Hybrid Physics-Informed Neural Networks for Automated Kinetic Parameter Calibration in Bio Manufacturing: A Human-in-the-Loop Case Study

Manav Madan[1], Andre Moser[2], V. C. Hass[2], Christoph Reich[1]

[1] IDACUS, Furtwangen University
{manav.madan,christoph.reich}@hs-furtwangen.de
[2] Institute for Applied Biology, Furtwangen University
{andre.moser, volkerclaus.hass}@hs-furtwangen.de

Abstract. Simulations of bio manufacturing processes such as microbial fermentation are governed by kinetic parameters. For example, the maximum rate at which a microorganism consumes sugar must be re-calibrated for every substrate composition before a mechanistic process model can predict how a culture will behave. Today this calibration is often performed manually by domain experts and may take hours. This paper presents a human-in-the-loop hybrid physics-informed neural network (PINN) that couples a fixed mechanistic bioprocess model with a lightweight network mapping sensor trajectories to five kinetic parameters. A central finding is that the data scientist alone, despite access to state-of-the-art optimisation tools, is systematically blocked at every critical juncture by incorrect domain assumptions i.e., initial conditions, feed concentrations, time-unit conventions, and substrate assignments. These remain invisible to the training algorithm. Two training approaches are compared: a piecewise model that resets at each measurement and a continuous model trained on the full trajectory. The piecewise model achieves low training error but performs worse than the original mechanistic model. The continuous model performs better, improving the baseline by up to 31.4% on unseen experiments. Final results from the expert interactions lead into a dual-role checklist for data scientists and domain experts undertaking hybrid PINN development on physical systems.

Keywords: hybrid physics-informed neural networks; bioprocess engineering; kinetic parameter estimation; human-in-the-loop machine learning; SPSA optimisation.

1 Introduction

Kinetic parameter estimation in bioprocess engineering is a labour-intensive and computationally expensive bottleneck. Fitting parameters such as maximum substrate consumption rates ($r_{SC,max}$), biomass yield coefficients ($Y_{X/SC}$), oxygen transfer coefficients (k_La), and product formation rates to mechanistic ordinary differential equation (ODE) models currently requires domain experts to iteratively adjust parameters by hand, either manually or with classical optimisation algorithms such as Nelder–Mead; for a single experiment this typically takes minutes to about an hour, but coordinating parameter estimates across multiple experiments simultaneously is considerably harder, since domain oversight is easily lost [3]. As industrial bioprocess development scales to larger experimental libraries, this manual step becomes rate-limiting for the whole pipeline. Physics-informed neural networks (PINNs) [1] offer an innovative principled path to automating this task. In a *hybrid* PINN [2], the neural network does not replace the mechanistic model; instead, it outputs a small set of parameters that feed into a pre-existing, unmodified mechanistic model. This is attractive for industrial settings because the mechanistic model encodes decades of biological and process knowledge that would be impractical to re-derive from data alone. This architecture has been applied to bioprocess

hybrid modelling [6], fermentation optimisation [7], and quality-by-design frameworks [8]. However, the existing literature has not systematically documented the structural dependency on domain knowledge that makes hybrid PINN deployment feasible in the first place. Mao et al. [9] found, through a qualitative study of 22 data-scientist–domain-expert teams in bio-medical science, that the most critical collaboration challenge is not technical but epistemic: the data scientist does not know what questions to ask, while the domain expert does not know what information the algorithm requires. Amershi et al. [10] similarly argued that the domain expert must be an active co-developer, not an occasional reviewer. This case study operationalises both insights in a concrete bioprocess setting in three ways: (i) a piecewise, reset-based training produces a deceptively low training error while failing once evaluated end-to-end, and that continuous, reset-free training resolves this; (ii) an eight distinct domain-knowledge gaps that blocked development and could not have been found by any algorithm operating on the sensor data alone; (iii) a dual-role checklist grounded is proposed. We are explicit from the outset about where the human enters this system: the trained network and mechanistic model contain no human-interaction component at runtime. The human-in-the-loop element is the development process and the iterative audit-and-correct cycle between data scientist and domain expert is what produces a usable model.

2 System Architecture

The hybrid PINN consists of three coupled components, summarised in Fig. 1. The mechanistic model (internally referred to as “Grundmodell”) is a six-state ODE system describing bioprocess dynamics for medium volume (V), carbon substrate (SC, glucose), nitrogen substrate (SN, yeast extract and peptone), biomass (X, yeast), carbon product (PC, ethanol), and solid product (PS, water), based on Monod kinetics [5]. It is used here as a black-box Python file with no exposed gradients; the neural network never modifies its equations. The KineticNN is a shallow fully-connected network (4→16→8→5, Tanh hidden activations, Sigmoid output) mapping a normalised, four-channel sensor vector — dissolved oxygen (DO), carbon dioxide (CO2), off-gas oxygen (O2), and pH — to five learnable kinetic parameters at each time point: the maximum specific glucose consumption rate (rSC,max), the biomass yield on glucose (YSC), the maximum specific product formation rate (rPC,max), the maximum specific nitrogen consumption rate (rSN,max), and the volumetric oxygen transfer coefficient (kLa). Because these values can differ slightly between nominally identical runs (e.g. inoculum variability, minor process deviations), they are more precisely experiment-specific effective kinetic parameters rather than universal physical constants. The network is initialised so that zero sensor input reproduces the mechanistic model's default parameters, and every output is bounded to a physiologically plausible range fixed a priori from literature-reported physiological ranges, independent of the target error i.e., so loosening the bounds is not an available lever for improving MAPE. The training infrastructure uses Simultaneous Perturbation Stochastic Approximation (SPSA) [4] with Adam momentum to estimate gradients through the black-box ODE, perturbing all network weights simultaneously with a random Rademacher vector $\Delta \in \{-1,+1\}^N$: $\hat{g} = (L(\theta+c\Delta) - L(\theta-c\Delta)) / (2c \cdot \Delta)$. This needs 6 ODE passes per epoch, versus N+1 = 174 for finite differences resulting into a 29× reduction. Total training time on a single GPU, across 30,000 epochs and six experiments, was 10–80 minutes, compared with an estimated 48 days for finite differences and with the 100+ hours per experiment typically required for manual expert fitting [3]. Hyperparameters are found via Optuna [11] with a median pruner on Stage-1 experiments only

(20 trials × 1,500 epochs). The final curriculum trains in three stages: batch glucose only (Stage 1), adding fed-batch glucose (Stage 2), then all six experiments (Stage 3).

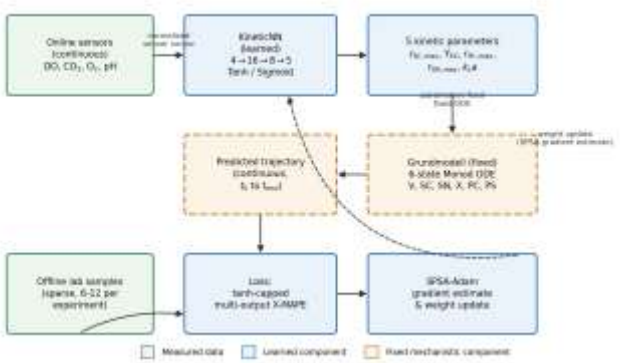


Fig. 1. System workflow. Two data sources feed the pipeline: continuous online sensors (green) drive the learned KineticNN (blue), and sparse offline lab samples (green) define the training loss. The five outputs of KineticNN are the only quantities the network learns; they are passed unmodified into the fixed, dashed-outline mechanistic-model (orange), whose predicted trajectory is compared against the lab samples. SPSA-Adam then estimates a gradient through the whole pipeline, including the black-box ODE, and updates only the KineticNN weights.

The training pool comprises six 1 L bioreactor experiments: two batch glucose runs (T1, SC_0=89 g/L; T2, SC_0=37.9 g/L), one fed-batch glucose run (T3, feed 500 g/L, SN_0=50 g/L), and three fed-batch soy-peptone runs — BioB (T4, T5) and Medo (T6), two industrial soy-peptone medium formulations — feed 216 g/L, SN_0=100 g/L, trust=0.10, Section 4.3). A seventh training-pool experiment (D1-5) was excluded on supervisor guidance. Two experiments from 2019 — V1 (10 L) and V2 (1 L), both collected by a different researcher — are held out entirely as unseen test data (Section 5.2); a third 2019 experiment (mDoE_HZDK_4) was excluded because a separate nitrogen pump introduces a dilution effect the current model cannot represent. We adopt a weighted deployment-mode error target of <15% X-MAPE (MAPE computed specifically on biomass X, the primary prediction target). This is consistent with reported relative errors of 6.9–12.8% for bioprocess soft sensors evaluated on held-out batches [14], and with the offline-assay sampling frequency of our data (6–12 lab measurements per 24-hour experiment), which itself bounds how tightly any model can be expected to track the true trajectory.

3 Piecewise vs. Continuous Training

The development happened in multiple rounds. Each time a new model was trained it was versioned numerically from v1 to v17. However, this paper present only two major developed solutions i.e., v10 and v17.

3.1 The Piecewise Paradigm (v10)

With only 6–12 offline lab measurements per 24-hour experiment, a single-trajectory ODE simulation from t=0 is numerically unstable: small early errors compound catastrophically before any measurement is encountered. The piecewise approach addresses this by resetting the ODE state to the measured values [X, SC, SN, PC] at every consecutive interval $[t_i, t_{i+1}]$, integrating forward, and comparing the prediction at t_{i+1} against the next measurement. The total loss is a weighted sum across all intervals and experiments, with weights derived from pre-training ODE diagnostics (a simulability factor) and domain confidence in feed concentrations (a trust factor).

The piecewise model (v10) reached a weighted training X-MAPE of 14.3%, meeting the <15% target. However, post-hoc analysis showed all five kinetic parameters had converged to their lower bounds. A degenerate solution in which the optimiser found a numerically convenient flat region that satisfied the sparse-data piecewise objective without encoding any genuine kinetic knowledge. This was confirmed by evaluating the same v10 weights in continuous mode (no resets, single simulation from t=0): the unmodified default parameters consistently matched or outperformed the v10 learned parameters (Table 3). The model had learned short-segment local interpolation, not global trajectory dynamics i.e., precisely the distinction von Stosch et al. [8] draw between models that fit data and models that encode knowledge.

3.2 The Continuous Paradigm (v17)

The continuous approach removes ODE resets entirely: the loss is computed on a single simulation from t_0 to t_{end}, aligning the training objective with the deployment scenario. A tanh (MAPE/CAP) transform is applied per output to prevent any single output. Particularly glucose (SC), which can start with a large initial error which results into overwhelming the gradient signal of the biomass term X. The continuous model (v17) reaches 2–5% X-MAPE on glucose training experiments, with learned parameters sitting at physically meaningful interior values (e.g. $r_{SC,max}$ at 22% of its allowed range, Y_{SC} at 22%, k_La at 17%, rather than pinned at a bound), and outperforms the mechanistic model defaults on both 2019 test experiments (Section 5.2). A reversal that piecewise training never achieved.

Table 1. Architecture and training-configuration comparison between the piecewise (v10) and continuous (v17) hybrid PINN. The two models share an identical network architecture; every difference in outcome stems from the training-loss paradigm. Corresponding MAPE figures are in Table 3.

Aspect	v10 — piecewise	v17 — continuous
ODE simulation mode	Reset to measured state at every interval $[t_i, t_{i+1}]$	Single simulation, t_0 to t_end, no resets
Training loss	Weighted sum of per-interval prediction errors	tanh-capped multi-output X-MAPE over the full trajectory
Alignment with deployment	Misaligned — never penalised for accumulated error	Aligned — training objective = deployment objective
KineticNN architecture	4→16→8→5, Tanh / Sigmoid	Same network; only loss and simulation mode differ
Optimiser	SPSA + Adam, 3 two-sided estimates / epoch	Same estimator, same curriculum
Learned parameters at convergence	All 5 at lower bound (degenerate)	Interior values (e.g. $r_sC_{,max}$ ≈ 22% of range)

4 The Expert-in-the-Loop: Eight Blocking Gaps

Before presenting the individual gaps, we clarify the role of human interaction. The architecture in Fig. 1 requires no human input during inference. The interaction happens one level up, in the model-development loop. Each of the developed versions between v1 to v17 followed the same six-step cycle: the data scientist (DS) trains a version and produces a diagnostic report (learned parameter values, per-experiment MAPE, physical-plausibility flags); the domain expert (DE) audits that report against process knowledge that is not present in any training file; a gap is identified and classified; DS and DE jointly agree a correction; and the DS retrains the next version. Every one of the eight gaps below was found through this loop, not by any automated

diagnostic. We group the eight gaps into three classes, following the taxonomy Amershi et al. [10] propose for interactive machine learning roles: data-value corrections (numeric initial conditions and concentrations), unit and convention fixes (time units, rate units, signal scales), and model-scope gaps (which experiments the current model can simulate at all, and why). Table 2 summarises all eight; Gap 5, the feed-rate unit conversion, was the single most impactful correction in the project.

Table 2. The eight domain-knowledge gaps that blocked development between v10 and v17, found exclusively through the human-in-the-loop cycle in Fig. 2, never by an automated diagnostic.

Gap	Category	Issue → Fix
1. Initial N (SN_0)	Data value	Assumed 8 g/L (default) → experiment-specific 30, 50, or 100 g/L, from cultivation-plan parsing; wrong SN_0 distorts nitrogen-limitation throughout.
2. Feed glucose (SC_F,Vin)	Data value	Assumed 300 g/L (fallback) → 500 g/L (T3), 216 g/L (BioB/Medo); T3 stuck near 36% MAPE for five versions until corrected.
3. Nitrogen feed (SN_F,Vin)	Data value	Training experiments have zero N in feed; 2019 tests exhibit a continuous N feed → approximated as increased initial SN_0 for V1/V2.
4. Control-file time axis	Unit/convention	Feed profiles store time in seconds; ODE integrates in hours → uncorrected, feed pump never advanced past t=0 in any fed-batch run.
5. Feed rate units	Unit/convention	Feed rates in m^3/s vs. m^3/h expected (3,600× error) → corrected fed-batch ΔX from +1.1 to +10.6 g/L — the single most impactful fix.
6. CO_2 sensor scale	Unit/convention	Different instruments report CO2 off-gas on different scales (0–100% or 0–5% v/v), so an identical raw reading could mean two very different concentrations depending on the sensor — without correction this fed inconsistent CO2 signals into the network across experiments → median-based rule (÷100 if median > 50), verified against instrumentation.
7. Soy peptone as sole substrate	Model scope	T4/T5/T6 have near-zero glucose ($SC_0 \approx 0$) yet continue to grow slowly — likely via direct nitrogen-source (soy peptone) catabolism, a pathway absent from the mechanistic model, which drives growth only through glucose → trust down-weighted to 0.10; a nitrogen-linked growth term is a possible model extension.
8. Cultivation plan as latent data	Model scope	Accurate feed/IC values existed in Cultivation_Plan.csv, unflagged and unreadable by a standard parser → custom parser; knowledge lived in informal artefacts, not a database [9].

5 Experimental Results

5.1 Training Experiments

Six experiments from a 1 L bioreactor were used (Section 2): two batch glucose (T1, T2), one fed-batch glucose (T3), and three fed-batch soy-peptone experiments (BioB T4, T5; Medo T6). A 70/30 temporal holdout was applied: the last 30% of time points per experiment were withheld from training and hyperparameter optimisation as a within-experiment test set. Because the mechanistic core is never modified, every improvement over the Default column in Table 3 is attributable solely to the learned kinetic parameters i.e., to the value the neural network adds on top of the physics model alone.

Table 3. Deployment-mode (continuous simulation) X-MAPE per training experiment, baseline (Default) shown first. "v10 deploy" re-evaluates the piecewise-trained weights with no resets, exposing the degenerate solution. * Soy peptone experiments, trust=0.10 (Gap 7); T6 remains above target due to the unresolved growth-pathway gap.

Experiment	Type	Default (physics only)	v10 train (piecewise)	v10 deploy (continuous)	v17 deploy (continuous)
T1	Batch	72.3%	17.9%	~70%	3.8%
T2	Batch	67.8%	5.6%	~68%	3.0%
T3	Fed-batch	–	18.4%	~65%	2.5%
T4 (BioB)	Fed-batch*	–	42.1%	–	3.8%
T5 (BioB)	Fed-batch*	–	14.5%	–	3.9%
T6 (Medo)	Fed-batch*	–	8.3%	–	25.1%

5.2 Test Experiments (Unseen)

Two experiments from 2019 (190312_V1, 10 L; 190606_A1-1_V2, 1 L) were used as unseen test data, with continuous nitrogen feed handled via the Gap 3 approximation (SN_F,Vin=0, SN_0 increased to 20 g/L for V1, 30 g/L for V2). A third 2019 experiment (mDoE_HZDK_4) was excluded because a separate nitrogen pump introduces a dilution effect the current model cannot represent; incorporating a dilution-adjusted glucose-feed calculation is planned as follow-up work.

Results confirm the central hypothesis: the v17 continuous model outperforms the default parameters on both unseen experiments (V1: 63.6% vs. 72.3%, +8.7%; V2: 36.4% vs. 67.8%, +31.4%), while the v10 piecewise model, evaluated the same way, performs at or below the default-parameter baseline. The smaller V1 improvement reflects the reactor-scale mismatch (10 L vs. 1 L training data); the larger V2 improvement, from a 1 L experiment in the same reactor class, is more directly interpretable as genuine within-distribution generalisation. Fig. 2 and Fig. 3 visualise both models on V2.

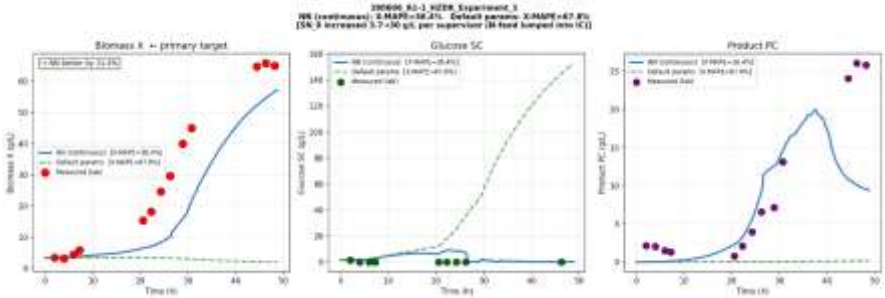


Fig. 2. Continuous model (v17) on the unseen V2 test set (logged under its original file name, HZDK_Experiment_1, in the source plot below). Left: biomass X (primary target) — the learned model (blue) tracks the measured points (red) far more closely than the default-parameter trajectory (dashed green), an X-MAPE improvement of 31.4% over the unmodified default parameters (67.8% X-MAPE). Centre: glucose SC — the default trajectory over-predicts residual glucose by roughly 10× at 50 h. Right: product PC — the learned model captures the rise and subsequent decline in product concentration that the flat default trajectory misses entirely.

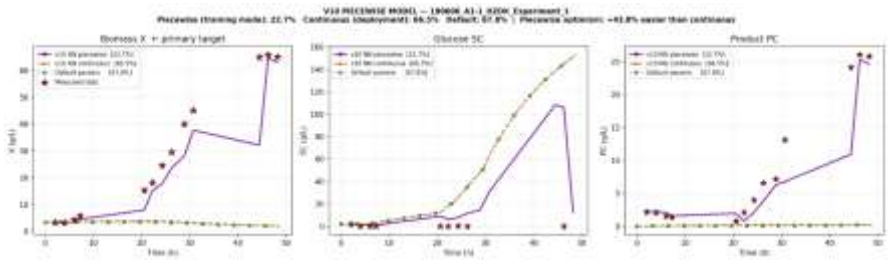


Fig. 3. The same v10 weights on V2, contrasting piecewise training-mode evaluation (purple, resets at every measurement, 22.7% MAPE) against continuous deployment-mode evaluation of the identical weights (orange, no resets, 66.5% MAPE) — close to the 72.3%–67.8% default-parameter baseline (green) shown in Fig. 3. The gap between the purple and orange curves is the piecewise paradigm's core failure mode: a model that looks accurate under its own training loss but has not learned the trajectory dynamics.

Three limitations are stated explicitly rather than left implicit. First, the calibration-time comparison in is an approximate, literature-informed estimate rather than a controlled same-experiment measurement against our own six experiments; a head-to-head timing comparison remains future work. Second, because SPSA is a stochastic gradient estimator, the learned parameters carry run-to-run variability we have not yet quantified; multi-seed training to report a parameter distribution is planned follow-up work rather than included here. Third, our baseline throughout is the mechanistic model's unfitted default parameters, not a classically-fitted baseline (e.g. Nelder-Mead) estimated on the same training data.

6 Dual-Role Checklist for Hybrid PINN Development

No published checklist exists specifically for hybrid PINN deployment on physical systems with legacy ODE models. The closest analogues such as the DOME standard for ML metadata [12] and the collaborative-ML guidelines of Amershi et al. [13] do not address the specific interface challenges between a data-driven component and a black-box mechanistic model. Table 4 condenses the checklist derived from our case study, structured by the three gap categories of Section 4. We distinguish two roles: DS (data scientist) owns the ML pipeline; DE (domain expert) owns the mechanistic model and process knowledge; DS+DE items require joint discussion before training begins. Although derived from a single bioprocess case study, the underlying checks generalise to any project coupling a learned component with a fixed legacy simulator.

Table 4. Condensed dual-role checklist for hybrid PINN development on legacy mechanistic models, derived from the eight gaps in Section 4.

Phase	Owner	Key checks
1. Before any code is written	DS+DE	Map ODE states to measured vs. inferred variables; list every IC (initial condition) and feed parameter with units and source file; confirm the ODE's time-unit convention against every control file; identify the basic conditions (e.g. substrate composition) per experiment; run a simulability pre-flight ($\Delta X \geq 1$ g/L) once minimal ODE tooling exists; agree and document trust weights.
2. Sensor and signal specification	DS+DE	Confirm physical unit, range, and scale variants for every sensor; identify artefacts and the most informative signals; separate continuous (online) from sparse (offline) streams; freeze normalisation statistics for inference.

Phase	Owner	Key checks
3. Model interface audit	DS	Audit every input file format for units, encoding, delimiters; add a unit check at every data–ODE boundary (a common source of silent failures); print a post-load sanity check vs. planned IC; ensure *all* time-dependent terms use a consistent time unit.
4. Training configuration	DS+DE	Supply physically motivated parameter bounds with justification; inspect the learned parameter distribution after every run for boundary saturation; choose piecewise vs. continuous loss deliberately; scale multi-output loss (e.g. tanh-capping) so no output dominates.
5. Validation and deployment	DS+DE	Define the held-out set before training (temporal holdout plus ≥ 1 unseen experiment); run default-parameter and interpolation baselines in the same evaluation mode as training; before deployment, confirm matching carbon source, reactor scale (within 2×), sampling frequency and sensor suite; save full model metadata.

7 Discussion

Six of the eight gaps in Section 4 were data-value or unit/convention issues. No unsupervised algorithm applied to the sensor data alone could have resolved them, because the information simply does not exist in those signals. Gap 5 is instructive: the error was syntactically valid Python and raised no runtime exception. The only signal of failure was a domain-informed physical sanity check i.e., recognising that 1.5×10^{-3} g/L/h of substrate delivery is inconsistent with the observed growth trajectory. This aligns with the 'dual-goal dilemma' Mao et al. [9] identify: data scientists seek the right answer (minimum MAPE), while domain experts ask the right question (is this physically consistent?). A physically inconsistent model with low training MAPE, as the piecewise v10 model demonstrates, provides false confidence. The piecewise-versus-continuous finding has a broader implication for sparse-data scientific ML: piecewise training is reasonable when data is sparse and full-trajectory simulation is unstable, but using it as the loss function misaligns training and deployment, since the model is never penalised for errors that accumulate over longer horizons.

8 Conclusion

A hybrid PINN is presented here for automated kinetic parameter calibration in bio manufacturing and showed that the training-loss paradigm i.e., piecewise vs. continuous, determines whether learned parameters are physically meaningful or deceptively low in training MAPE but useless in deployment. The continuous model outperforms the default parameter baseline by up to 31.4% on completely unseen 2019 experiments. More broadly, this work documented eight domain-knowledge gaps that blocked development at different stages and synthesised them into a dual-role checklist spanning data-value corrections, unit-and-convention fixes, and model-scope gaps. The central message for practitioners is that domain expertise in hybrid PINN projects is not an optional enhancement but a structural prerequisite: a data scientist with state-of-the-art optimisation tools cannot compensate for incorrect initial conditions, wrong unit conventions, or an unmodelled growth pathway. Automating kinetic calibration is therefore not a problem of replacing the domain expert with a model, but of building a development process in which expert knowledge and machine optimisation correct each other, iteratively, until both agree the result is trustworthy.

References

1. Raissi, M., Perdikaris, P., & Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686–707.
2. Bangi, M. S. F., & Kwon, J. S. I. (2020). Deep hybrid modeling of chemical process: Application to hydraulic fracturing. Computers & Chemical Engineering, 134, 106696.
3. Mandenius, C.-F., & Brundin, A. (2008). Bioprocess optimization using design-of-experiments methodology. Biotechnology Progress, 24(6), 1191–1203.
4. Spall, J.C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control, 37(3), 332–341.
5. Monod J. The Growth of Bacterial Cultures. Annu. Rev. Microbiol. 1949;3(1):371–94. doi:10.1146/annurev.mi.03.100149.002103
6. Moser, A., Appl, C., Brüning, S., & Hass, V.C. (2020). Mechanistic Mathematical Models as a Basis for Digital Twins. Advances in Biochemical Engineering/Biotechnology.
7. Rogers, A.W., Song, Z., Ramon, F.V., Bhonsale, S., Van Impe, J. (2023). Investigating physics-informed neural networks for bioprocess hybrid model construction. In Proceedings of the 13th International Symposium on Process Systems Engineering. Elsevier.
8. von Stosch, M., Davy, S., Francois, K., et al. (2014). Hybrid modeling for quality by design and PAT — benefits and challenges of applications in biopharmaceutical industry. Biotechnology Journal, 9, 719–726.
9. Mao, Y., Wang, D., Muller, M., Varshney, K.R., Baldini, I., Dugan, C., & Mojsilović, A. (2019). How Data Scientists Work Together With Domain Experts in Scientific Collaborations: To Find The Right Answer Or To Ask The Right Question? Proceedings of the ACM on Human-Computer Interaction, 3(GROUP), 1–23.
10. Amershi, S., Cakmak, M., Knox, W.B., & Kulesza, T. (2014). Power to the People: The Role of Humans in Interactive Machine Learning. AI Magazine, 35(4), 105–120.
11. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2623–2631.
12. Walsh, I., Fishman, D., Garcia-Gasulla, D., et al. (2021). DOME: recommendations for supervised machine learning validation in biology. Nature Methods, 18(10), 1122–1127.
13. Amershi, S., Weld, D., Vorvoreanu, M., et al. (2019). Guidelines for Human-AI Interaction. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 1–13.
14. Siegl, M., Kämpf, M., Geier, D., Andreeßen, B., Max, S., Zavrel, M., & Becker, T. (2023). Generalizability of Soft Sensors for Bioprocesses through Similarity Analysis and Phase-Dependent Recalibration. Sensors, 23(4), 2178.