# Bayesian Optimization using Partially Observable Gaussian Process Network

Saksham Kiroirwal[1], Julius Pfrommer[1], and Jürgen Beyerer[1,2]

[1]Cognitive Industrial Systems, Fraunhofer IOSB, Karlsruhe, Germany
[2]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{saksham.kiroirwal, julius.pfrommer, juergen.beyerer}@iosb.fraunhofer.de

## Abstract

Bayesian Optimization (BO) is a highly successful technique for the optimization of noisy functions, but it can be difficult to scale. Gaussian Process Networks (GPN) replace a single black-box approximation with a network of connected noisy functions. This exploits sparsity in the causal links between process variables and improves the sample complexity for approximating the target function. In addition to the GPN setting, in many real-world process networks, we observe not only the final output but also intermediate observations. For example, by adding sensor instrumentation within a "black-box" process network. These intermediate observations are often incomplete and noisy. We propose Partially Observable Gaussian Process Network (POGPN), and its inference method, which addresses the limitations of GPN by handling noisy observations and uncertainty propagation while incorporating the structural knowledge of the process network. We empirically show superior performance of POGPN for Bayesian Optimization using benchmark functions and an ODE-based Penicillin production process.

## 1 Introduction

Suboptimal processes and operational inefficiencies are among the major factors contributing to losses across industries. In manufacturing alone, suboptimal processes can cause unplanned downtime, resulting in staggering annual losses of billions of dollars [10], and leading to higher resource utilization. Optimization of such complex processes has been at the core of operations research. Process optimization that focuses on reducing resource usage while maximizing quality can provide significant economic benefits [16]. Bayesian optimization (BO) [7] is a powerful tool for black-box optimization that has been successfully applied from business operations [20] to complex manufacturing [14].

Conventionally, a process is modeled as a single black-box $\mathbf{P}$ represented by the dashed box in Figure 1. However, most systems are composed of multiple interconnected subprocesses $\{P^{(w)}\}_{w\in\mathcal{W}}$. We assume their causal links form a directed acyclic graph (DAG) $\mathcal{G}$. Using additional sensors, we can observe intermediate output $\mathbf{y}^{(w)}$ from a subprocess $w$, as illustrated in Figure 1. The subprocess dependencies are a crucial form of prior knowledge that can help reduce the curse of dimensionality.

Section 2 explains the definition of a process network with partial observability. Works by [2, 19, 12] introduced a Gaussian Process Network (GPN) for BO of such process networks. As discussed in Appendix A.5, the assumptions about GPNs by [2, 19, 12, 4] pose limitations. Our POGPN overcomes these limitations, leading to improved performance in high-dimensional settings or when intermediate observations are noisy, as shown in Section 3. The experimental results are discussed in Section 4.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ML x OR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

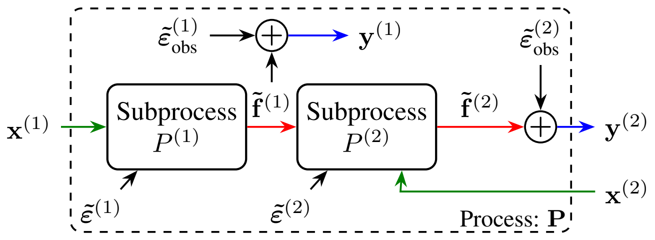
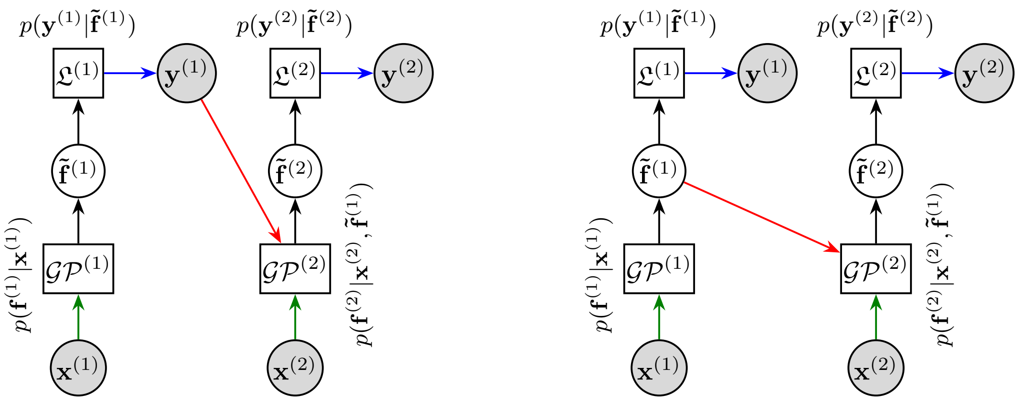

$\tilde{\varepsilon}^{(1)}_{\text{obs}}$
$\mathbf{y}^{(1)}$
$\tilde{\varepsilon}^{(2)}_{\text{obs}}$
$\mathbf{x}^{(1)}$
Subprocess $P^{(1)}$
$\tilde{\mathbf{f}}^{(1)}$
Subprocess $P^{(2)}$
$\tilde{\mathbf{f}}^{(2)}$
$\mathbf{y}^{(2)}$
$\mathbf{x}^{(2)}$
$\tilde{\varepsilon}^{(1)}$
$\tilde{\varepsilon}^{(2)}$
Process: $\mathbf{P}$

Figure 1: Example process network comprised of two subprocesses $w \in \{1, 2\}$ with $\tilde{\mathbf{f}}^{\mathcal{P}\text{a}(1)} = \varnothing$ and $\tilde{\mathbf{f}}^{\mathcal{P}\text{a}(2)} = \tilde{\mathbf{f}}^{(1)}$. The latent output $\tilde{\mathbf{f}}^{(1)}$ and $\tilde{\mathbf{f}}^{(2)}$ is partially observable as $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$.


$p(\mathbf{y}^{(1)}|\tilde{\mathbf{f}}^{(1)})$
$p(\mathbf{y}^{(2)}|\tilde{\mathbf{f}}^{(2)})$
$\mathfrak{L}^{(1)}$
$\mathbf{y}^{(1)}$
$\mathfrak{L}^{(2)}$
$\mathbf{y}^{(2)}$
$\tilde{\mathbf{f}}^{(1)}$
$\tilde{\mathbf{f}}^{(2)}$
$\mathcal{GP}^{(1)}$
$\mathcal{GP}^{(2)}$
$p(\mathbf{f}^{(1)}|\mathbf{x}^{(1)})$
$p(\mathbf{f}^{(2)}|\mathbf{x}^{(2)}, \tilde{\mathbf{f}}^{(1)})$
$\mathbf{x}^{(1)}$
$\mathbf{x}^{(2)}$

(a) Gaussian process network

(b) Partially observable Gaussian process network

Figure 2: Mapping of Fig. 1 to a factor-graph for GPN and our POGPN. Circles represent variables, but only the gray variables are observed. Gaussian Processes and likelihoods of respective observations (factors) are denoted with rectangles. Note that the original GPN cannot model latent variables as they connect subprocesses only via observed output.

## 2 Process Networks with Partial Observability

The nodes in the DAG $\mathcal{G}$ are topologically ordered. Each subprocess $P^{(w)}$ takes as inputs, the controllable inputs $\mathbf{x}^{(w)}$ and the latent/true outputs $\tilde{\mathbf{f}}^{\mathcal{P}\text{a}(w)}$ from parent nodes. A subprocess maps its inputs to a latent/true output *instance* $\tilde{\mathbf{f}}^{(w)}$. The output instance is further obscured by measurement noise. The observed output $\mathbf{y}^{(w)}$ is thus a noisy observation of the latent state. We formally define each subprocess $P^{(w)}$ using the tuple of distribution over process function and observed outputs as

$$P^{(w)} = \left\langle \mathcal{GP}^{(w)},\ \mathfrak{L}^{(w)} \right\rangle = \left\langle p\big(\mathbf{f}^{(w)}|\mathbf{x}^{(w)}, \tilde{\mathbf{f}}^{\mathcal{P}\text{a}(w)}\big),\ p\big(\mathbf{y}^{(w)}|\mathbf{f}^{(w)}\big) \right\rangle . \tag{1}$$

The GPN implemented by [2, 19, 12] models the tuple for each node $w$ as a $\mathcal{GP}^{(w)}$ and its likelihood of observation $\mathfrak{L}^{(w)}$. GPN uses closed-form MLL for inference of each node GP, as explained in Appendix A.5. This restricts the model to only Gaussian likelihoods, and the posterior calculation using Monte Carlo (MC) samples does not match the training with deterministic inputs. We address this by introducing POGPN as shown in Figure 2b and developing the Evidence Lower BO (ELBO) objective for POGPN, conditioned on the graph $\mathcal{G}$, as shown in Appendix A.6. This introduces the process stochasticity into the training and posterior both, thus bridging the limitation of existing GPN and making a more robust model.[1].

The goal is to find the optimal input $\mathbf{x}_{\text{best}} \in \bigcup_{w \in \mathcal{W}} \mathcal{X}^{(w)}$ that reduces the instantaneous regret as fast as possible within a budget of $B$ trial evaluations. The rapid minimization of regret is crucial for industries where we need to adapt to changing conditions rapidly or we need to optimize a high-dimensional black-box process network within a fixed budget.

[1]POGPN package is available at `https://github.com/Sam4896/pogpn`

## 3 Experiments

We benchmark POGPN against two state-of-the-art models: GPN [2] and STGP [9]. All results are averaged over three independent runs each. The evaluation uses benchmark functions decomposed into process networks of varying dimensions ($D_x \in \{25, 100, 200\}$), as shown in Appendix A.7, with a fixed BO budget of $B = 100$ and different additive process noises $\varepsilon^{(w)}$. All models are implemented in BoTorch [3] using a squared exponential kernel, a Gaussian likelihood, and dimensionally scaled priors, as proposed in [9]. For POGPN, the number of inducing points for a node is the same as the respective training data size, and inducing points are initialized using Greedy Improvement Reduction [15] for the objective node and Greedy Variance Reduction [5] for all others. We use coordinate descent using the Adam optimizer (lr $= 0.02$) with an exponential moving average (window = 25, $\eta = 0.95$) stopping criteria on both ELBO and PLL objectives with $\beta = 1.0$. We use the qLogEI [1] acquisition function.

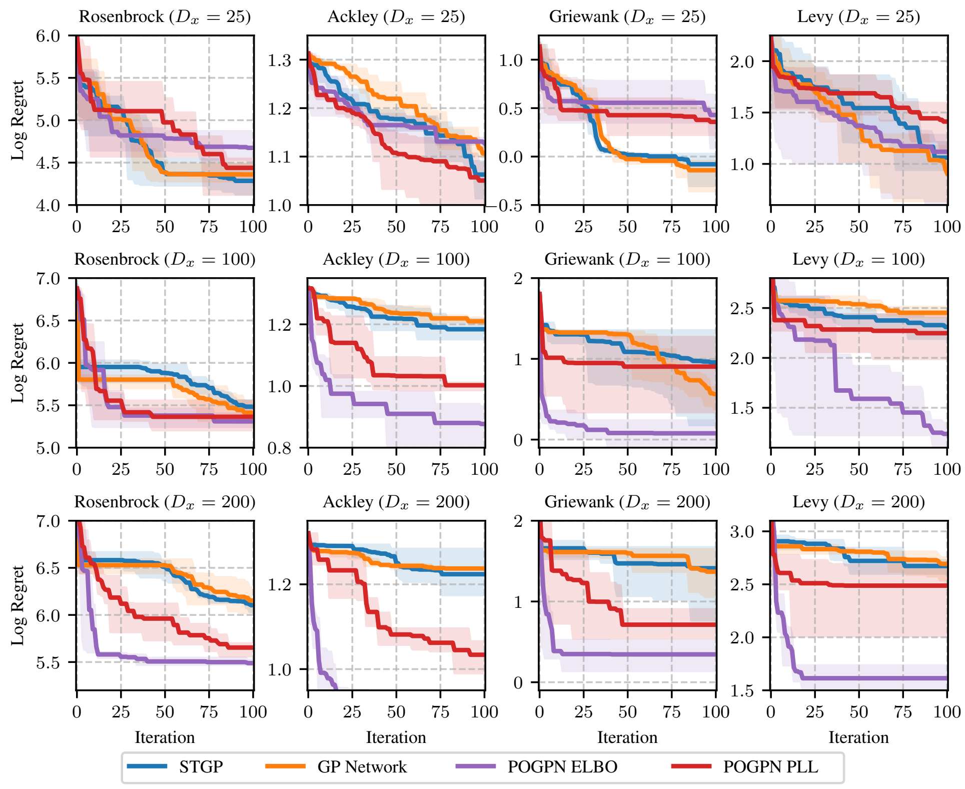


Figure 3: Lower noise $\tilde{\varepsilon}^{(w)}_{\text{obs}}, \tilde{\varepsilon}^{(w)} \sim \mathcal{N}(0, 0.1)$, $B = 100$ and $b_{\text{init}} = 2D_x$ (max 100).

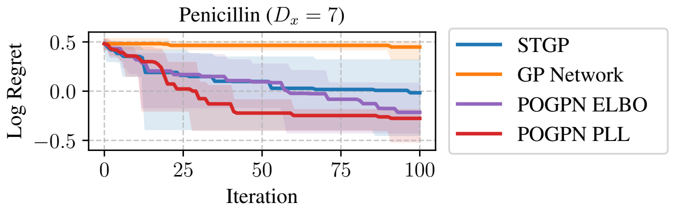


Figure 5: Results for the Penicillin production process.

We further evaluate POGPN on a stochastic batch Penicillin production process [13], detailed in Appendix A.7.1. The process takes seven inputs ($D_x = 7$), consisting of initial culture conditions and fermentation operating parameters. During production, we observe a time series of five process states, alongside the total production time. Our goal is to find the optimal inputs to maximize the final penicillin concentration. We model this as a three-node network: the inputs form the root node, four dimensional intermediate node, modelled using multi-task Gaussian process, consisting of (biomass concentration, $CO_2$ concentration, culture volume at the end of production and the total production time taken), and the final objective node as the produced Penicillin concentration at the end of production.

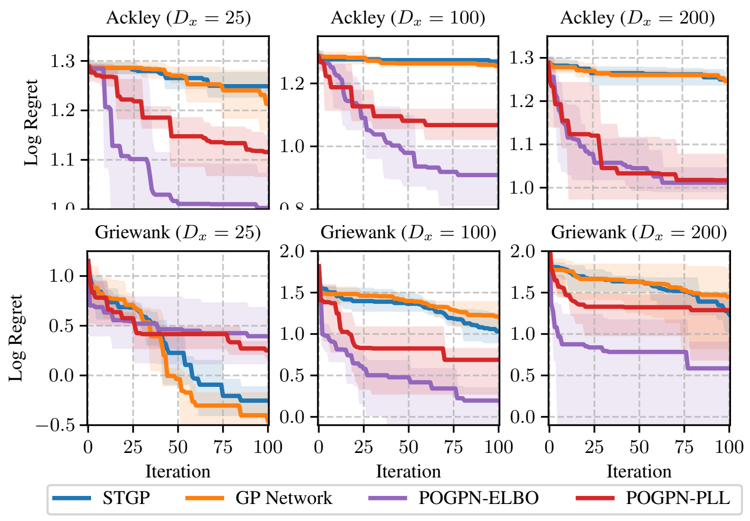


Figure 4: Higher noise $\tilde{\varepsilon}_{\text{obs}}^{(w)}, \tilde{\varepsilon}^{(w)} \sim \mathcal{N}(0, 0.5)$, $B = 100$ and fixed $b_{\text{init}} = 50$.

## 4 Discussion and Future Work

Figure 3, 4, and 5 show that POGPN can achieve lower regret faster than other models, for both higher noise levels and high dimensions, across various benchmarks. In most of the benchmarks, POGPN is mostly able to find the optima or is close to the other best models. It can also be seen that the performance of GPN is suboptimal in comparison to the other models. Thus, it shows that not only the process network structure but also the correct inference method plays a crucial role in the performance of Bayesian optimization. In the future, the effect of hyperparameters, such as regularization strength $\beta$ and the number of inducing points, on the performance of POGPN for BO can be explored.

We propose Partially Observable Gaussian Process Network (POGPN), a novel methodology at the intersection of Gaussian processes and probabilistic graphical models, that can be a valuable tool for Machine Learning and Operations Research. POGPN assumes a joint latent distribution of subprocesses. This is different from the existing Gaussian process networks that assume an independent latent distribution of subprocesses. The inference method, POGPN-ELBO and POGPN-PLL, performed via Monte Carlo sampling, helps in modeling an "uncertainty-aware" model network. Our work demonstrates how embedding structural domain knowledge of the process network with our inference method can serve as a valuable form of prior knowledge for Bayesian optimization in high-dimensional input spaces and partial observability, not just for process networks but also for complex differential equation systems. The methodology significantly enhances the sample efficiency of Bayesian optimization. However, since POGPN employs variational inference, its performance can be improved by utilizing more sophisticated inference methods for larger process networks.

## Acknowledgement

The research in this work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 459291153 (FOR5399).

## References

[1] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023.

[2] Raul Astudillo and Peter Frazier. Bayesian optimization of function networks. *Advances in neural information processing systems*, 34:14463–14475, 2021.

[3] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[4] Poompol Buathong, Jiayue Wan, Raul Astudillo, Samuel Daulton, Maximilian Balandat, and Peter I Frazier. Bayesian optimization of function networks with partial evaluations. *arXiv preprint arXiv:2311.02146*, 2023.

[5] David R Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Convergence of sparse variational inference in gaussian processes regression. *Journal of Machine Learning Research*, 21(131):1–63, 2020.

[6] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *Artificial intelligence and statistics*, pages 207–215. PMLR, 2013.

[7] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[8] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[9] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions. *arXiv preprint arXiv:2402.02229*, 2024.

[10] IDS-INDATA. The real cost of manufacturing downtime (2020–2025): Sector insights & forecast. Technical report, 2025. Accessed: 2025-08-26.

[11] Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. Parametric gaussian process regressors. In *International conference on machine learning*, pages 4702–4712. PMLR, 2020.

[12] Shunya Kusakawa, Shion Takeno, Yu Inatsu, Kentaro Kutsukake, Shogo Iwazaki, Takashi Nakano, Toru Ujihara, Masayuki Karasuyama, and Ichiro Takeuchi. Bayesian optimization for cascade-type multistage processes. *Neural Computation*, 34(12):2408–2431, 2022.

[13] Qiaohao Liang and Lipeng Lai. Scalable bayesian optimization accelerates process optimization of penicillin production. In *NeurIPS 2021 AI for Science Workshop*, 2021.

[14] Hendrik Mende, Lukas Jäschke, Gustavo Schewinski, Dennis Grunert, and Robert H Schmitt. Multi-objective lookahead bayesian optimization for process parameter optimization in non-isothermal glass molding. *Procedia CIRP*, 134:366–371, 2025.

[15] Henry B Moss, Sebastian W Ober, and Victor Picheny. Inducing point allocation for sparse gaussian processes in high-throughput bayesian optimisation. In *International Conference on Artificial Intelligence and Statistics*, pages 5213–5230. PMLR, 2023.

[16] Markus Niessen. First-time-right production with bayesian process optimization. 2023. Data Science and Measurement Technology, Annual Report.

[17] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[18] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. *Advances in neural information processing systems*, 30, 2017.

[19] Scott Sussex, Anastasiia Makarova, and Andreas Krause. Model-based causal bayesian optimization. *arXiv preprint arXiv:2211.10257*, 2022.

[20] Junbo Wang. Bayesian optimization for adaptive network reconfiguration in urban delivery systems. In *2025 8th International Symposium on Big Data and Applied Statistics (ISBDAS)*, pages 195–198. IEEE, 2025.

## A Appendix

### A.1 Gaussian Process and Variational Inference

A Gaussian process (GP) defines a distribution over functions [17]. For a dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, a GP with a Gaussian likelihood $p(y_n|f_n) = \mathcal{N}\big(f_n, \sigma_\varepsilon^2\big)$ specifies a Gaussian distribution over the outputs. The prior over the function value $f_n$ at an input $\mathbf{x}_n$ is $p(f_n|\mathbf{x}_n) = \mathcal{N}\big(m(\mathbf{x}_n), k(\mathbf{x}_n, \mathbf{x}_n')\big)$, with mean function $m(\cdot)$ and covariance kernel $k(\cdot, \cdot')$. GP hyperparameters are optimized by maximizing the marginal log-likelihood (MLL)

$$\mathcal{L}_{\text{GP}} = -\sum_{n=1}^{N} \log \mathbb{E}_{p(f_n|\mathbf{x}_n)}\big[p(y_n|f_n)\big]. \tag{2}$$

GPs are not scalable for large datasets, and inference is intractable for hierarchical models [17]. Stochastic Variational Gaussian Processes (SVGP) [8] address this by introducing $I$ inducing locations $\mathbf{Z} = \{\mathbf{z}_i \in \mathcal{X}\}_{i=1}^I$. The function evaluations at $\mathbf{Z}$ are called inducing points $\mathbf{u}$ have the prior

$$p(\mathbf{u}; \mathbf{Z}) = \mathcal{N}\big(m(\mathbf{Z}), k(\mathbf{Z}, \mathbf{Z}')\big).$$

The inducing points are assumed to have a variational marginal distribution $q(\mathbf{u}) = \mathcal{N}\big(\boldsymbol{\mu}_\mathbf{u}, \boldsymbol{\Sigma}_\mathbf{u}\big)$. For input $\mathbf{x}_n$, SVGP approximates the true GP posterior with a variational distribution $q(f_n)$ conditioned on the inducing points as

$$q(f_n|\mathbf{x}_n, \mathbf{Z}) = \mathbb{E}_{q(\mathbf{u})}\big[p(f_n|\mathbf{u})\big] = \mathcal{N}\big(\mu_{q(f)}, \Sigma_{q(f)}\big), \tag{3}$$

where

$$\begin{aligned} \mu_{q(\mathbf{f})} &= m(\mathbf{x}_n) + \boldsymbol{\Phi}(\mathbf{x}_n)^\top \big(\boldsymbol{\mu}_\mathbf{u} - m(\mathbf{Z})\big), \\ \Sigma_{q(\mathbf{f})} &= k(\mathbf{x}_n, \mathbf{x}_n) - \boldsymbol{\Phi}(\mathbf{x}_n)^\top \big(k(\mathbf{Z}, \mathbf{Z}) - \boldsymbol{\Sigma}_\mathbf{u}\big)\boldsymbol{\Phi}(\mathbf{x}_n), \\ \boldsymbol{\Phi}(\mathbf{x}_n) &= k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{x}_n). \end{aligned} \tag{4}$$

The GP hyperparameters and variational parameters ($\mathbf{Z}$, $\boldsymbol{\mu}_\mathbf{u}$, $\boldsymbol{\Sigma}_\mathbf{u}$) are optimized by maximizing the Evidence Lower Bound (ELBO) as

$$\mathcal{L}_{\substack{\text{SVGP}\\\text{ELBO}}} = -\sum_{n=1}^{N} \Big[\mathbb{E}_{q(f_n)}[\log p(y_n|f_n)]\Big] + \beta \text{KL}\big(q(\mathbf{u})||p(\mathbf{u})\big), \tag{5}$$

where $\text{KL}(\cdot, \cdot)$ is the Kullback-Leibler (KL) divergence. Alternatively, the Predictive Log-Likelihood (PLL) introduced by [11] often offers better predictive performance

$$\mathcal{L}_{\substack{\text{SVGP}\\\text{PLL}}} = -\sum_{n=1}^{N} \Big[\log \mathbb{E}_{q(f_n)}[p(y_n|f_n)]\Big] + \beta \text{KL}\big(q(\mathbf{u})||p(\mathbf{u})\big). \tag{6}$$

### A.2 Deep Gaussian Process

A Deep Gaussian Process (DGP) [6] stacks GPs in hierarchical layers $\{\mathbf{f}^{(l)} \in \mathbb{R}^{D_l}\}_{l=1}^L$, where the output of one layer, with latent dimension $D_l$, is the input to the next. One of the primary inference methods is doubly stochastic variational inference [18]. Similar to SVGP, inducing points $\{\mathbf{U}^{(l)} \in \mathbb{R}^{I(l) \times D_l}\}_{l=1}^L$ are placed at locations $\{\mathbf{Z}^{(l-1)} \in \mathbb{R}^{I(l) \times D_{l-1}}\}_{l=1}^L$. Here, $I(l)$ is the number of inducing points for layer $l$ and $D_0 = D_x$ is the input dimension. Using $\mathbf{f}_n^{(0)} = \mathbf{x}_n$, the marginal distribution $q(\mathbf{f}_n^{(L)})$ for layer $L$ is found using (3) and (4) as

$$q(\mathbf{f}_n^{(L)}) = \int \prod_{l=1}^{L} q(\mathbf{f}_n^{(l)}) \mathrm{d}\mathbf{f}_n^{(l-1)} \text{ where } q(\mathbf{f}_n^{(l)}) = q(\mathbf{f}_n^{(l)}|\mathbf{f}_n^{(l-1)}, \mathbf{Z}^{(l-1)}). \tag{7}$$

The variational posterior is found by minimizing the negative ELBO as

$$\mathcal{L}_{\substack{\text{DGP}\\\text{ELBO}}} = -\sum_{n=1}^{N} \mathbb{E}_{q\left(f_n^{(L)}\right)}\big[\log p(y_n|f_n^{(L)})\big] + \beta \sum_{l=1}^{L} \text{KL}\big(q(\mathbf{U}^{(l)})||p(\mathbf{U}^{(l)})\big). \tag{8}$$

### A.3 Bayesian Optimization: Expected Improvement (EI)

The Expected Improvement (EI) for a candidate point $\mathbf{x} \in \mathcal{X}$ and best observed output $y_\star$ is defined as $\text{EI}(\mathbf{x}_b) = \mathbb{E}_{p(y|\mathbf{x}_b,\mathcal{D})}[\max(0, y_b - y_\star)]$. For a Gaussian predictive posterior $\mathcal{N}(\mu_{p(f_b)}, \Sigma_{p(f_b)})$, EI has a closed-form solution [7]

$$\text{EI}(\mathbf{x}) = \Sigma_{p(f_b)}\, h\!\left(\frac{\mu_{p(f_b)} - y_\star}{\Sigma_{p(f_b)}}\right), \quad h(\eta) = \phi(\eta) + \eta\,\Phi(\eta),$$

where $\eta = (\mu_{p(f_b)} - y_\star)/\Sigma_{p(f_b)}$, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the Cumulative Distribution Function (CDF) and Probability Density Function (PDF) of the standard normal distribution. Numerical issues of EI were first analyzed by [1], and they proposed Log Expected Improvement (LogEI) as $\text{LogEI}(\mathbf{x}) = \log h(\eta) + \log \Sigma_{p(f_b)}$.

### A.4 Inducing Points Allocator (IPA)

The initialization of inducing locations is critical for SVGP performance. A common approach for regression is Greedy Variance Reduction (GVR) [5], which iteratively selects points from observed inputs $\mathbf{X}$ as $\mathbf{z}_i = \arg\max_{\mathbf{z}\in\mathbf{X}} \sigma_{i-1}(\mathbf{z})$, where $\sigma^2_{i-1}(\mathbf{z}) = k(\mathbf{z},\mathbf{z}) - k(\mathbf{Z}_{1:i-1},\mathbf{z})^T k(\mathbf{Z}_{1:i-1},\mathbf{Z}_{1:i-1})^{-1} k(\mathbf{Z}_{1:i-1},\mathbf{z})$.

However, GVR can be suboptimal for BO [15]. To address this, Moss et al. [15] introduced a quality function $\psi(\cdot)$ as a selection criterion for $\mathbf{x}_n$ to be selected as an inducing location, defined as $\mathbf{z}_i = \arg\max_{\mathbf{z}\in\mathbf{X}} \psi(\mathbf{z})\sigma_{i-1}(\mathbf{z})$, where setting $\psi(\cdot) = 1$ recovers GVR. For BO, they propose Greedy Improvement Reduction (GIR), where the EI-inspired quality function $\psi(\mathbf{x}_n) = y_n - y_\star$ represents the improvement over the incumbent. This prioritizes locations with high potential improvement, making GIR better suited for BO.

### A.5 Related Work and Limitations

Gaussian Process Network (GPN) or Function Network (FN), proposed by [2] and later extended by [19], models $\mathbf{P}$ as a Directed Acyclic Graph (DAG) as

$$\tilde{f}_n^{(w)} \sim \mathcal{GP}^{(w)}\left(m^{(w)}(\mathbf{a}_n^{(w)}), k^{(w)}(\mathbf{a}_n^{(w)}, \mathbf{a}_n^{(w)\prime})\right), \quad p(y_n^{(w)}|\tilde{f}_n^{(w)}) = \mathcal{N}\left(f_n^{(w)}|\sigma^2_{\text{obs},\varepsilon}\right),$$

where $\mathbf{a}_n^{(w)} = \left(\mathbf{x}_n^{(w)}, \mathbf{y}_n^{\mathcal{P}\text{a}(w)}\right)$ is the combined input vector of input variables $\mathbf{x}_n^{(w)}$ and observed parent nodes $\mathbf{y}_n^{\mathcal{P}\text{a}(w)}$ of subprocess $P^{(w)}$. The hyperparameters of independent node GPs are trained using closed-form MLL as defined in (2). Figure 2a shows the GPN for the process network in Figure 1. The predictive posterior for GPN is calculated using Monte Carlo (MC) sampling as

$$\tilde{f}_n^{(w)}(\mathbf{j}_n^{(w)}, \boldsymbol{\zeta}_{1:S}^{(w)}) = m^{(w)}(\mathbf{j}_n^{(w)}) + \boldsymbol{\zeta}_{1:S}^{(w)}\, k^{(w)}(\mathbf{j}_n^{(w)}, \mathbf{j}_n^{(w)\prime}), \quad \boldsymbol{\zeta}_{1:S}^{(w)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{9}$$

where $\mathbf{j}_n^{(w)} = \left(\mathbf{x}_n^{(w)}, \tilde{\mathbf{f}}_n^{\mathcal{P}\text{a}(w)}\right)$ and $\boldsymbol{\zeta}_{1:S}^{(w)}$ are MC samples from a standard Normal. Astudillo et al. [2] define the Expected Improvement (EI) of GPN, named as EI-FN using sample average approximation (SAA) as

$$\widehat{\text{EI-FN}}(\mathbf{x}_b, \boldsymbol{\zeta}_{1:S}) = \frac{1}{S}\sum_{s=1}^{S} \max\left(0, \tilde{f}^{(W)}(\mathbf{x}_b, \boldsymbol{\zeta}_{1:S}) - y_\star^{(W)}\right), \tag{10}$$

where $\mathbf{x}_b = (\mathbf{x}_b^{(w)})_{w\in\mathcal{W}}$ is the concatenated input vector and $\boldsymbol{\zeta}_{1:S}$ are the combined MC samples for all subprocesses to calculate the $\tilde{f}^{(W)}$ recursively using (9). The value of other acquisition functions like LogEI can be calculated similar to (10). While a novel idea to use the process network structure, GPN by [2, 19, 12, 4] still has certain limitations:

1. GPN posterior in (9) uses deterministic inputs for training ($\mathbf{a}^{(w)}$) and MC samples for prediction ($\mathbf{j}^{(w)}$). This is only valid for noise-free outputs $y^{(w)}$, an assumption that is restrictive for real-world processes as explained in Section 2. This makes the child node GPs unaware of the stochasticity in parent node GPs during training.

2. The GPN of the process network in Figure 1 has been shown in Figure 2a. GPN assumes that the noisy observations $\mathbf{y}^{\mathcal{P}\mathrm{a}(w)}$ from parent nodes serve as inputs to child nodes, while in fact it is the true or latent outputs $\tilde{\mathbf{f}}^{\mathcal{P}\mathrm{a}(w)}$ from parent nodes that serve as inputs to a subprocess $P^{(w)}$.
3. The use of a closed-form MLL for inference restricts the model to Gaussian likelihoods and does not scale well to large datasets.

### A.6 Partially Observable Gaussian Process Network (POGPN)

We discuss the main contribution of our paper and the derivation of Evidence Lower Bound (ELBO) and Predictive Log Likelihood (PLL) for POGPN using doubly stochastic variational inference (DSVI), inspired by [18]. Instead of node observations $\{\mathbf{y}^{(w)}\}_{n=1}^{N^{(w)}}$ sharing a common distribution space, we propose that the latent functions $\{\mathbf{f}^{(w)}\}_{n=1}^{N^{(w)}}$ reside in the same space and influence the child subprocesses. POGPN models the process network $\mathbf{P}$ as a topologically ordered DAG where each node is a vector-valued GP, $\tilde{\mathbf{f}}_n^{(\mathbf{w})} \sim \mathcal{GP}^{(w)}(\cdot, \cdot')$. Each node may have an arbitrary likelihood of observation, $p(\mathbf{y}_n^{(w)}|\mathbf{f}_n^{(w)})$, allowing for different dimensionalities and data types (continuous or categorical) for node observations $\mathbf{y}^{(w)}$. This formulation makes a Deep Gaussian Process (DGP) a special case of POGPN where only the final node is observed.

Unlike GPNs, a POGPN subprocess $P^{(w)}$ takes the parent's latent outputs $\mathbf{f}^{\mathcal{P}\mathrm{a}(w)}$ as input, not the noisy observations $\mathbf{y}^{\mathcal{P}\mathrm{a}(w)}$. By expressing the latent function as a distribution, we can marginalize over the parent node's output, providing robustness to its stochasticity and decoupling the observation from the underlying process. Figure 2b shows the POGPN representation of the process network from Figure 1.

**Evidence Lower Bound (ELBO).** Similar to the DGP in Section A.2, we introduce inducing points, $\mathbf{Z}^{\mathcal{P}\mathrm{a}(w)}$ and $\mathbf{Z}^{\mathcal{X}^{(w)}}$, for parent nodes and node inputs, respectively, such that $\mathbf{Z}^{(w)} = (\mathbf{Z}^{\mathcal{P}\mathrm{a}(w)}, \mathbf{Z}^{\mathcal{X}^{(w)}})$. We approximate the true posterior with a variational posterior, which can be expressed using (7) as

$$q\big(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}}\big) = \prod_{w\in\mathcal{W}} q\big(\mathbf{f}_n^{(w)}; \{\mathbf{f}_n^{\mathcal{P}\mathrm{a}(w)}, \mathbf{x}_n^{(w)}\}, \mathbf{Z}^{(w)}\big). \tag{11}$$

The Kullback-Leibler (KL) divergence between the variational and true posteriors is

$$\begin{aligned}&-\mathrm{KL}\big(q(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}})\|p(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}}|\{\mathbf{y}_n^{(w')}, \mathbf{x}_n^{(w')}\}_{w'\in\mathcal{W}})\big)\\ &= \underbrace{\mathbb{E}_{q(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}})}\big[\log p(\{\mathbf{y}_n^{(w)}\}_{w\in\mathcal{W}}|\{\mathbf{f}_n^{(w')}, \mathbf{x}_n^{(w')}\}_{w'\in\mathcal{W}})\big]}_{\text{Log Likelihood Loss (LL loss)}}\\ &\quad \underbrace{-\mathrm{KL}\big(q(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}})\|p(\{\mathbf{f}_n^{(w')}\}_{w'\in\mathcal{W}})\big)}_{\text{KL loss}} + \text{Evidence}\end{aligned} \tag{12}$$

The POGPN-ELBO is the sum of the 'LL loss' and 'KL loss' terms from (12). We now simplify these terms to enable inference. Using the DAG structure and conditional independence of observations, the 'LL loss' term simplifies as

$$\begin{aligned}&p\left(\{\mathbf{y}_n^{(w)}\}_{w\in\mathcal{W}}|\{\mathbf{f}_n^{(w')}\}_{w'\in\mathcal{W}}\right) = p\left(\mathbf{y}_n^{(W)}|\{\mathbf{f}_n^{(w')}\}_{w'\in\mathcal{W}}\right) p\left(\mathbf{y}_n^{(W-1)}, \{\mathbf{f}_n^{(w')}\}_{w'\in\mathcal{W}}\right)\\ &= \prod_{w\in\mathcal{W}} p(\mathbf{y}_n^{(w)}|\{\mathbf{f}_n^{(w')}\}_{w'\in\mathcal{W}}) = \prod_{w\in\mathcal{W}} p(\mathbf{y}_n^{(w)}|\mathbf{f}_n^{(w)})\end{aligned} \tag{13}$$

where $W = |\mathcal{W}|$. Using (11) and (13), the 'LL loss' term from (12) becomes

$$\begin{aligned}&\mathbb{E}_{q(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}})}\big[\log p\left(\{\mathbf{y}_n^{(w)}\}_{w\in\mathcal{W}}|\{\mathbf{f}_n^{(w)}, \mathbf{x}_n^{(w)}\}_{w\in\mathcal{W}}\right)\big]\\ =&\mathbb{E}_{q(\{\mathbf{f}_n^{(w)}\}_{w\in\mathcal{W}})}\big[\sum_{w\in\mathcal{W}} \log p\left(\mathbf{y}_n^{(w)}|\mathbf{f}_n^{(w)}\right)\big] = \sum_{w\in\mathcal{W}} \mathbb{E}_{q(\mathbf{f}_n^{(w)})}\big[\log p\left(\mathbf{y}_n^{(w)}|\mathbf{f}_n^{(w)}\right)\big],\end{aligned} \tag{14}$$

where the marginal $q(\mathbf{f}_n^{(w)})$ is calculated using (3) and (4) as

$$q(\mathbf{f}_n^{(w)}) = \int \prod_{t\in\mathcal{W}}^{w} q\left(\mathbf{f}_n^{(t)}|\{\mathbf{f}_n^{\mathcal{P}\mathrm{a}(t)}, \mathbf{x}_n^{(t)}\}, \mathbf{Z}^{(t)}\right) \mathrm{d}\mathbf{f}_n^{\mathcal{P}\mathrm{a}(t)}. \tag{15}$$

For $N^{(w)}$ observations per node, inference is performed by minimizing the negative POGPN-ELBO defined as

$$\mathcal{L}^{(\mathcal{W})}_{\substack{\text{POGPN}\\\text{ELBO}}} = \sum_{w\in\mathcal{W}} \mathcal{L}^{(w)}_{\substack{\text{Node}\\\text{ELBO}}} = \underbrace{-\sum_{w\in\mathcal{W}} \frac{1}{c^{(w)}} \sum_{n}^{N^{(w)}} \mathbb{E}_{q(\mathbf{f}_n^{(w)})}\left[\log p(\mathbf{y}_n^{(w)}|\mathbf{f}_n^{(w)})\right]}_{\text{LL loss}}$$
$$+ \underbrace{\beta \sum_{w\in\mathcal{W}} \mathrm{KL}\big(q(\mathbf{u}^{(w)})\|p(\mathbf{u}^{(w)})\big)}_{\text{KL loss}}. \tag{16}$$

A normalization constant $c^{(w)}$ makes the 'LL loss' comparable across nodes with different output dimensionalities. We set $c^{(w)} = D_{y^{(w)}}$ to give each node equal importance, though it can also be used for importance-based training. Unobserved nodes contribute only to the 'KL loss' term, similar to DGP, where only the final node is observed, and the other nodes contribute only to the 'KL loss' term.

**Predictive Log Likelihood (PLL)** Inspired by (6) we define the POGPN-PLL as

$$\mathrm{LL}_{\mathrm{PLL}} = \underbrace{-\sum_{w\in\mathcal{W}} \frac{1}{c^{(w)}} \sum_{n=1}^{N^{(w)}} \log \mathbb{E}_{q(\mathbf{f}_n^{(w)})}\left[p(\mathbf{y}_n^{(w)}|\mathbf{f}_n^{(w)})\right]}_{\text{LL loss}}, \tag{17}$$

while the 'KL loss' term remains the same as in (16). Like DGP-DSVI [18], we calculate the 'LL loss' and posterior for new point $\mathbf{x}_b = (\mathbf{x}_b^{(w)})_{w\in\mathcal{W}}$ using $S$ samples as

$$q(\mathbf{f}_b^{(w)}) \approx \frac{1}{S}\sum_{s=1}^{S} \prod_{t\in\mathcal{W}}^{w} q\left(\mathbf{f}_b^{(t)}|\{\tilde{\mathbf{f}}_{b,s}^{\mathcal{P}\mathrm{a}(t)}, \mathbf{x}_b^{(t)}\}, \mathbf{Z}^{(t)}\right). \tag{18}$$

The value of a given acquisition function is calculated similarly to (10). Since POGPN uses variational inference, it is a versatile framework that can be used for non-Gaussian likelihoods as well, which is a novel idea and a significant advantage over other Gaussian process frameworks.

## A.7 Process Networks for Experiments

### A.7.1 Penicillin production process

The penicillin production model replicates a fed-batch fermentation process, a well-known benchmark in chemical engineering. This process involves a complex interaction between microbial growth, substrate consumption, and product formation within a bioreactor under dynamic conditions. The process is defined by seven inputs ($D_x = 7$), which specify both the initial culture conditions and the operating parameters for the fermentation, each with specific bounds. The initial conditions include the culture volume ($V_0 \in [60, 120]$ L), biomass concentration ($X_0 \in [0.05, 18]$ g/L), and substrate concentration ($S_0 \in [0.05, 18]$ g/L). The key operating parameters are the bioreactor temperature ($T \in [293, 303]$ K), substrate feed rate ($F \in [0.01, 0.50]$ L/hr), substrate concentration in the feed ($s_f \in [500, 700]$ g/L), and the process pH ($\in [5.0, 6.5]$). The primary objective is to find the optimal set of inputs to maximize the final penicillin concentration ($P_{\text{final}}$).

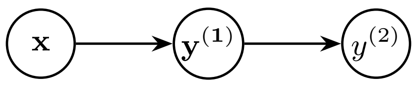


Figure 6: Process network for the Penicillin production process.

The state of the fermentation process is described by a five-dimensional vector, $\mathbf{y} = [P, X, S, V, \mathrm{CO}_2]$, representing the concentrations of penicillin, biomass, substrate, culture volume and off-gas carbon dioxide respectively. The evolution of this state vector is governed by a system of stochastic differential equations (SDEs), which are integrated using the Euler-Maruyama method. The simulation for each batch trajectory terminates when the culture volume exceeds its maximum ($V > 180$ L), the substrate is depleted ($S < 0$), or when penicillin production plateaus ($dP/dt < 10^{-11}$).

For our Bayesian optimization framework, we model this process as a three-node network: the inputs $\mathbf{x} = (V_0, X_0, S_0, T, F, s_f, \mathrm{pH})$ form the root node. The intermediate node, $\mathbf{y}^{(1)} =$

$(X_{\text{final}}, V_{\text{final}}, \text{CO2}_{\text{final}}, \text{time}_{\text{total}})$, captures the final states and total time from the simulation and is modeled with a multi-task Gaussian Process with multi-task Gaussian likelihood. The final objective node is the resulting penicillin concentration, $\mathbf{y}^{(2)} = P_{\text{final}}$ modeled as a single-task Gaussian Process with Gaussian likelihood. Bayesian optimization is then used to find the optimal set of inputs to maximize the final penicillin concentration.

The process dynamics are governed by the following system of SDEs:

$$\begin{aligned}
\frac{dV}{dt} &= F - F_{\text{loss}} + \varepsilon^{(V)} \\
\frac{dX}{dt} &= \mu X - \frac{X}{V}\frac{dV}{dt} + \varepsilon^{(X)} \\
\frac{dS}{dt} &= -\frac{\mu}{Y_{xs}}X - \frac{\mu_{pp}}{Y_{ps}}X - m_X X + \frac{F s_f}{V} - \frac{S}{V}\frac{dV}{dt} + \varepsilon^{(S)} \\
\frac{dP}{dt} &= \mu_{pp}X - KP - \frac{P}{V}\frac{dV}{dt} + \varepsilon^{(P)} \\
\frac{d\text{CO}_2}{dt} &= \alpha_1 \frac{dX}{dt} + \alpha_2 X + \alpha_3 + \varepsilon^{(\text{CO}_2)}
\end{aligned}$$

where the specific growth rate of biomass $\mu$ and the specific rate of penicillin production $\mu_{pp}$ are given by:

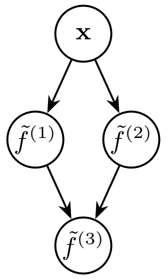
$$\begin{aligned}
\mu &= \frac{\mu_X}{1.0 + K_1/H + H/K_2}\frac{S}{K_X X + S}(k_g e^{-E_g/(RT)} - k_d e^{-E_d/(RT)}) \\
\mu_{pp} &= \mu_p \frac{S}{K_p + S + S^2/K_I}
\end{aligned}$$

and the evaporation loss $F_{\text{loss}}$ is modeled as:

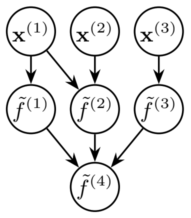
$$F_{\text{loss}} = \lambda V \left(\exp\left(5.0\frac{T - T_o}{T_v - T_o}\right) - 1.0\right)$$

The term $H$ is related to the agitation power $H_a$ by $H = 10^{-H_a}$. The model includes various other parameters representing physical and chemical properties of the process, such as yield coefficients $(Y_{xs}, Y_{ps})$, maintenance coefficients $(m_X)$, and kinetic constants.

### A.7.2 Synthetic function benchmarks



(a) Ackley, Rosenbrock, and Griewank benchmark functions



(b) Levy benchmark function

Figure 7: Process networks for the Ackley, Rosenbrock, Griewank, and Levy benchmark functions.

Table 1: Benchmark function details. The objective for Ackley, Griewank, and Rosenbrock is $y^{(3)}$, and for Levy it is $y^{(4)}$.

| **Benchmark** | **Node Equations**: $y^{(w)} = \tilde{f}^{(w)}(\cdot) + \tilde{\varepsilon}^{(w)}_{\text{obs}}$ |
|---|---|
| **Ackley** | $\tilde{f}^{(1)}(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{D_x}\sum_{d=1}^{D_x} x_{(d)}^2}} + \tilde{\varepsilon}^{(1)}$ |
| $\mathbf{x} \in [-32, 32]^{D_x}$ | $\tilde{f}^{(2)}(\mathbf{x}) = -e^{\frac{1}{D_x}} \sum_{d=1}^{D_x} \cos(2\pi x_{(d)}) + \tilde{\varepsilon}^{(2)}$ |
| | $\tilde{f}^{(3)}(\tilde{f}^{(1)}, \tilde{f}^{(2)}) = \tilde{f}^{(1)} + \tilde{f}^{(2)} + 20 + e + \tilde{\varepsilon}^{(3)}$ |
| **Griewank** | $\tilde{f}^{(1)}(\mathbf{x}) = \sum_{d=1}^{D_x} \frac{x_{(d)}^2}{4000} + \tilde{\varepsilon}^{(1)}$ |
| $\mathbf{x} \in [-600, 600]^{D_x}$ | $\tilde{f}^{(2)}(\mathbf{x}) = \prod_{d=1}^{D_x} \cos(\frac{x_{(d)}}{\sqrt{d}}) + \tilde{\varepsilon}^{(2)}$ |
| | $\tilde{f}^{(3)}(\tilde{f}^{(1)}, \tilde{f}^{(2)}) = 1 + \tilde{f}^{(1)} - \tilde{f}^{(2)} + \tilde{\varepsilon}^{(3)}$ |
| **Rosenbrock** | $\tilde{f}^{(1)}(\mathbf{x}) = \sum_{d=1}^{D_x} (1 - x_{(d)})^2 + \tilde{\varepsilon}^{(1)}$ |
| $\mathbf{x} \in [-5, 10]^{D_x}$ | $\tilde{f}^{(2)}(\mathbf{x}) = \sum_{d=1}^{D_x-1} 100(x_{d+1} - x_{(d)}^2)^2 + \tilde{\varepsilon}^{(2)}$ |
| | $\tilde{f}^{(3)}(\tilde{f}^{(1)}, \tilde{f}^{(2)}) = \tilde{f}^{(1)} + \tilde{f}^{(2)} + \tilde{\varepsilon}^{(3)}$ |
| **Levy** | $\tilde{f}^{(1)}(\mathbf{x}^{(1)}) = \sin^2(\pi w_1) + \tilde{\varepsilon}^{(1)}$ |
| $\mathbf{x} \in [-10, 10]^{D_x}$; $\mathbf{x}^{(1)} = x_{(1)}$; $\mathbf{x}^{(2)} = x_{(2:D_x-1)}$; | $\tilde{f}^{(2)}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sum_{d=1}^{D_x-1} (w_d - 1)^2 \big(1 + 10\sin^2(\pi w_d + 1)\big)$ |
| $\mathbf{x}^{(3)} = x_{(D_x)}$; | $\tilde{f}^{(3)}(\mathbf{x}^{(3)}) = (w_{D_x} - 1)^2(1 + \sin^2(2\pi w_{D_x})) + \tilde{\varepsilon}^{(3)}$ |
| $w_d = 1 + \frac{x_{(d)}-1}{4}$ | $\tilde{f}^{(4)}(\tilde{f}^{(1)}, \tilde{f}^{(2)}, \tilde{f}^{(3)}) = \sum_{i=1}^{3} \tilde{f}^{(i)} + \tilde{\varepsilon}^{(4)}$ |