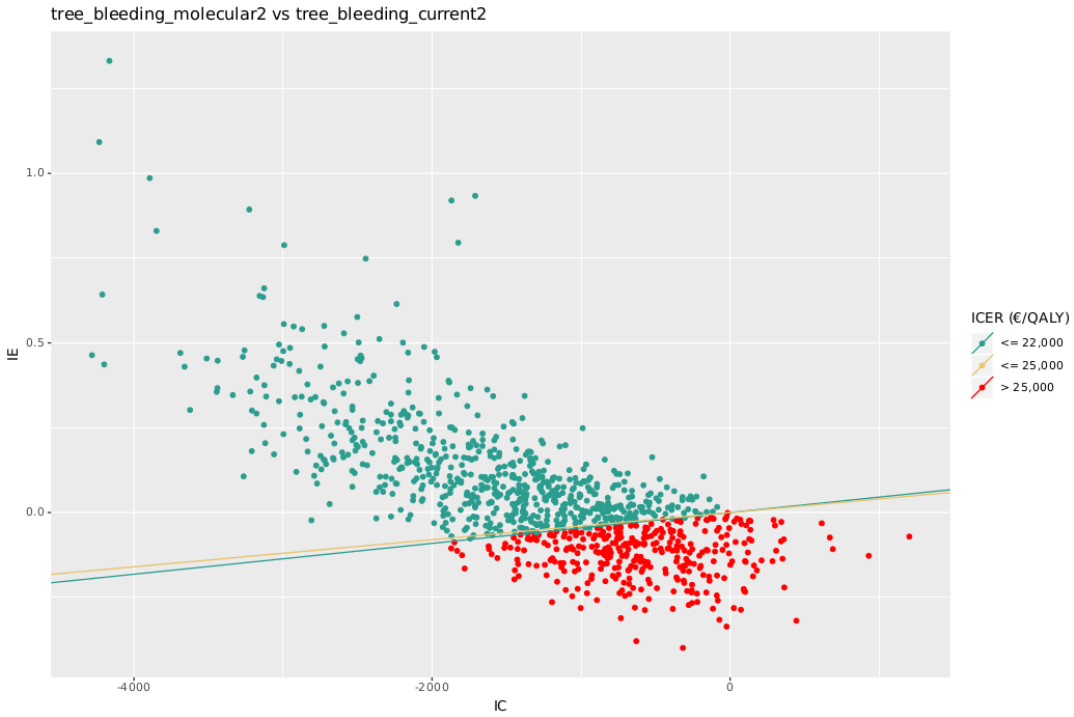

FIGURE 2.4: Example of a scatterplot of a PSA, showing results from 1,000 random iterations and how they relate to the cost-effectiveness threshold (WTP). Green and red points represent cost-effective and non-cost-effective simulations respectively.

Chapter 3

Calibration workflow

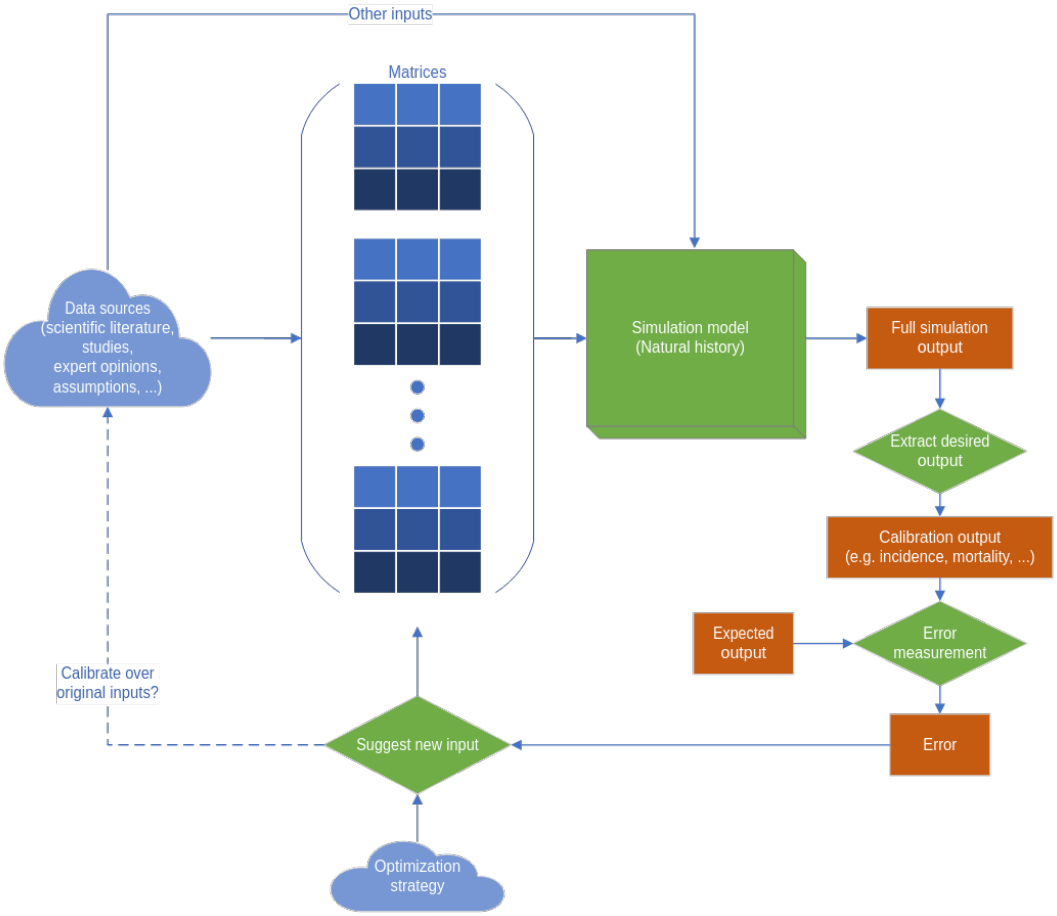

FIGURE 3.1: Conceptual summary of the calibration procedure.

Chapter 4

Research proposal

Cost-effectiveness models are used to evaluate and compare different medical strategies (e.g. strategies to detect cancer), in terms of both health and cost, and to help decision makers determine the optimal allocation of medical resources. These models have several inputs that describe the environment, the disease and the strategies used (inputs like probabilities, sensitivities and specificities for medical procedures, costs, utility values, ...) and can have several outputs as needed for the analysis (e.g. average life expectancy, average cost, incidence of the disease, mortality, ...).

The stated goal of the research project is to find efficient ways to calibrate these cost-effectiveness models by changing some input parameters so a particular output (usually incidence or mortality) matches a given value found in the scientific literature. In summary, **we can frame the problem as the optimization of a black box function computing (for example) the euclidean distance between the simulated output and a theoretical value**.

In some of our models we find that classical optimization methods need many simulations to converge to a good solution, demanding a lot of time and computing resources. Also, in many cases we are able to provide information about the statistical distribution of some of the inputs that could help in the optimization process. For these reasons I believe Bayesian Optimization (BO) might be a good alternative to adapt to our particular modeling needs.

Some preliminary (ongoing) experiments, using both bayesian and classical methods, are available in a jupyter notebook in `https://github.com/david-gomez-guillen/phd`, using the hypermapper [Nardi et al., 2019] python library for BO and scipy for classical optimization methods. This first set of tests focus on the performance of different optimization methods on well-known analytical functions that are easy to compute. Once acquainted with BO usage, the next step would be to repeat the tests on the actual cost-effectiveness models.

The conclusions and challenges found in the tests so far include:

1. BO using Gaussian Processes might be too slow for some of our more lightweight simulation models, classical methods converge faster even if they need more function evaluations. It should not be a problem for more computationally expensive simulation models.

2. BO tests performed converge to worse optima compared to classical optimization methods (e.g. Nelder-Mead, BFGS). It might be due to code issues: either implementation problems or an incorrect usage of the library.

3. Gaussian processes regression used in BO becomes more expensive for each iteration due to having to calculate the inverse of a matrix that grows with the number of observations. Regression becomes very slow to compute after a number of evaluations, especially for high-dimensional inputs (depending

on the available hardware) [Das, Roy, and Sambasivan, 2015][Terry and Choe, 2021].

4. Regular bayesian methodology (updating iteratively the model after each observation) is more difficult to parallelize than classical methods, though some alternatives exist [Daulton, Balandat, and Bakshy, 2021][Wang et al., 2016].

5. BO in these tests performed with priors for the optimum [Souza et al., 2021] (implemented in hypermapper: `https://github.com/luinardi/hypermapper/wiki/prior-injection`) do not seem to converge faster. It might be due to library issues too: optimum priors implementation is labeled as experimental in the documentation.

6. When optimizing the actual cost-effectiveness models in future tests it would be interesting to add constraints for the inputs [Gardner, Kusner, and Jake, n.d.][Ungredda and Branke, 2021]. E.g: one input parameter being greater than another.

Chapter 5

Thoughts & ideas

5.1 Sequential Model-Based Optimization (SMBO)

Current tests are performed using regular Bayesian Optimization (BO) via Gaussian Processes. Resources are available in `src/R` (interactive demo code), `notebooks` (jupyter notebooks for the same demos) and `output/gp` (output plots for different regression scenarios).

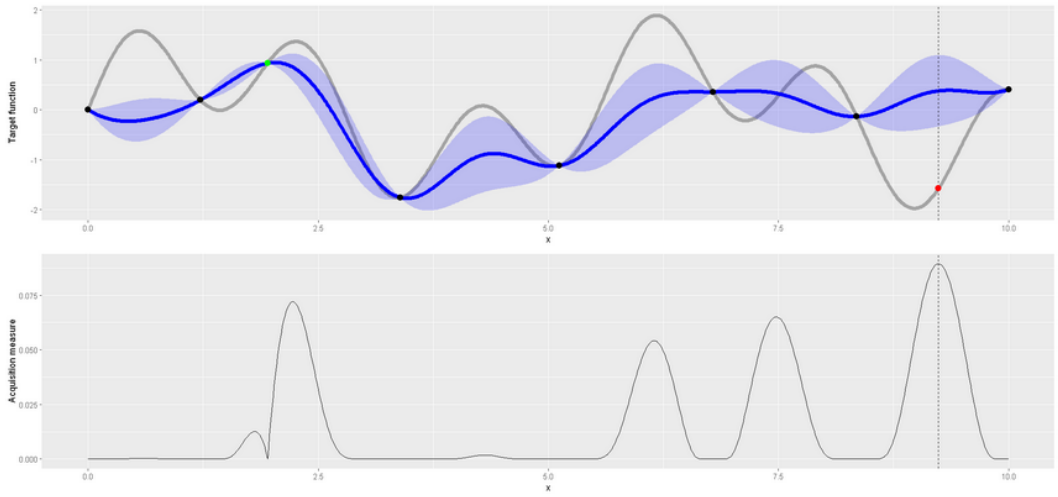

FIGURE 5.1: Bayesian Optimization using a Gaussian process after the eighth observed point. The acquisition function used is the Expected Improvement.

The error functions that we are trying to optimize, though, are not particularly time-consuming, so other more naive approaches work better by producing good results in a much shorter time (though many more evaluations of the error function). In order to improve this method for these use cases we can focus on different areas.

5.1.1 Surrogate model

Bayesian Optimization uses a surrogate model to approximate the target function and determine the optimal candidates to evaluate for each iteration. The default choice in many cases is a Gaussian Process but other alternatives exist, such as random forests or Tree Parzen Estimators (TPE) [Rasmussen and Williams, 2006][Bergstra et al., 2011].

5.1.2 Kernel selection/composition

One crucial hyperparameter in the BO method is the kernel. This function determines how the uncertainty over the regression curve is determined and its properties can greatly influence the fit and inference performance of the surrogate model (figure 5.2). An important step in the optimization procedure will be the selection and/or composition of kernels for the optimization problem [Repický, Pitra, and Holen ˇa, n.d.][Duvenaud, n.d.][Duvenaud, Nickisch, and Rasmussen, 2011][Duvenaud et al., 2013].

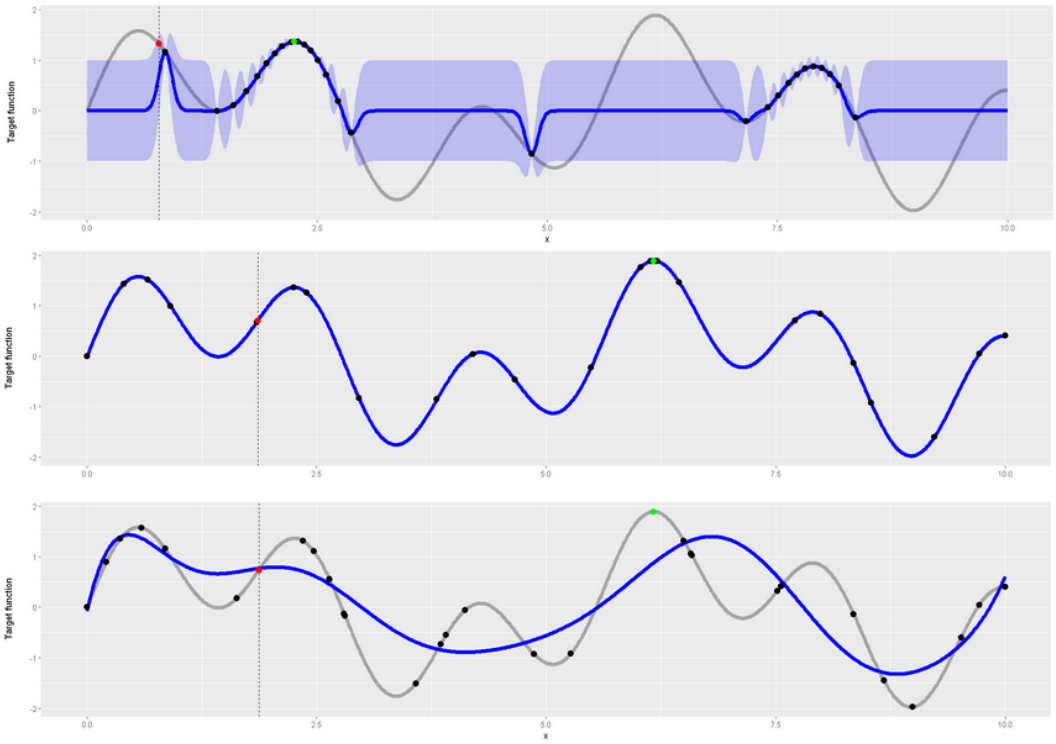

FIGURE 5.2: Three GP regressions with different kernels $K(x, x')$ over the same target function: $e^{-100(x-x')^2}$ (top), $e^{-(x-x')^2}$ (middle) and $e^{-0.1(x-x')^2}$ (bottom). In the top plot each point has small influence over the rest, sampling new points inefficiently and resembling other local optimization techniques. In the bottom plot each point has too much small influence over the rest making a good fit impossible if the data varies more than expected. In the middle plot we can see a good fit with an appropriate kernel for the target function.

- https://www.cs.toronto.edu/~duvenaud/cookbook/
- https://nipunbatra.github.io/blog/ml/2021/09/03/param-learning-sgd.html

5.1.3 Batch iterations

Each iteration we can select more than one candidate to evaluate. In expensive target functions this might be prohibitive, but in our case we might benefit for adding more information to the model. The batch size can be a modulation parameter depending on the computational cost of the function.

In the case of batches of more than one, we have to consider how those candidates are chosen: the maximum of the acquisition function is one, but the rest should be chosen such that the information provided is significant and not redundant given the rest (e.g. not too close to each other) [Nguyen et al., 2017][Gonzalez et al., n.d.].

5.1.4 Constraints

We suggest the use of the Constrained Expected Improvement acquisition function to incorporate constraints in the BO method [Gardner, Kusner, and Jake, n.d.]. An example can be seen in figure 5.3.

Other alternative methods exist to incorporate constraints as well [Paulson and Lu, 2021][Ungredda and Branke, 2021][Hernández-Lobato et al., 2016][Swiler et al., 2020][Lam and Willcox, n.d.].

One related subject to constraints is the possibility to place priors on the input values to guide the optimization process [Souza et al., 2021]. Priors do not impose hard restrictions, only a starting hypothesis on the optimal values that gets progressively overshadowed in the presence of more evidence (i.e. data) [Stark, 2015].

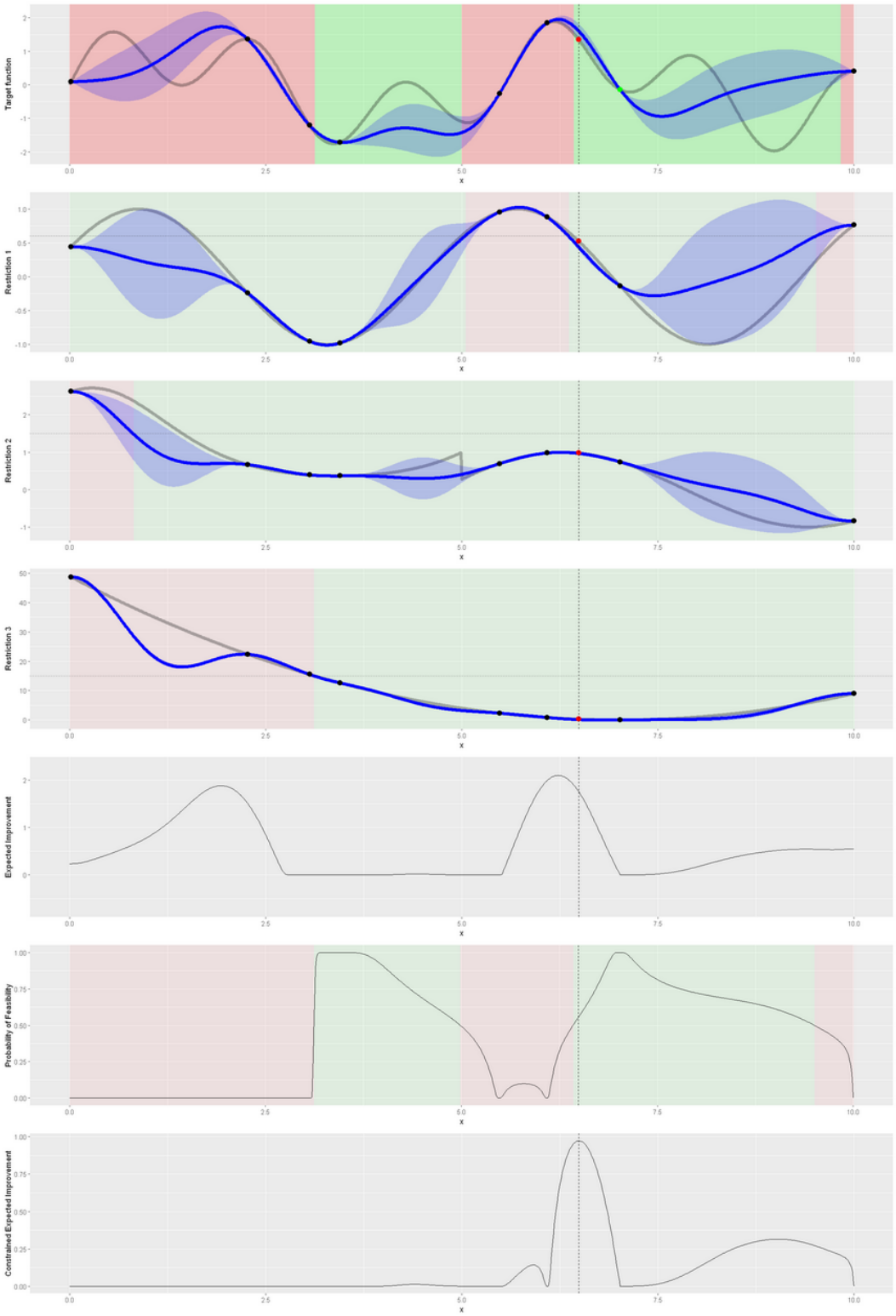

FIGURE 5.3: (Constrained) Bayesian Optimization using a Gaussian process after the eighth observed point. The acquisition function used is the Constrained Expected Improvement, combining the Expected Improvement with the probability of feasibility acquisition functions.

5.2 Calibration over original inputs

In the standard calibration procedure we change the transition probabilities in the matrices, assuming they are independent and enforcing constraints in the error calculation step. One alternative would be to calibrate over those relevant original inputs (i.e. those found in the literature, studies, expert opinions, ...) and then calculate the transition probabilities (and possibly other inputs of the model) from them. With this method we preserve and inject the knowledge we have about the domain into the model (that is, how to calculate the probabilities from the scientific sources, assumptions, implicit restrictions, ...).

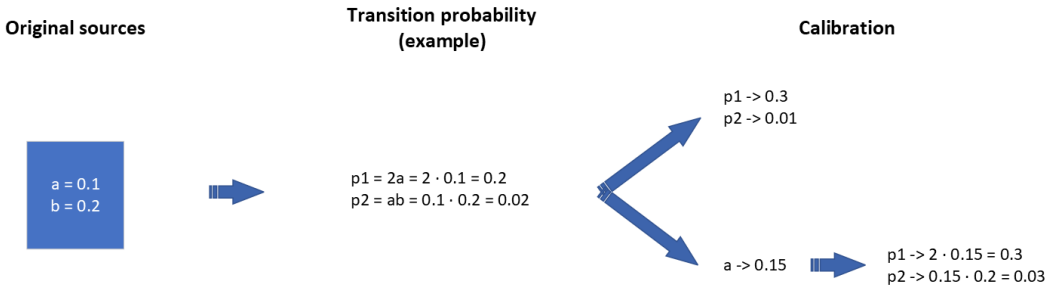

FIGURE 5.4: Calibration procedures: directly over the probabilities ("Calibration", top) and over the original inputs ("Calibration", bottom).

Since we calibrate to account for the uncertainty in the natural history and the probabilities in the matrices are calculated from parameters, we could classify the uncertainty sources in two: uncertainty associated to the inputs and uncertainty associated to the calculation of the probability.

5.2.1 Parameter uncertainty

If we are sure about the calculation of the transition probability, like a well-known relationship (e.g. Bayes formula), the remaining sources of uncertainty are the inputs themselves. In this case we can set up the optimizer to change these inputs, recalculate the probabilities and run the model.

For interpretation and sanity check purposes we can check both the changed input value and the newly-calculated transition probability.

5.2.2 Calculation uncertainty

Beyond the inputs, the calculation itself might be uncertain as well, for example by making a very rough approximation or assuming a probabilistic distribution with a poor fit. In this case we can insert an error term or scaling factor in the formula to account for the misspecification, with a neutral initial value (0 if error, 1 if factor). Then, the calibration could include these error/scaling terms in the set of parameters to be optimized.

For interpretation and sanity check purposes, besides the probabilities themselves as usual, we can check the error term/scaling factor. If they are too different from the initial values we might conclude that the calculation is not trustworthy and we might have to review our assumptions. Also, if the calculation is a very rough

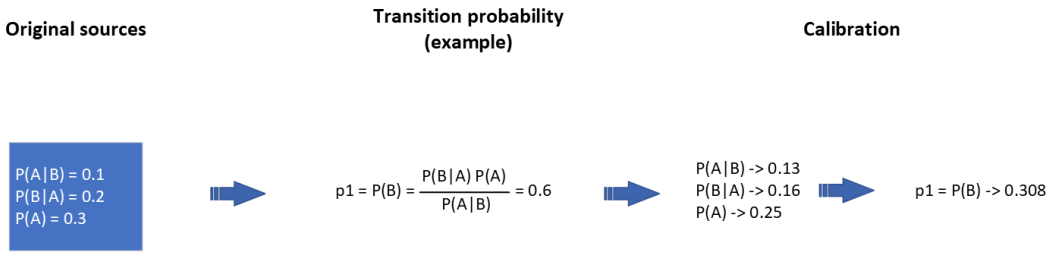

FIGURE 5.5: If we are certain of the relationship between the probability and the inputs, we can calibrate over the inputs and we will get the calibrated probability by preserving the link with the original inputs.

estimate another alternative would be to reject the calculation itself and optimize the probability value as usual.

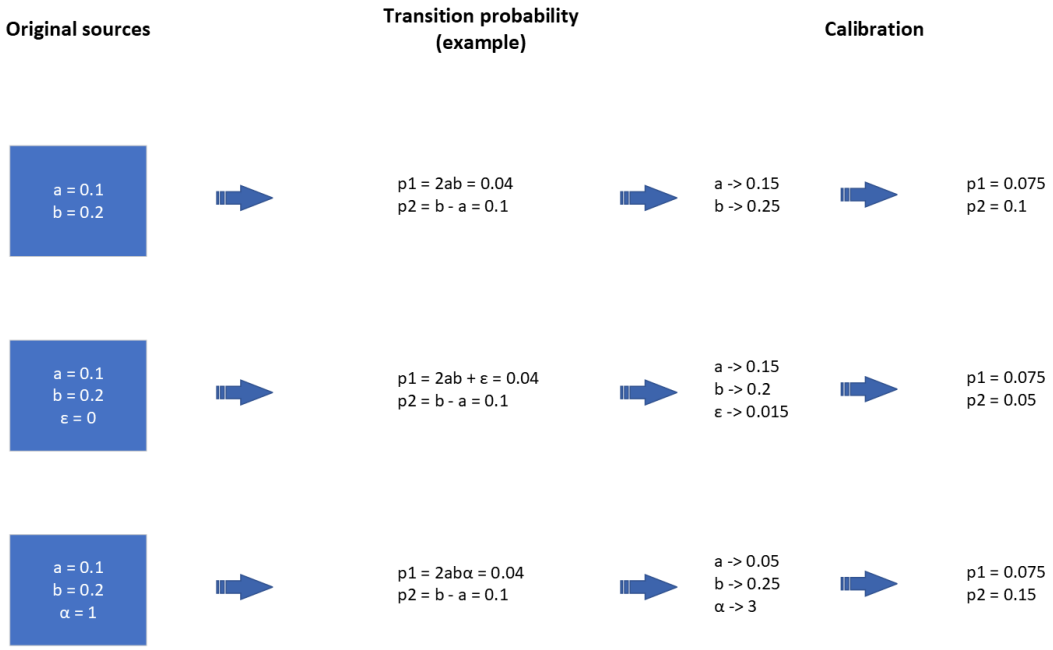

FIGURE 5.6: To account for the possibility of a misspecified calculation we can add additional terms to a formula. We can see calibration with no formula flexibility (top row), an additive error term for p1 (middle row) or a multiplicative factor for p1 (bottom row).

5.2.3 Comments

Pros

- Preserving the link and implicit knowledge between the original sources and the calculated probabilities
- Preserving relationships between probabilities and (some kinds of) constraints

Cons

- Overcomplicating the calibration procedure in simple cases
- Relationships might not be too complicated

5.3 Input/probabilities dependencies using graph theory

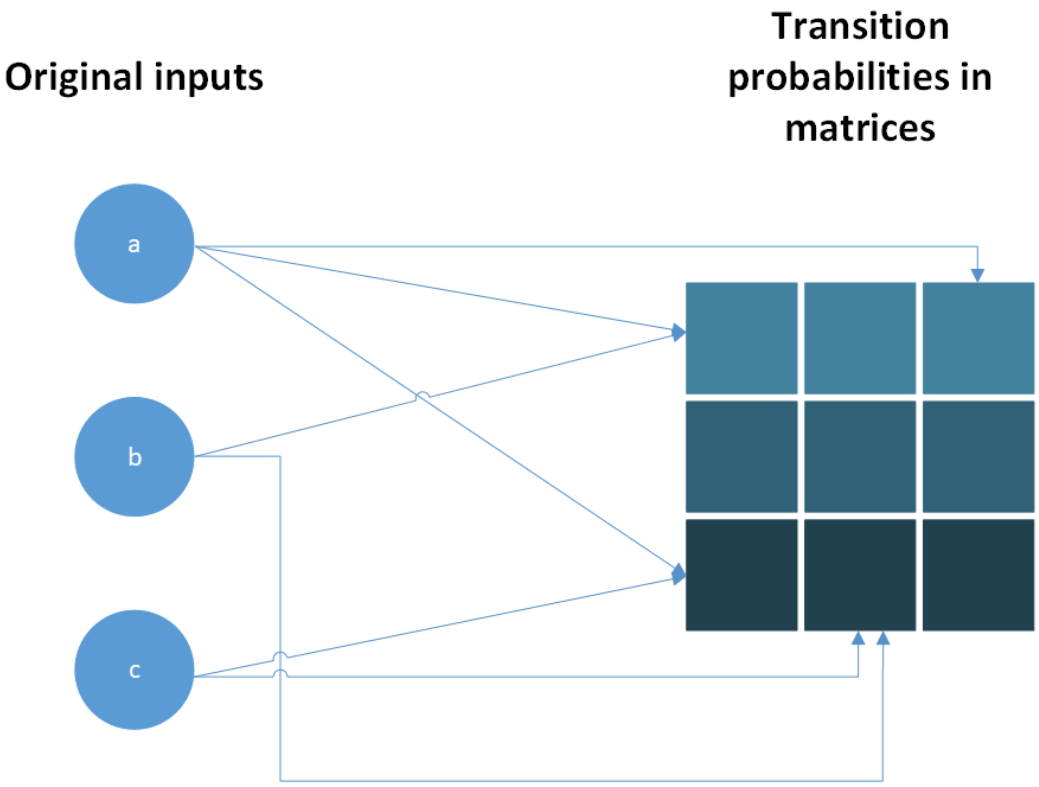

FIGURE 5.7: We could use graph theory to help optimization considering the dependencies between inputs and the transition probabilities in the matrices.

5.3.1 Comments

Pros

- Might help the calibration procedure by exploiting domain knowledge

Cons

- Overcomplicating the calibration procedure in simple cases
- Relationships might not be too complicated
- Too vague at this point

Chapter 6

Tests performed

6.1 Test model: lung cancer

- Total of 9 matrices (one per age group: 35-39, 40-44, . . . , 75-79).
- 7 health states: healthy, stages I-II, stage IIIa, stage IIIb, LC survival, death from LC, death from other causes → 7x7 matrices. Sometimes the LC survival state is excluded from the calibration resulting in 6x6 matrices.
- Matrices represent monthly steps in the simulation. Since they are applied for 5-year groups, each matrix is used in 5 * 12 = 60 iterations in the model.
- A simplified calibration can be performed without running the model, only the matrices are used. This is a fast approximation since we are not considering some factors of the full model (e.g. prevalence of smoking):
  - LC survival state is excluded → 6x6 matrices
  - From the 6x6=36 probabilities per matrix, only 11 probabilities are allowed to change. The rest are either constant (zeroes, ones) or one minus the sum of the rest of the row.
  - The error measurement is a weighted sum of the absolute differences of the LC incidence, LC mortality and mortality from other causes. The weights are 0.45, 0.45 and 0.10 respectively.

6.2 Simplified calibration

6.2.1 1 matrix

- Source file: models/lung/calibration_wrapper.R (N_MATRICES set to 1)
- Only the first age group is being calibrated (35-39): 1x11 = 11 parameters.

Algorithm	Initial matrix	Nelder-Mead	Particle swarm	Bayesian
Error	1.1545799674960	0.6633085653748	0.66298515	0.6629851533965
Time (s)	-	0.76	22.97	114.89
Evaluations	-	252	10100	21

6.2.2 2 matrices

- Source file: models/lung/calibration_wrapper.R (N_MATRICES set to 2)
- The first and second age groups are being calibrated (35-39 and 40-44): 2x11 = 22 parameters.

Algorithm	Initial matrix	Nelder-Mead	Particle swarm	Bayesian
Error	1.6495138536869	0.7333381543348456	0.72801101	0.7287116136105645
Time (s)	-	4.44	24.46	421.69
Evaluations	-	2092	10100	70

6.2.3 All (9) matrices

- Source file: models/lung/calibration_wrapper.R (N_MATRICES set to 9)
- All age groups are being calibrated: 9x11 = 99 parameters.
- Standard bayesian optimization takes too much time and the process was aborted before completion. Other strategies could be attempted: calibrate matrices sequentially, restrict number of parameters, optimize gaussian process regression (see section 4), ...

Algorithm	Initial matrix	Nelder-Mead	Particle swarm	Bayesian
Error	6.6842251473203	4.0210661197016	4.3594607	6.142333876
Time (s)	-	57.58	35.64	31352
Evaluations	-	19800	9407	50

6.2.4 Method comparison

6.2.5 Dimensionality comparison

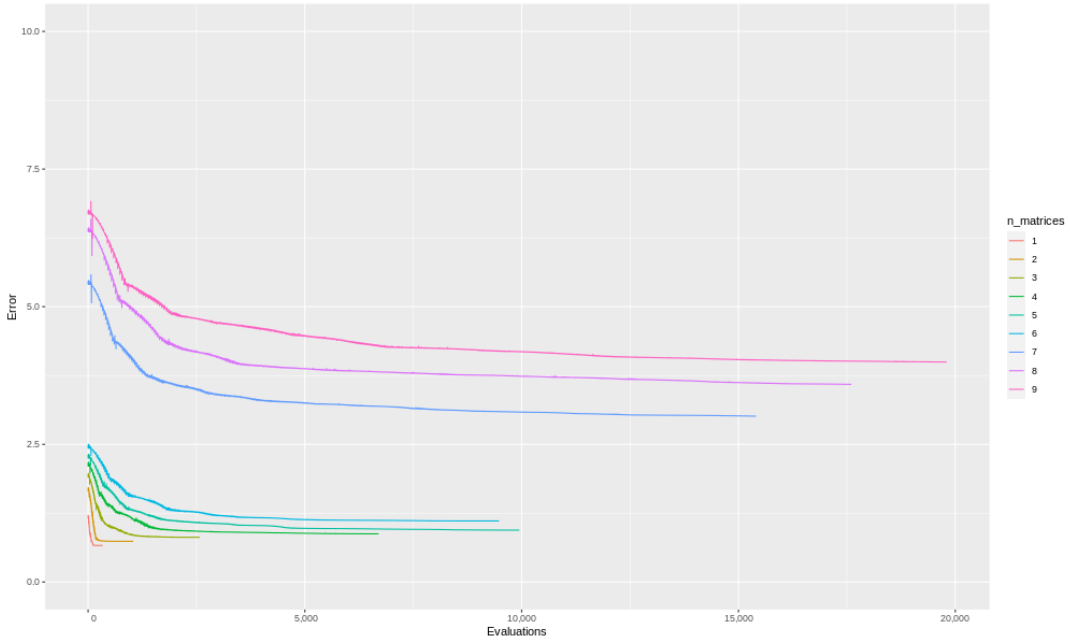

FIGURE 6.1: Nelder-Mead error by iteration step

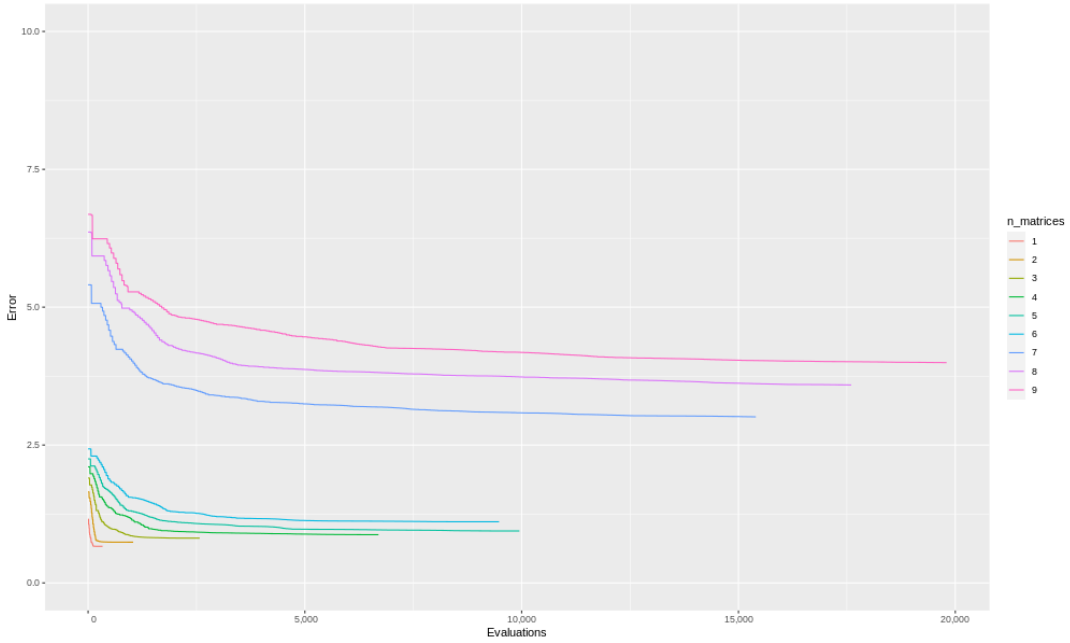

FIGURE 6.2: Nelder-Mead smoothed error by iteration step (cumulative minimum)

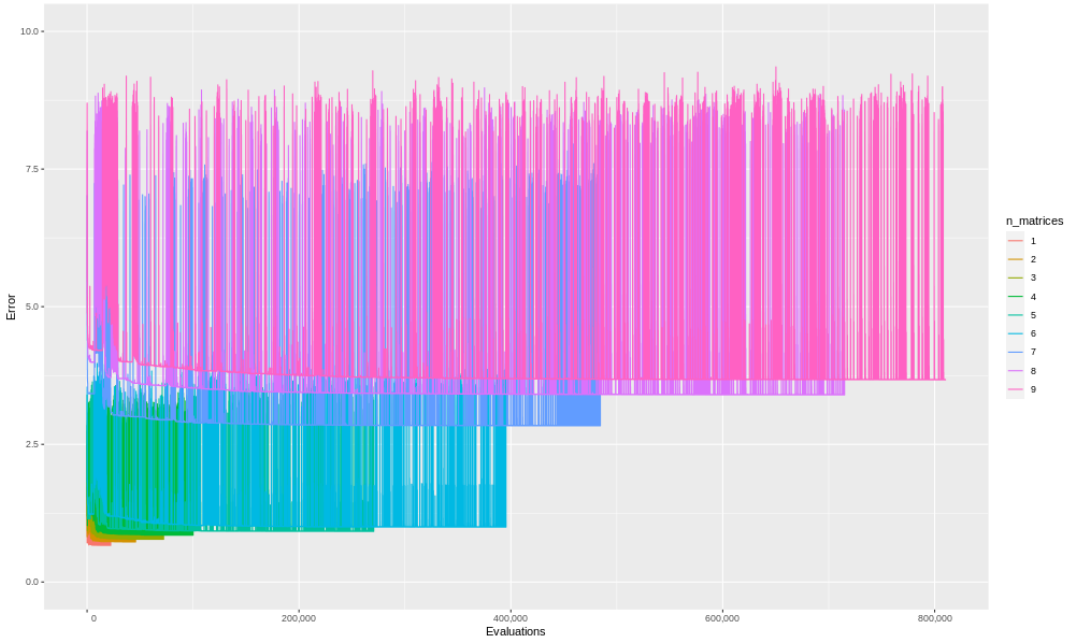

FIGURE 6.3: Simulated Annealing error by iteration step

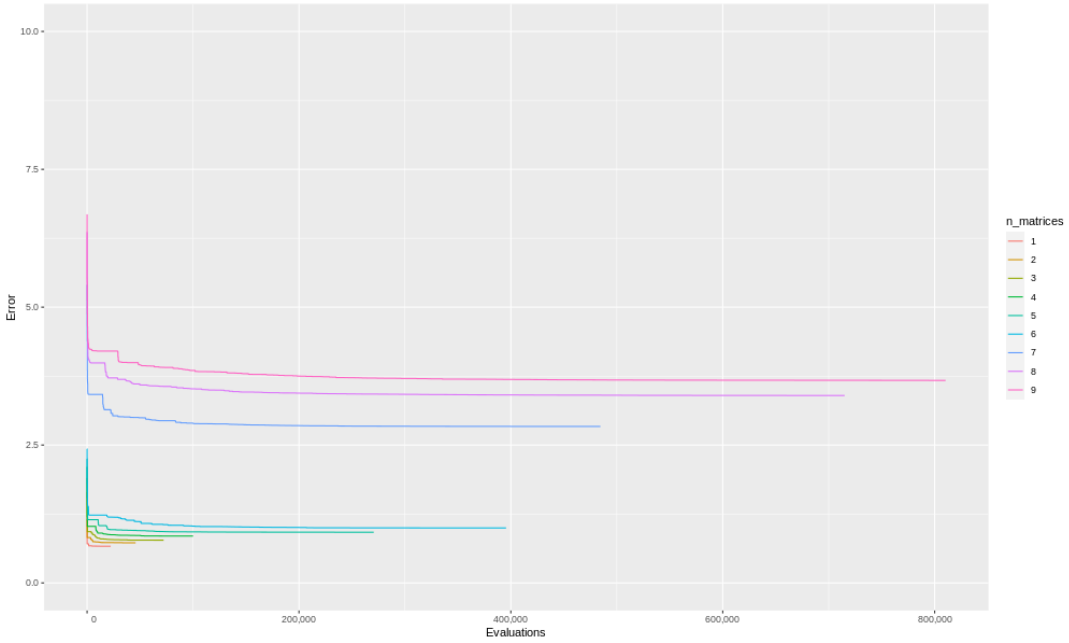

FIGURE 6.4: Simulated Annealing smoothed error by iteration step (cumulative minimum)

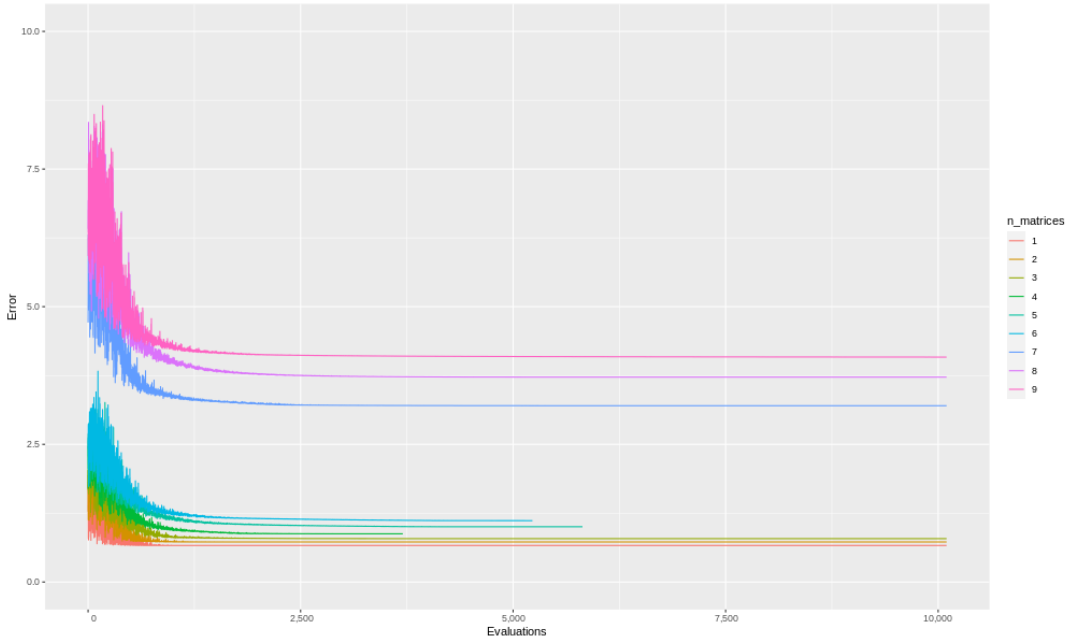

FIGURE 6.5: Particle Swarm Optimization error by iteration step

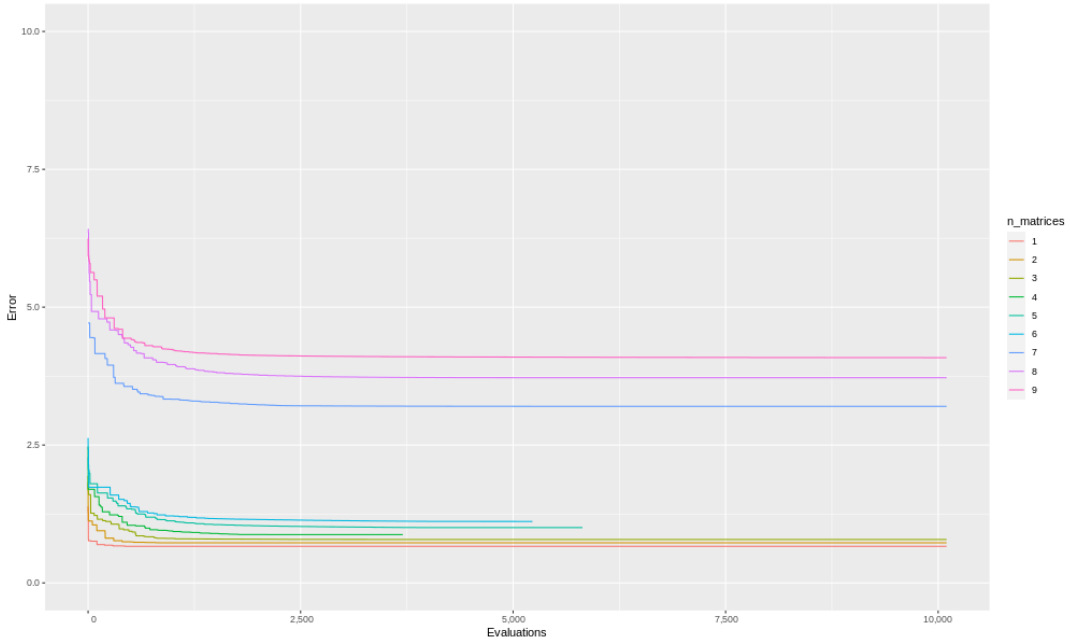

FIGURE 6.6: Particle Swarm Optimization smoothed error by iteration step (cumulative minimum)

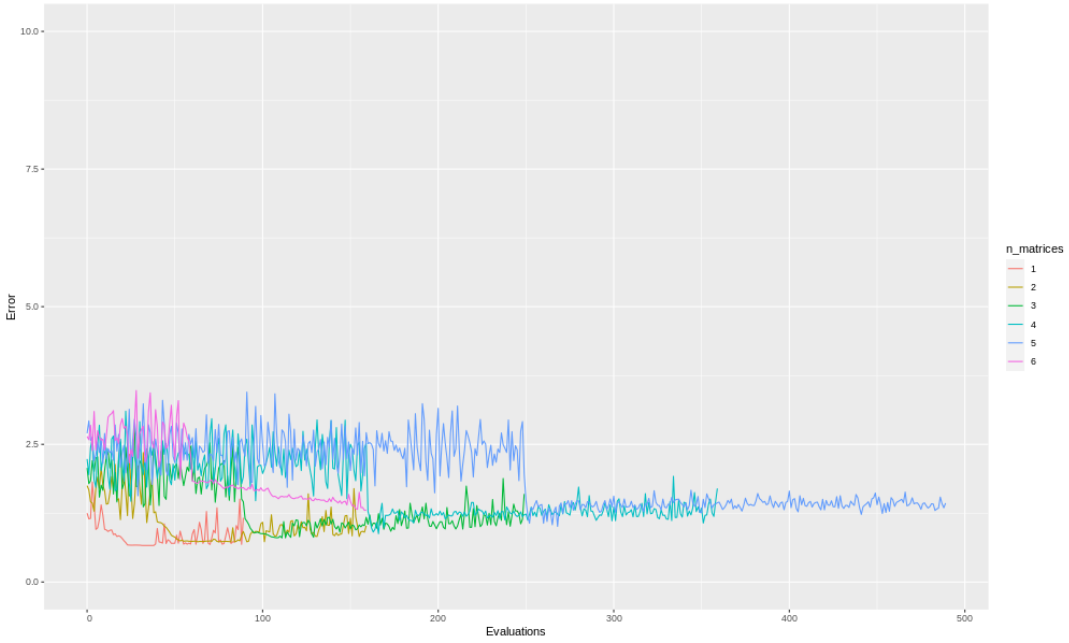

FIGURE 6.7: Bayesian Optimization error by iteration step

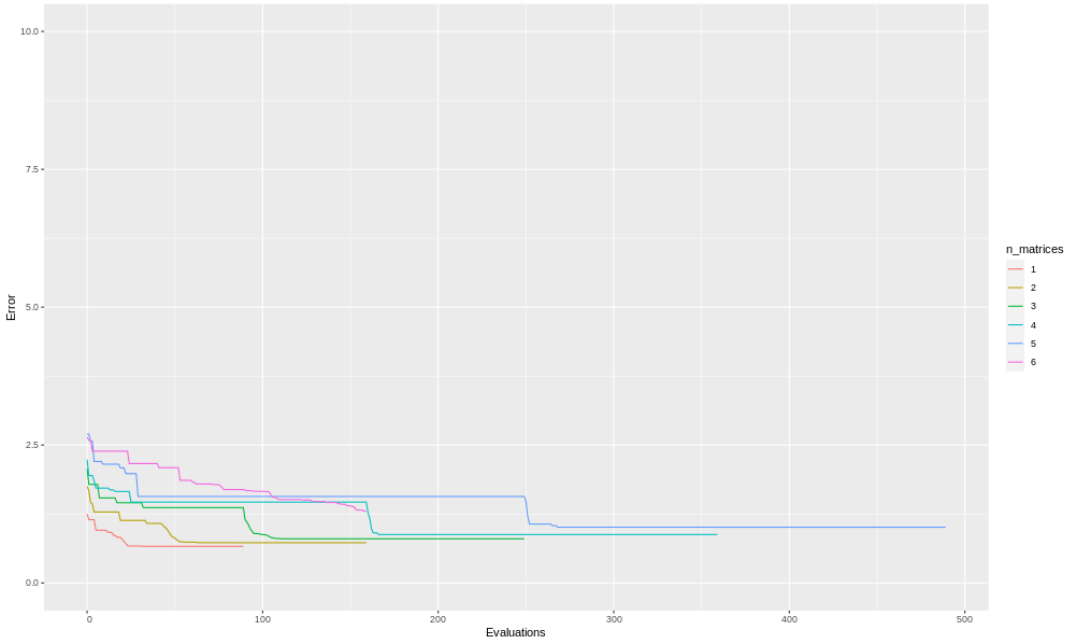

FIGURE 6.8: Bayesian Optimization smoothed error by iteration step (cumulative minimum)

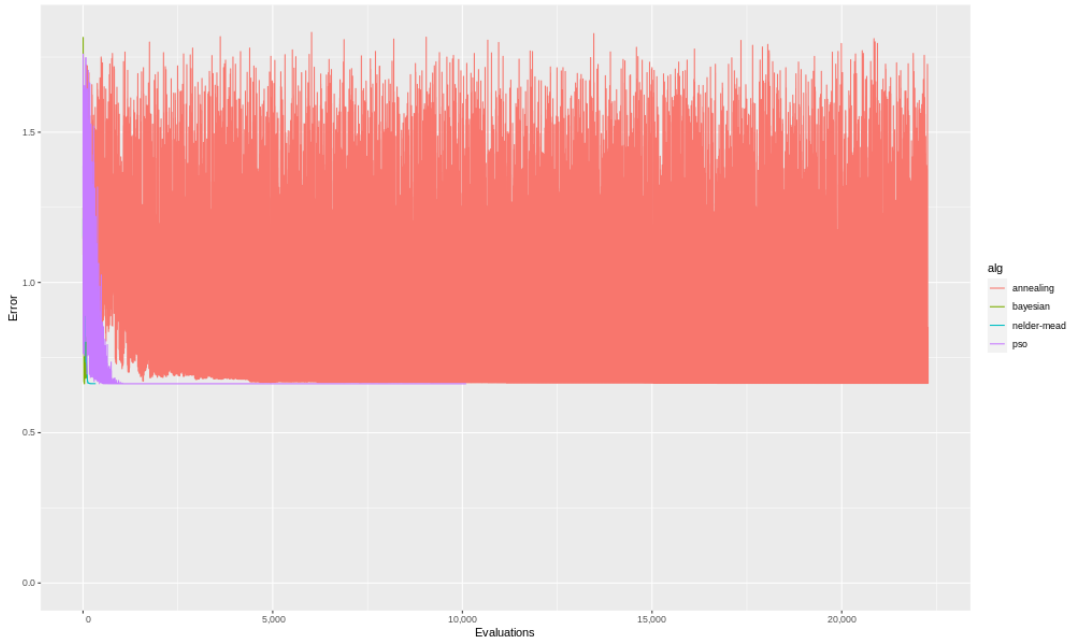

FIGURE 6.9: Errors on n_matrices = 1 by method

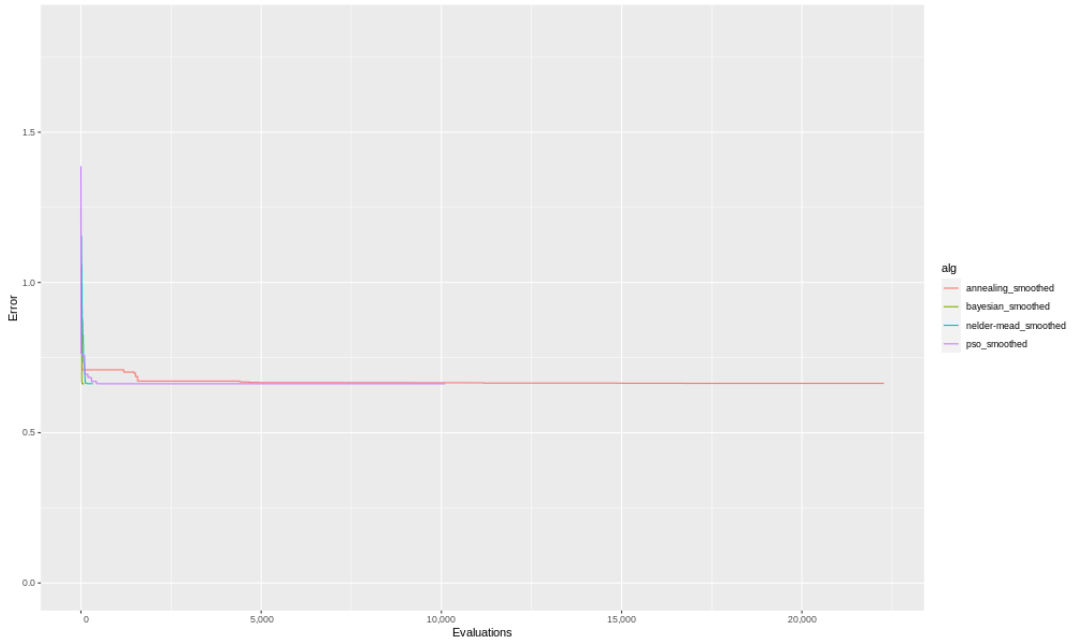

FIGURE 6.10: (Smoothed) errors on n_matrices = 1 by method

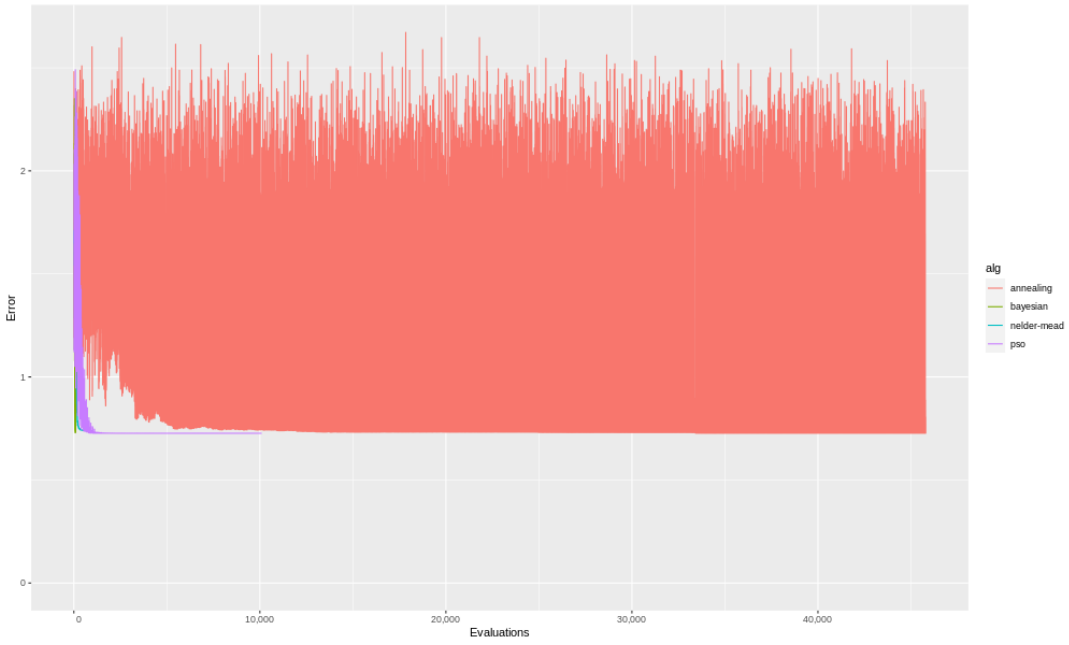

FIGURE 6.11: Errors on n_matrices = 2 by method

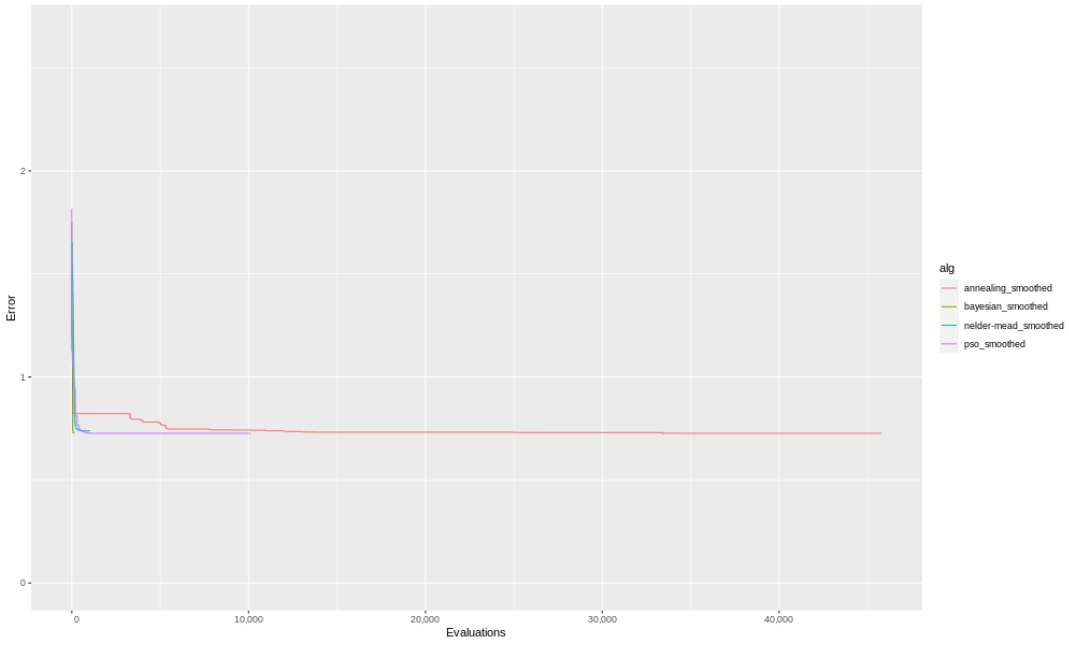

FIGURE 6.12: (Smoothed) errors on n_matrices = 2 by method

FIGURE 6.13: Errors on n_matrices = 3 by method

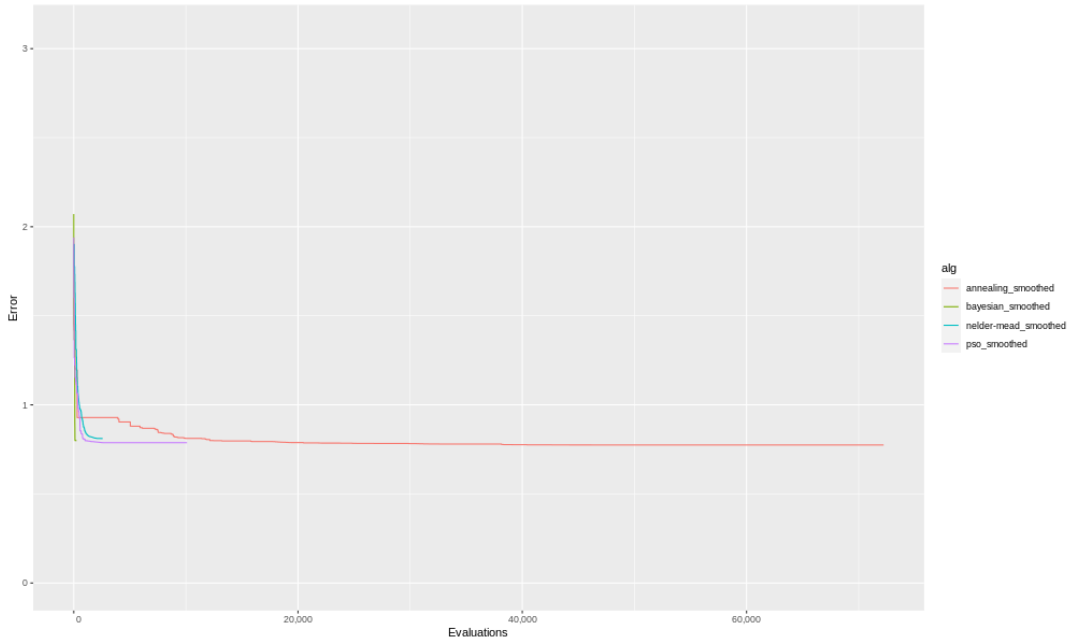

FIGURE 6.14: (Smoothed) errors on n_matrices = 3 by method

FIGURE 6.15: Errors on n_matrices = 4 by method

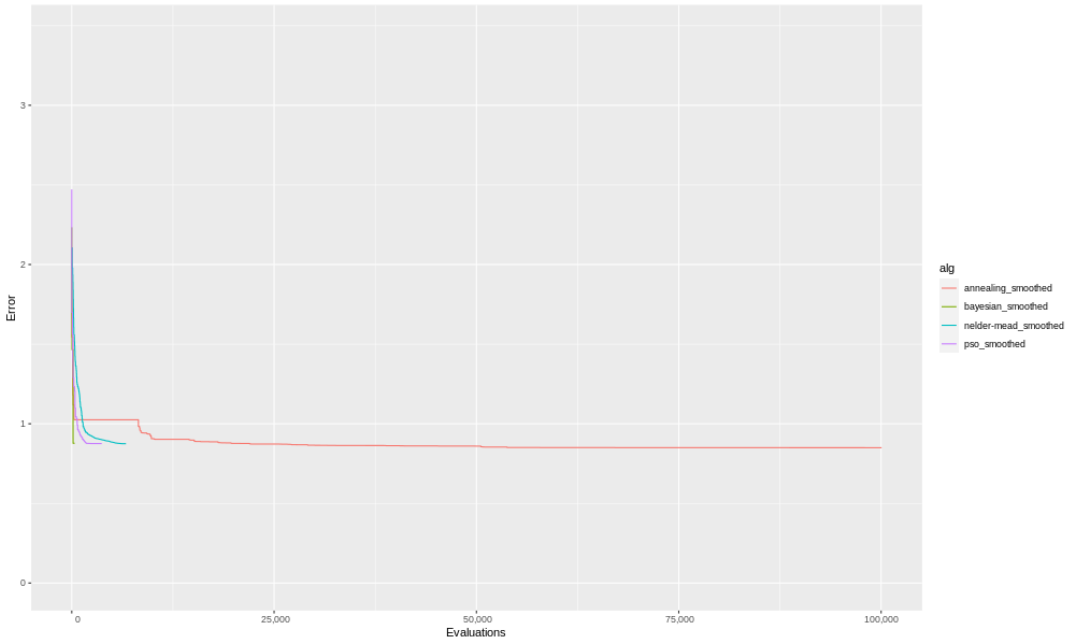

FIGURE 6.16: (Smoothed) errors on n_matrices = 4 by method

FIGURE 6.17: Errors on n_matrices = 5 by method

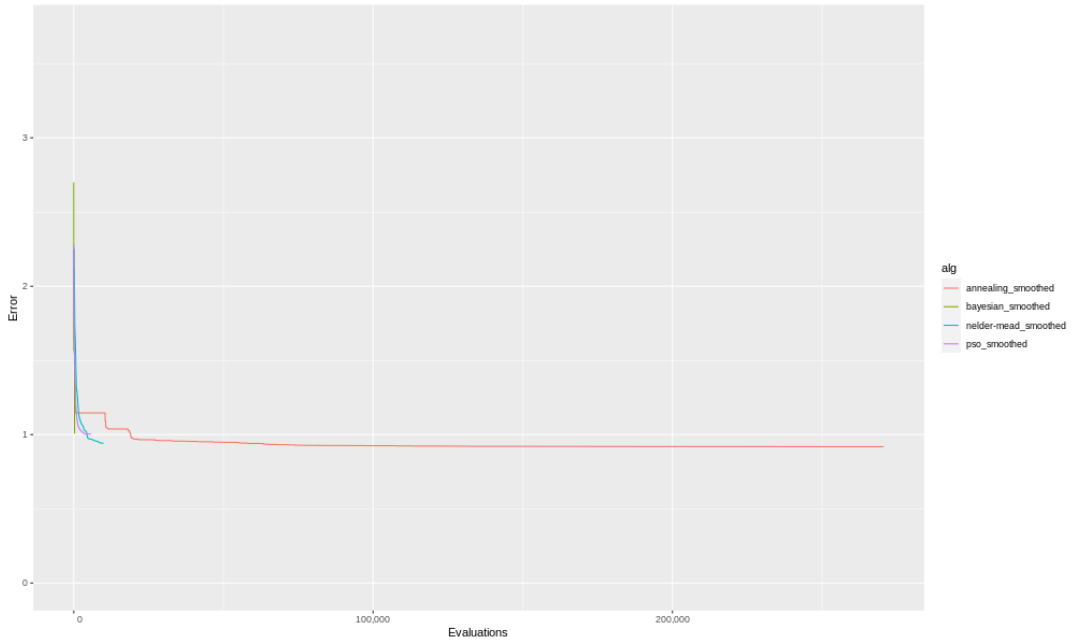

FIGURE 6.18: (Smoothed) errors on n_matrices = 5 by method

FIGURE 6.19: Errors on n_matrices = 6 by method

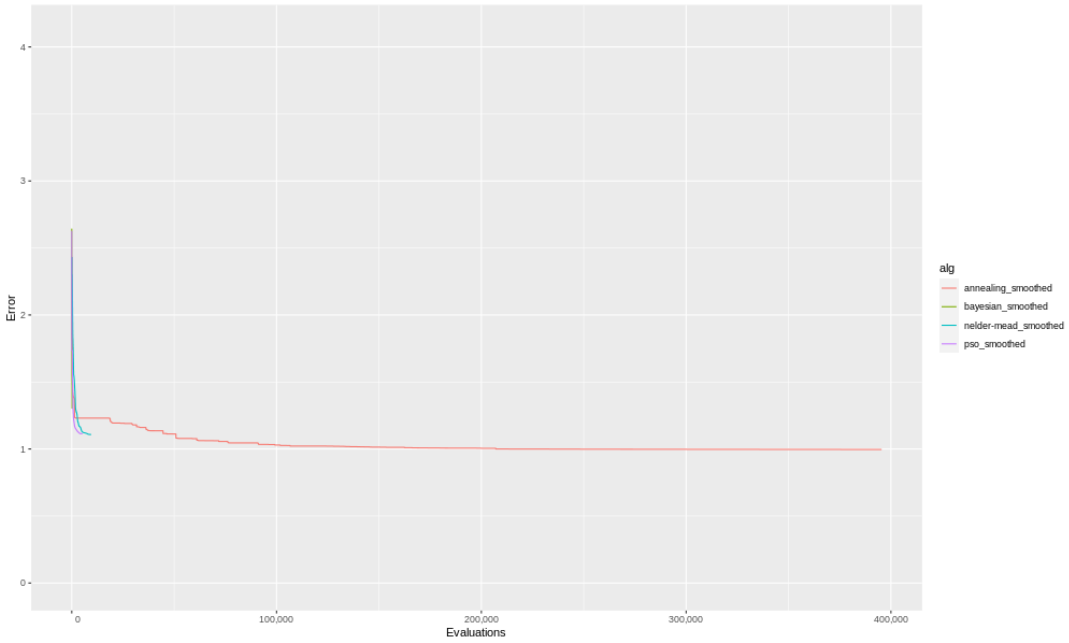

FIGURE 6.20: (Smoothed) errors on n_matrices = 6 by method

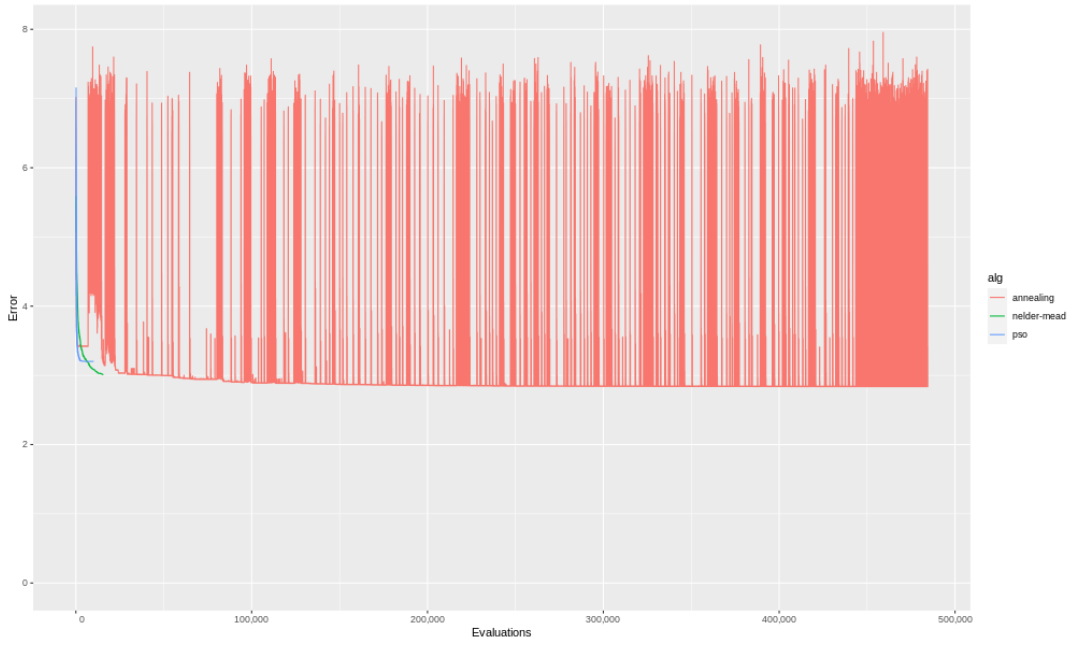

FIGURE 6.21: Errors on n_matrices = 7 by method

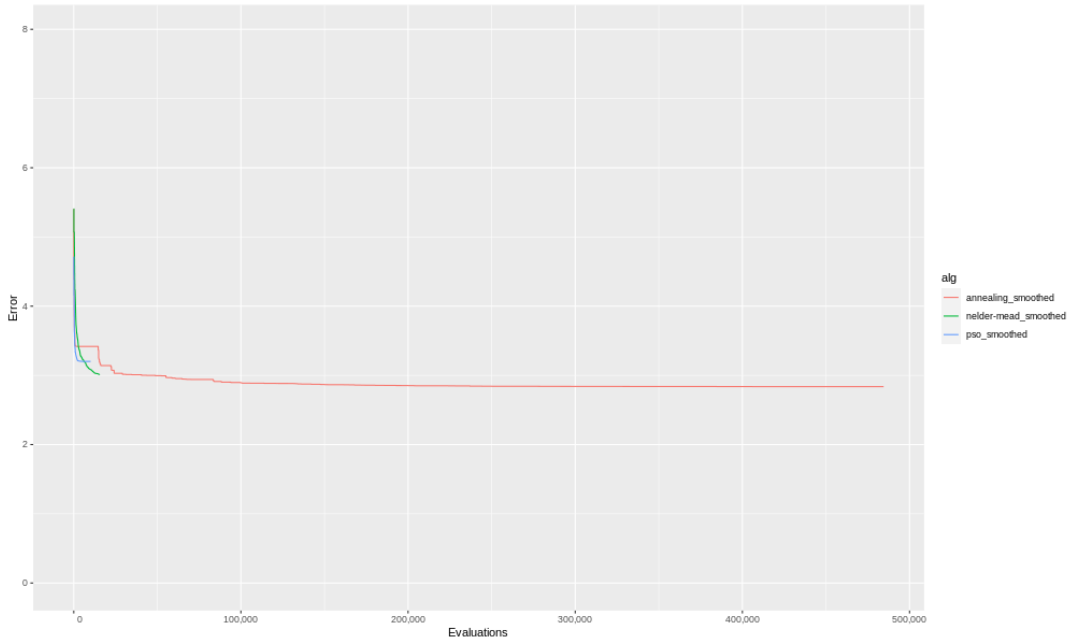

FIGURE 6.22: (Smoothed) errors on n_matrices = 7 by method

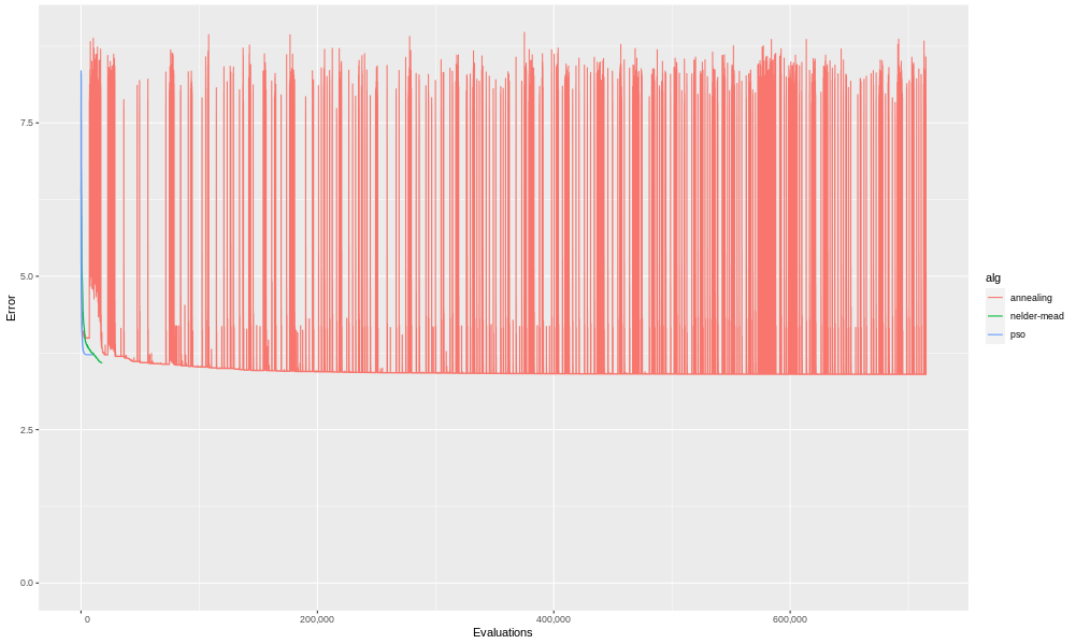

FIGURE 6.23: Errors on n_matrices = 8 by method

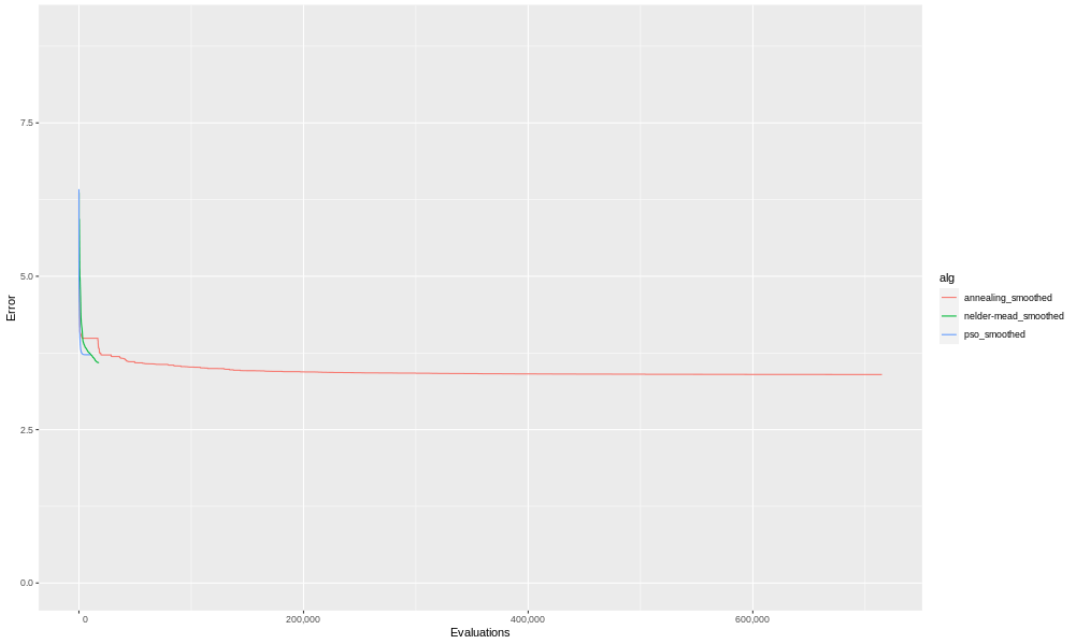

FIGURE 6.24: (Smoothed) errors on n_matrices = 8 by method

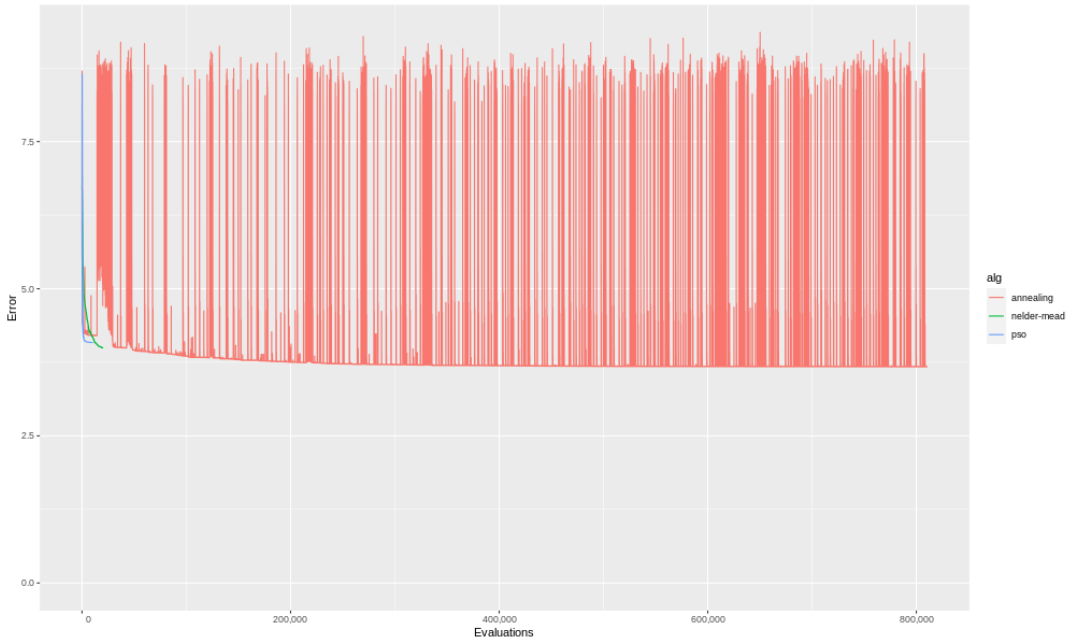

FIGURE 6.25: Errors on n_matrices = 9 by method

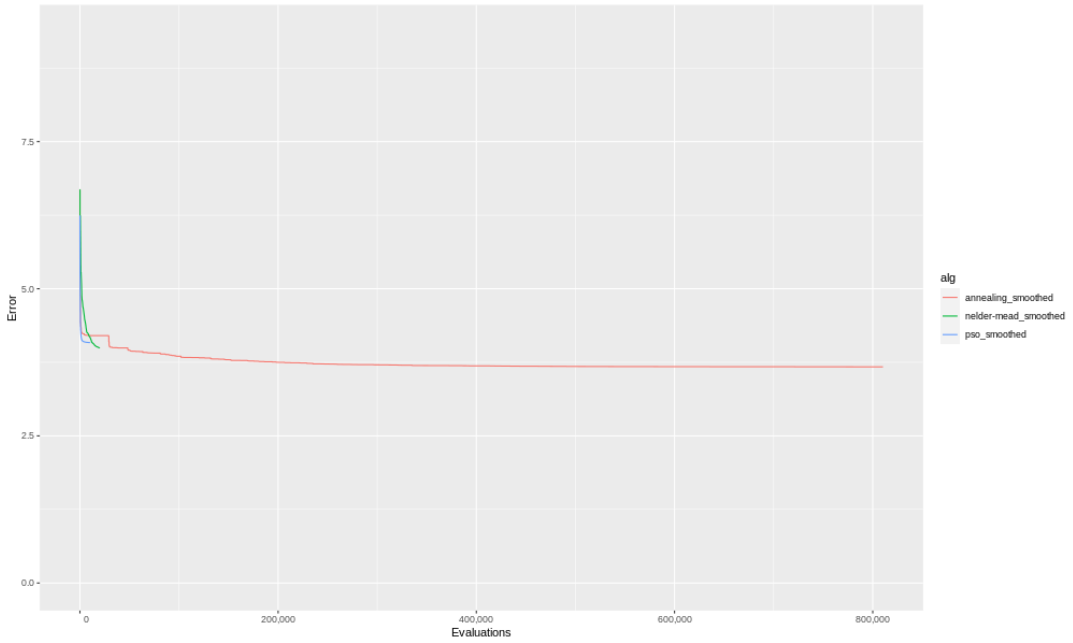

FIGURE 6.26: (Smoothed) errors on n_matrices = 9 by method

6.3 Bayesian Optimization (BO) tests

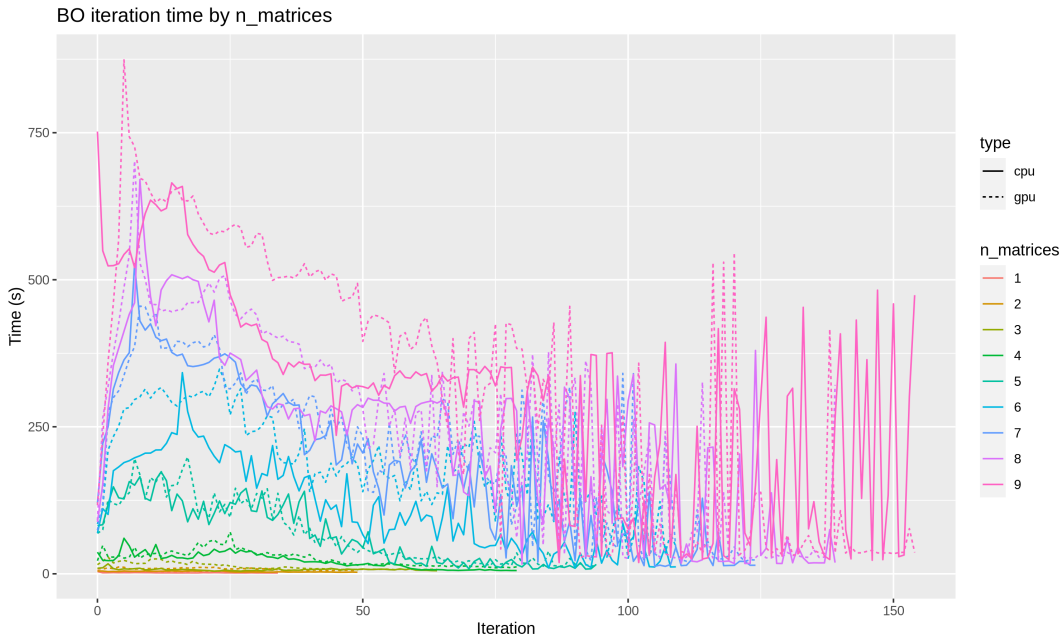

FIGURE 6.27: BO iteration time. It increases when the dimensionality increases (different colors) but surprisingly the first iterations take more time than the latter ones, where the opposite is expected. This is a transitory effect as can be seen in figure 6.28, using a larger number of iterations. Another surprising outcome is the GPU computation (using a NVIDIA Quadro P620) takes more time than the CPU, but figure 6.31 shows that the differences are not likely to be significant in the long run. With a powerful GPU this result is expected to change.

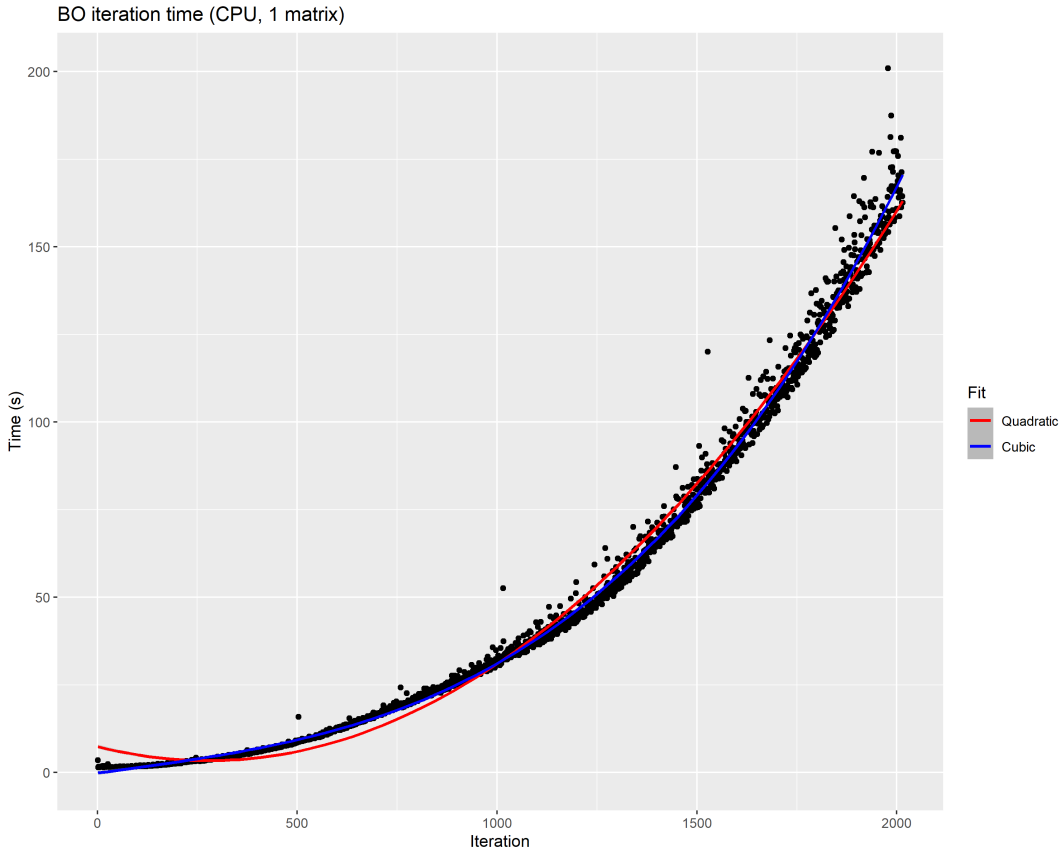

FIGURE 6.28: CPU BO iteration time using a larger number of iterations. It is clear that the iteration times follow a cubic polynomial trend, due to the matrix inversion required in the gaussian process regression stage of the optimization procedure, as seen in figure 6.30.

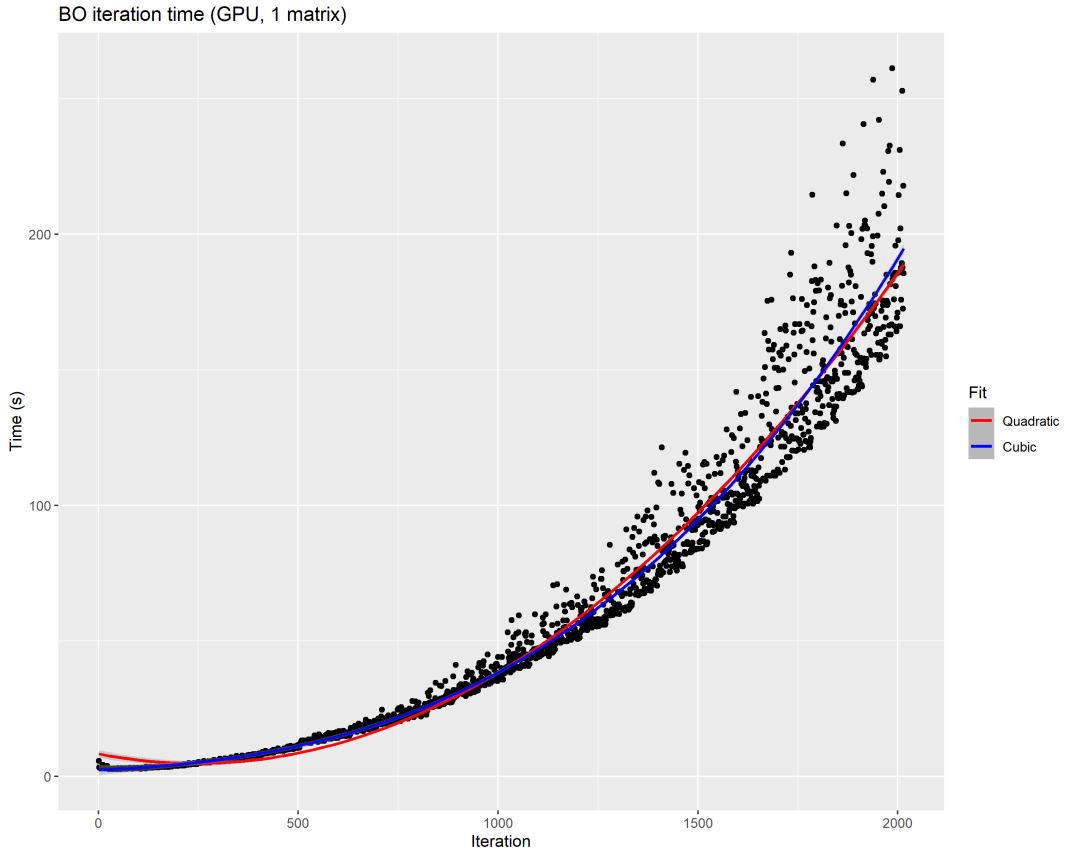

FIGURE 6.29: GPU BO iteration time using a larger number of iterations. It is clear as well that the iteration times follow a cubic polynomial trend as the CPU optimization, but with a larger variance in the last iterations.

$$f^*|X,y,X^* \sim \mathcal{N}(\bar{f}^*,\Sigma^*)$$

$$\bar{f}^* = \mu^* + K(X^*,X)[K(X,X)+\sigma_n^2 I]^{-1}(y-\mu)$$

$$\Sigma^* = K(X^*,X^*) - K(X^*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X^*)$$

Predictive posterior distribution for GPR [1]

The predictions are the means f_bar*, and variances can be obtained from the diagonal of the covariance matrix Σ*. Notice that calculation of the mean and variance requires the inversion of the K matrix, which scales with the number of training points cubed.

FIGURE 6.30: Gaussian process predictive posterior distribution

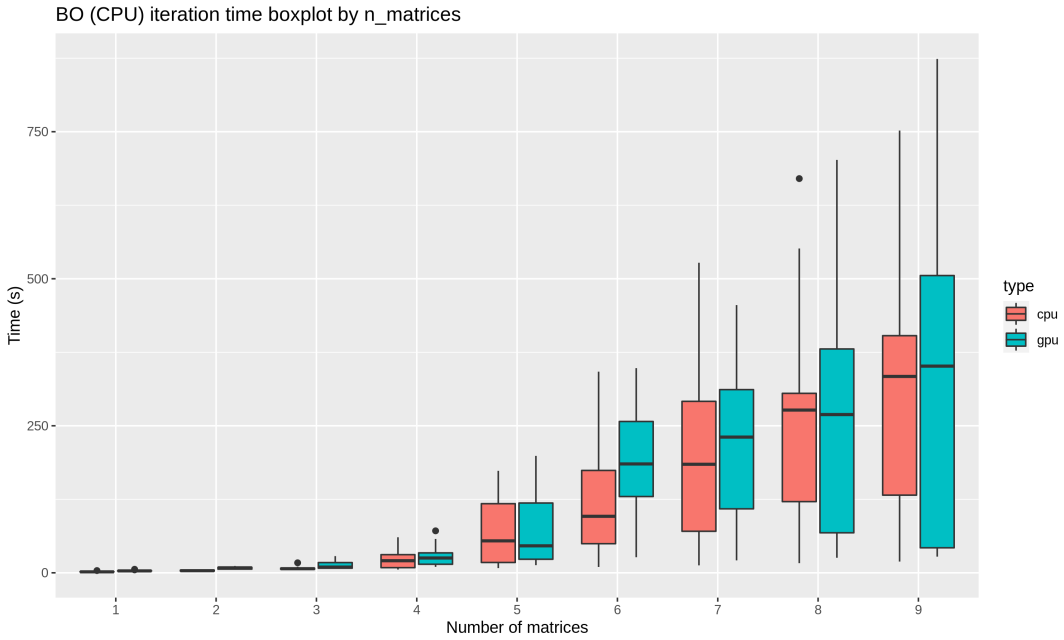

FIGURE 6.31: BO iteration time boxplot by number of matrices and computation paradigm. No significant differences between CPU/GPU are apparent; the higher values with 6 matrices using the GPU is likely to be due to chance.

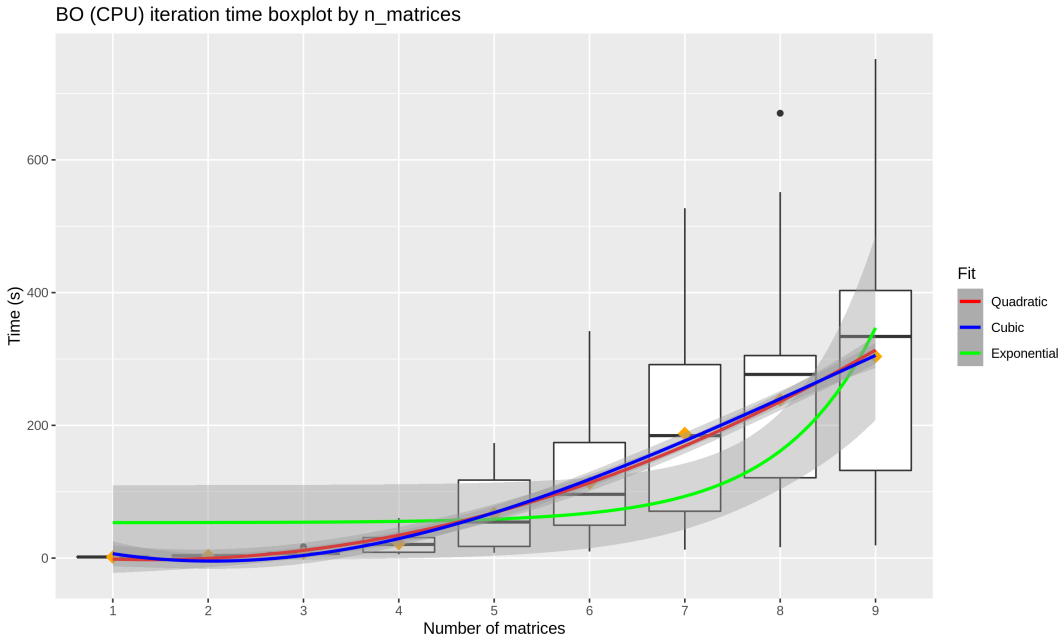

FIGURE 6.32: CPU BO iteration time boxplot by number of matrices using the CPU with the means displayed in orange. As the number of matrices (and hence dimensions in the search space) increase the iteration time follows a quadratic trend instead of the expected exponential curve (?).

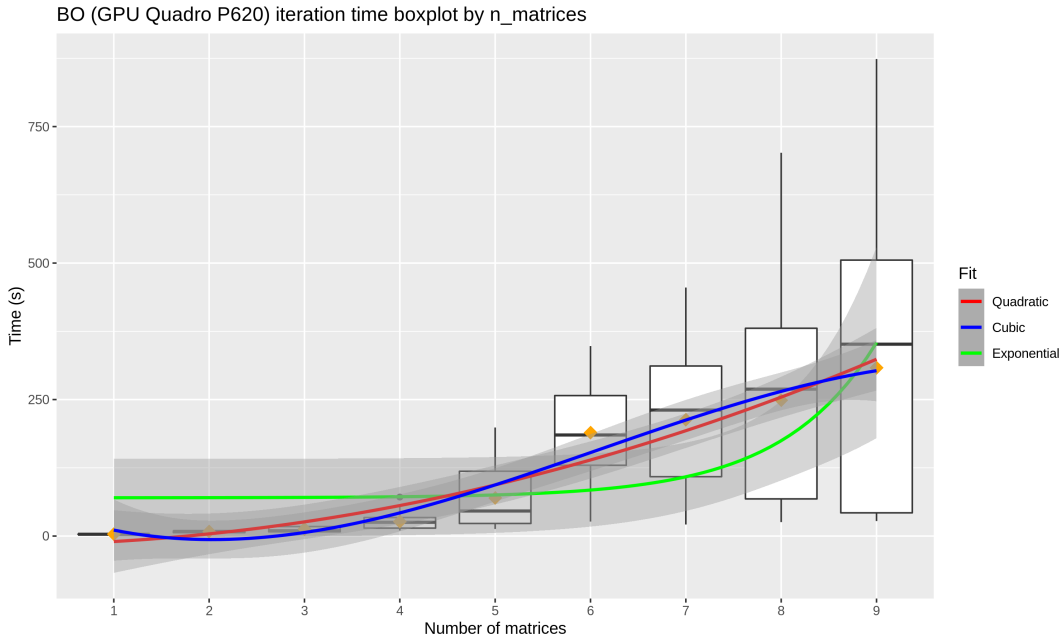

FIGURE 6.33: GPU BO iteration time boxplot by number of matrices using the CPU with the means displayed in orange. As the number of matrices (and hence dimensions in the search space) increase the iteration time follows a quadratic trend instead of the expected exponential curve (?).

List of Abbreviations

QALY	**Q**uality **A**djusted **L**ife **Y**ear
CEA	**C**ost **E**ffectiveness **A**nalysis
WTP	**W**illingness **T**o **P**ay

Bibliography

Bergstra, James et al. (2011). "Algorithms for Hyper-Parameter Optimization". In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc. URL: https://papers.nips.cc/paper/2011/hash/86e8f7ab32cfd12577bc2619bc635690-Abstract.html (visited on 11/17/2021).

Das, Sourish, Sasanka Roy, and Rajiv Sambasivan (2015). "Fast Gaussian Process Regression for Big Data". In: *CoRR* abs/1509.05142. arXiv: 1509.05142. URL: http://arxiv.org/abs/1509.05142.

Daulton, Samuel, Maximilian Balandat, and Eytan Bakshy (2021). *Parallel Bayesian Optimization of Multiple Noisy Objectives with Expected Hypervolume Improvement*. DOI: 10.48550/ARXIV.2105.08195. URL: https://arxiv.org/abs/2105.08195.

Duvenaud, David, Hannes Nickisch, and Carl Edward Rasmussen (Dec. 19, 2011). "Additive Gaussian Processes". In: *arXiv:1112.4394 [cs, stat]*. arXiv: 1112.4394. URL: http://arxiv.org/abs/1112.4394 (visited on 02/25/2022).

Duvenaud, David et al. (May 13, 2013). *Structure Discovery in Nonparametric Regression through Compositional Kernel Search*. DOI: 10.48550/arXiv.1302.4922. arXiv: 1302.4922[cs,stat]. URL: http://arxiv.org/abs/1302.4922 (visited on 08/08/2022).

Duvenaud, David Kristjanson (n.d.). "Automatic Model Construction with Gaussian Processes". In: (), p. 157.

Gardner, Jacob R, Matt J Kusner, and Gardner Jake (n.d.). "Bayesian Optimization with Inequality Constraints". In: (), p. 9.

Gonzalez, Javier et al. (n.d.). "Batch Bayesian Optimization via Local Penalization". In: (), p. 10.

Hernández-Lobato, José Miguel et al. (Sept. 4, 2016). "A General Framework for Constrained Bayesian Optimization using Information-based Search". In: *arXiv:1511.09422 [stat]*. arXiv: 1511.09422. URL: http://arxiv.org/abs/1511.09422 (visited on 10/27/2021).

Lam, Remi and Karen Willcox (n.d.). "Lookahead Bayesian Optimization with Inequality Constraints". In: (), p. 11.

Nardi, Luigi et al. (2019). "HyperMapper: a Practical Design Space Exploration Framework". In: *27th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2019, Rennes, France, October 21-25, 2019*. IEEE Computer Society, pp. 425–426. DOI: 10.1109/MASCOTS.2019.00053. URL: https://doi.org/10.1109/MASCOTS.2019.00053.

Nguyen, Vu et al. (Apr. 15, 2017). *Budgeted Batch Bayesian Optimization With Unknown Batch Sizes*. DOI: 10.48550/arXiv.1703.04842. arXiv: 1703.04842[cs]. URL: http://arxiv.org/abs/1703.04842 (visited on 08/08/2022).

Paulson, Joel A. and Congwen Lu (May 10, 2021). "COBALT: COnstrained Bayesian optimizAtion of computationaLly expensive grey-box models exploiting derivaTive information". In: *arXiv:2105.04114 [math]*. arXiv: 2105.04114. URL: http://arxiv.org/abs/2105.04114 (visited on 10/27/2021).

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. OCLC: ocm61285753. Cambridge, Mass: MIT Press. 248 pp. ISBN: 978-0-262-18253-9.

Repický, Jakub, Zbyneˇk Pitra, and Martin Holenˇa (n.d.). "Automated Selection of Covariance Function for Gaussian Process Surrogate Models". In: (), p. 8.

Souza, Artur et al. (Apr. 19, 2021). "Bayesian Optimization with a Prior for the Optimum". In: *arXiv:2006.14608 [cs, stat]*. arXiv: 2006.14608. URL: http://arxiv.org/abs/2006.14608 (visited on 10/27/2021).

Stark, Philip B. (Jan. 2015). "Constraints versus Priors". In: *SIAM/ASA Journal on Uncertainty Quantification* 3.1, pp. 586–598. ISSN: 2166-2525. DOI: 10.1137/130920721. URL: http://epubs.siam.org/doi/10.1137/130920721 (visited on 11/15/2021).

Swiler, Laura et al. (2020). "A Survey of Constrained Gaussian Process Regression: Approaches and Implementation Challenges". In: *Journal of Machine Learning for Modeling and Computing* 1.2, pp. 119–156. ISSN: 2689-3967. DOI: 10.1615/JMachLearnModelComput.2020035155. arXiv: 2006.09319[cs,math,stat]. URL: http://arxiv.org/abs/2006.09319 (visited on 08/03/2022).

Terry, Nick and Youngjun Choe (Aug. 2021). "Splitting Gaussian processes for computationally-efficien regression". In: *PLOS ONE* 16.8, pp. 1–17. DOI: 10.1371/journal.pone.0256470. URL: https://doi.org/10.1371/journal.pone.0256470.

Ungredda, Juan and Juergen Branke (May 27, 2021). "Bayesian Optimisation for Constrained Problems". In: *arXiv:2105.13245 [cs, stat]*. arXiv: 2105.13245. URL: http://arxiv.org/abs/2105.13245 (visited on 10/27/2021).

Wang, Jialei et al. (2016). *Parallel Bayesian Global Optimization of Expensive Functions*. DOI: 10.48550/ARXIV.1602.05149. URL: https://arxiv.org/abs/1602.05149.