Explaining Process Control Optimisation Recommendations via GradientSHAP and Implicit Differentiation

Paul Darm[1], Cem Alpturk[3], Kenneth Ulrich[3], William Duncan[2], Ali Anwar[1] and Annalisa Riccardi[1]

[1]University of Strathclyde, Glasgow, United Kingdom
[2]Weir, Glasgow, United Kingdom
[3]Weir, Malmö, Sweden

Abstract

Automated optimisation is increasingly adopted in industrial processes, yet a trust gap persists between engineers who design these algorithms and operators who must act on their recommendations. Explainable AI methods like SHAP (SHapley Additive exPlanations) have transformed interpretability for machine learning predictions; optimisation outputs could benefit from similar techniques. We present an approach that integrates Implicit Function Theorem (IFT) based sensitivity analysis with SHAP attribution and narrative generation via Large Language Models (LLM), producing explanations tailored for operators. Our approach leverages IFT to compute exact parameter sensitivities $\partial p^*/\partial x$ from the optimality conditions, enabling efficient GradientSHAP computation. For an industrial High Pressure Grinding Roll (HPGR) control optimisation problem with 22 features, we achieve equivalent SHAP attributions (correlation >0.99 with KernelSHAP) with over $40\times$ speedup, enabling real-time natural language explanations. We validate on industrial scenarios and present feedback from domain experts on generated explanations.

Keywords

Explainable AI, Explainable Optimisation, Implicit Function Theorem, SHAP, Sensitivity Analysis, LLM, HPGR

1. Introduction

Modern industrial processes increasingly rely on mathematical optimisation for operational control, offering significant economic and efficiency benefits. However, seamless adoption faces a critical barrier: operators often distrust automated recommendations they cannot understand [1]. In high-stakes applications such as mineral processing, where safety and equipment constraints are paramount, explanations can ensure that optimized parameters lead to more efficient and safe operation by enabling operators to validate recommendations before implementation.Explainable Artificial Intelligence (XAI) has made significant progress in interpreting machine learning model predictions [2, 3]. We apply a similar philosophy to explain optimisation algorithm outputs, answering operator questions like: *"Why did the optimiser recommend higher pressure today?"* or *"What input changes are driving this speed recommendation?"* by providing attributions of input features in form of approximated SHAP values to an LLM. The LLM then generates a narrative based on the SHAP values to explain why the parameters settings have changed.

This paper makes the following contributions:

1. *GradientSHAP for optimisation:* We apply GradientSHAP to explain optimisation algorithm outputs efficiently by deriving the required gradients $\partial p^*/\partial x$ from the Implicit Function Theorem.
2. *SHAP-based explanations:* We use the resulting attributions to generate natural language explanations for optimised parameters for an HPGR process for operators, validating it through feedback from domain experts.

Late-breaking work, Demos and Doctoral Consortium, colocated with the 4th World Conference on eXplainable Artificial Intelligence: July 01–03, 2026, Fortaleza, Brazil

paul.darm@strath.ac.uk (P. Darm); cem.alpturk@mail.weir (C. Alpturk); kenneth.ulrich@mail.weir (K. Ulrich); william.duncan@mail.weir (W. Duncan); ali.anwar@strath.ac.uk (A. Anwar); annalisa.riccardi@strath.ac.uk (A. Riccardi)

2. Related Work

Explainability for Machine Learning. The challenge of interpreting black-box model predictions has driven extensive research in explainable AI. SHAP [2] provides game-theoretic feature attributions with properties like local accuracy and consistency. GradientSHAP [4] computes attributions by integrating gradients along paths from baseline to instance, offering efficiency for differentiable models. LIME [3] fits local surrogate models via perturbation sampling. Henkel et al. [5] applied SHAP to neural networks within model predictive control for model transparency. Martens et al. [6] demonstrated that LLMs can convert numerical attributions into human-readable narratives.

Explainability in Optimisation. Unlike XAI for ML, explainability in optimisation remains nascent [7]. Korikov et al. [8] explored counterfactual explanations via inverse optimisation, identifying minimal changes to make a desired solution optimal. Biemans et al. [9] presented a LIME-inspired sampling approach to generate feature importance attributions for supply chain scheduling, requiring N optimisation solves per explanation. An LLM was then used to generate narratives for specific supply chain scheduling decisions.

Differentiable Optimisation. A growing body of work treats optimisation problems as differentiable components within larger learning systems, computing gradients of optimal solutions via the implicit function theorem applied to optimality or KKT conditions [10, 11, 12]. These methods enable end-to-end learning of cost functions, constraints, or dynamics from data.

Our work sits at the intersection of these three lines: we apply implicit differentiation, the same technique used to embed optimisers as trainable layers, to instead compute exact sensitivities with a single solve, which we then use to generate SHAP-based narative explanations of optimiser recommendations.

3. Problem Formulation

3.1. HPGR Parameter Optimisation

High Pressure Grinding Rolls (HPGR) are critical comminution equipment in mineral processing [13], crushing ore between two counter-rotating rolls under high hydraulic pressure (Fig. 1). Operators must continuously adjust two control parameters, operating pressure (p_{pressure}) and peripheral roll speed (p_{speed}), to achieve production targets while respecting equipment constraints.

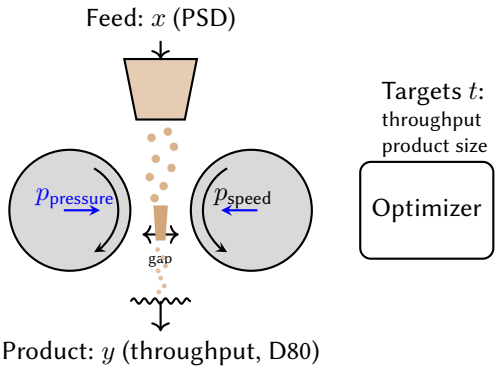

Figure 1: HPGR simplified schematic. Two counter-rotating rolls crush feed material under hydraulic pressure. The optimiser adjusts pressure and speed to minimise deviation from production targets given current feed properties.

The parameter optimisation problem is formulated as:

$$\min_{p \in \mathbb{R}^2} L(p, x, t) \tag{1}$$

where:

- $p = [p_{\text{pressure}}, p_{\text{speed}}]^T$ are control parameters
- $x \in \mathbb{R}^n$ is the process input particle size distribution (PSD), a 20-dimensional array of cumulative passing percentages at standard sieve sizes
- $t \in \mathbb{R}^m$ are production targets (throughput, product fineness)
- $L(p, x, t)$ is a loss function measuring deviation from targets

In practice, operating bounds constrain the feasible region, but for the scenarios in this paper, optimal solutions lie in the interior, allowing us to apply unconstrained sensitivity analysis. Extension to constraint boundaries is addressed in future work.

The loss function L uses a physics-based population balance model [13] with data-fitted breakage parameters, implemented in JAX and fully differentiable, that predicts throughput and product fineness (D80—the particle size below which 80% of the product passes). The loss measures weighted deviations from targets:

$$L(p, x, t) = w_1(\hat{y}_{\text{throughput}} - t_{\text{throughput}})^2 + w_2(\hat{y}_{\text{D80}} - t_{\text{D80}})^2$$

3.2. Explanation Task

Given two operating points:

- *Baseline:* inputs x_0, targets t_0, optimal parameters $p_0^* = \arg\min_p L(p, x_0, t_0)$
- *Instance:* inputs x_1, targets t_1, optimal parameters $p_1^* = \arg\min_p L(p, x_1, t_1)$

We seek to explain: *Why did optimal parameters change from p_0^* to p_1^*?*
Specifically, we want an attribution breakdown:

$$\Delta p^* = p_1^* - p_0^* = \sum_i \phi_i^{(x)} + \sum_j \phi_j^{(t)}$$

where $\phi_i^{(x)}$ represents the contribution of input x_i and $\phi_j^{(t)}$ represents the contribution of target t_j.

4. Methodology

4.1. IFT for Sensitivity Computation

Let $s = (x, t)$ denote the combined vector of process inputs and production targets. The optimal parameters $p^*(s)$ are implicitly defined by the optimality conditions. For unconstrained optimisation, first-order optimality requires:

$$\nabla_p L(p^*, s) = 0$$

By the Implicit Function Theorem, if the Hessian $H = \nabla_p^2 L(p^*, s)$ is non-singular, then $p^*(s)$ is locally differentiable:

$$\frac{\partial p^*}{\partial s} = -H^{-1} \cdot \frac{\partial^2 L}{\partial p \partial s}\bigg|_{(p^*, s)} \tag{2}$$

See Dontchev and Rockafellar [14] for the general theory of implicit differentiation in optimisation. The Hessian is computed once; scalar variables (moisture, targets) are batched together, while array-valued inputs (PSD) require a separate batch. Automatic differentiation via JAX [15] computes these efficiently, adding negligible overhead compared to the optimisation solve.

4.2. GradientSHAP for Optimisation

With sensitivities $\partial p^*/\partial s$ available via IFT, we apply GradientSHAP [2] to compute SHAP values. GradientSHAP integrates gradients along a path from a baseline to the instance of interest. For a differentiable function $h : \mathbb{R}^d \to \mathbb{R}$, the SHAP value for feature i is:

$$\phi_i = (s_i - s_i^{\text{baseline}}) \int_{\alpha=0}^{1} \left. \frac{\partial h}{\partial s_i} \right|_{s^{\text{baseline}} + \alpha(s - s^{\text{baseline}})} d\alpha \tag{3}$$

In our case, h is the optimal parameter function $p^*(s)$, mapping inputs and targets to optimal operating parameters. Unlike neural networks where gradients are readily available, computing $\partial p^*/\partial s$ for an optimiser's output is not directly possible. IFT enables this by differentiating through the optimality conditions (Eq. 2), which in turn enables GradientSHAP with theoretical efficiency gains over black-box methods such as KernelSHAP that require repeated optimisation solves.

4.3. Narrative Generation with LLMs

We use OpenAI (GPT-5) to convert SHAP attributions into operator-friendly narratives. The prompt provides domain knowledge about HPGR physics and the numerical SHAP breakdown; the LLM then interprets these values in operationally meaningful terms. Unlike surrogate-model approaches, GradientSHAP attributions derive directly from gradients of the actual optimiser, capturing exactly how the optimal output would change for small perturbations of each input, providing a principled foundation for faithful explanations. Crucially, the LLM acts as a translator of numerical attributions, not a reasoner about physics: the prompt grounds it in SHAP values computed from the actual optimiser, recommended parameter values come from the optimiser rather than the LLM, and operators validate all recommendations before implementation.

5. Results

5.1. Sample Efficiency and Computational Performance

We examine convergence behavior of each approach. Figure 2 shows KernelSHAP requires approximately 100 samples to reduce variance, while Figure 3 demonstrates GradientSHAP produces stable results with as few as 2–3 path integration samples. This difference in sample efficiency yields over 40× speedup (Table 1). GradientSHAP timing is relatively stable across sample counts because the expensive JAX compilation of the Hessian computation occurs once and is amortised across path points.

Table 1
Timing comparison between KernelSHAP and GradientSHAP. Speedup is relative to converged KernelSHAP (1000 samples).

Method	*Samples*	*Mean Time (s)*	*Speedup*
KernelSHAP	10	20.5	–
KernelSHAP	100	38.1	–
KernelSHAP	1000	362.9	1.0× (ref)
GradientSHAP	2	7.7	47.1×
GradientSHAP	3	8.0	45.3×
GradientSHAP	5	8.6	42.2×

5.2. GradientSHAP Validation

Having established that GradientSHAP converges with few samples, we now validate that it produces correct SHAP values by comparing against converged KernelSHAP [2] across 72 baseline-instance pairs.

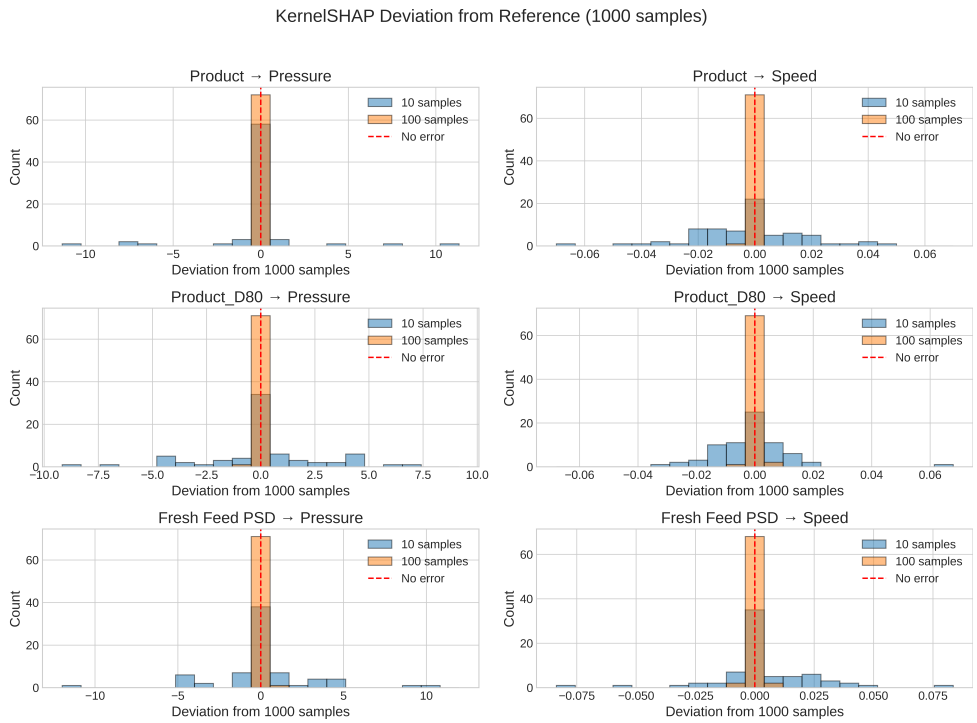

Figure 2: KernelSHAP convergence: deviation from 1000-sample reference. Blue: 10 samples (wide spread); Orange: 100 samples (concentrated near zero). KernelSHAP requires approximately 100 samples for stable results.

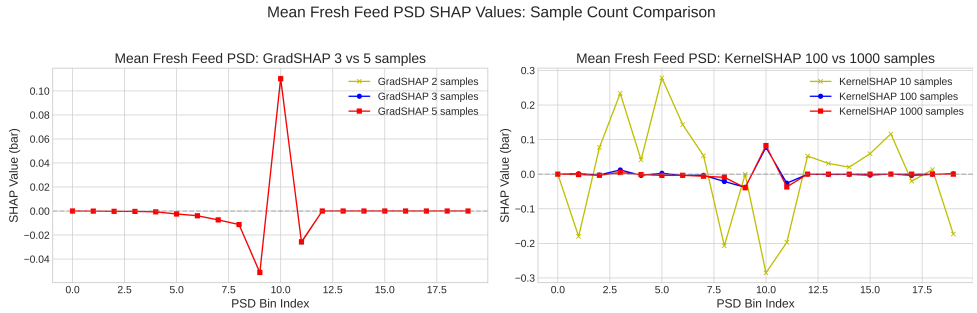

Figure 3: Sample count comparison for PSD SHAP values. Left: GradientSHAP shows rapid convergence with few samples. Right: KernelSHAP requires many more samples for stability.

Figure 4 shows correlation plots for scalar input features, demonstrating strong agreement ($r > 0.99$) between converged KernelSHAP and GradientSHAP with few path integration samples.

For array-valued inputs, Figure 5 compares element-wise SHAP values averaged over the dataset for the 20-dimensional particle size distribution (PSD). Both methods produce nearly identical attribution profiles, with characteristic peaks around bins 7–10 indicating that mid-range particle sizes have the strongest influence on optimal parameters. PSD changes primarily affect pressure recommendations (SHAP values ± 0.10 bar) with weaker effects on speed ($\pm 0.2\%$). The close agreement confirms that GradientSHAP correctly approximates KernelSHAP for both scalar and array-valued inputs.

6. Explanations

The SHAP attributions computed via IFT-based GradientSHAP were used to generate natural language explanations for multiple test scenarios. We gathered informal qualitative feedback from three domain experts (two process engineers and one HPGR specialist), who evaluated the explanations for clarity and practical utility. Overall, they found the generated explanations helpful and accurate in capturing why the optimiser recommended specific parameter changes; a formal user study is planned as future

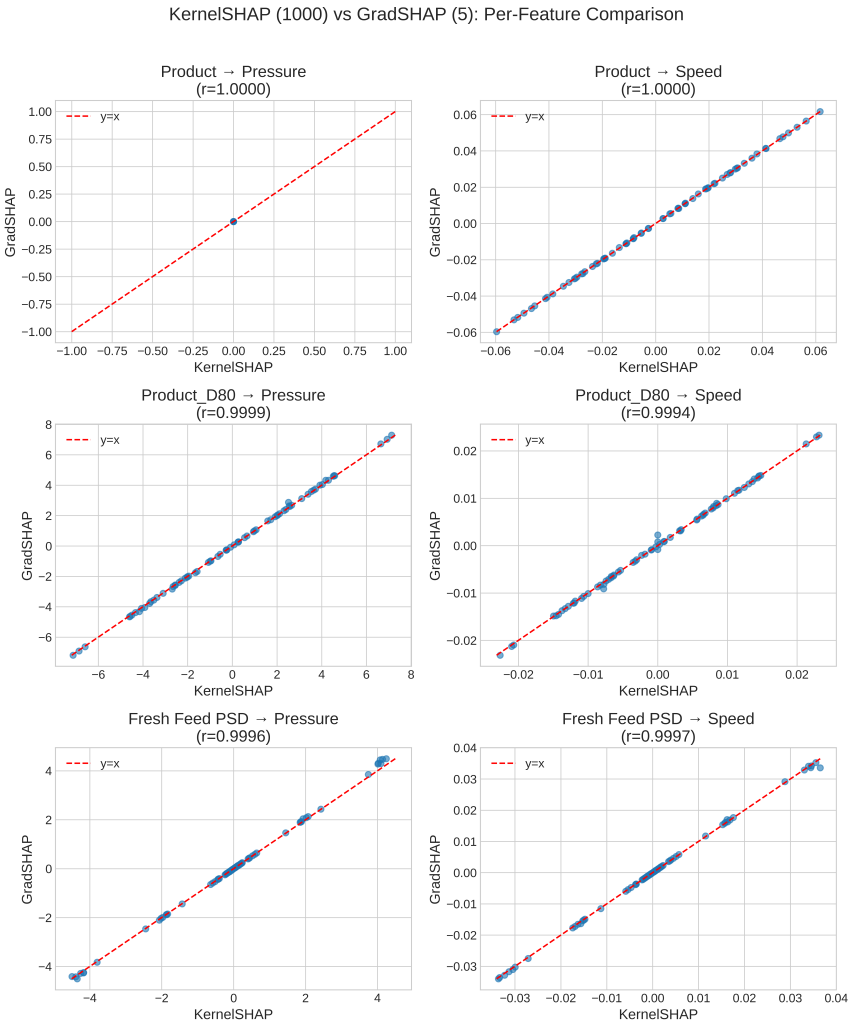

Figure 4: Correlation between converged KernelSHAP and GradientSHAP for scalar input features. Correlation exceeds 0.99 for all features except product target (pressure-independent), validating that GradientSHAP correctly approximates SHAP values.

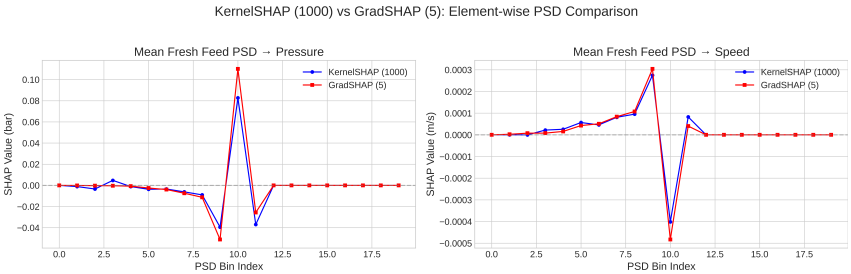

Figure 5: Mean element-wise SHAP values for fresh feed PSD, averaged over the dataset. Left: pressure output; Right: speed output. KernelSHAP (1000 samples, blue) and GradientSHAP (5 samples, red) show excellent agreement across all 20 PSD bins.

work.

However, the evaluation also revealed important design principles for effective explanations. First, experts consistently skipped detailed attribution breakdowns and sought the final recommendation immediately; explanations should therefore lead with a one-sentence actionable summary before presenting supporting details. In addition, returning raw SHAP values together with the explanation for transparency were unhelpful to practitioners unfamiliar with the method; as one reviewer noted, *'As someone with no background in SHAP, the section 'What These SHAP Values Mean isn't clear to me.".*

So raw SHAP values were omitted. Finally, technical precision of the algorithm output and thus the explanation is not useful for practitioners which preferred percentages over raw units ("73% of max speed"), rounded values ("165 bar" not "166.8 bar").

Based on this feedback, the explanation generation was adapted to incorporate these principles. The following scenario demonstrates an explanation that follows these principles.

6.1. Scenario: Reduced Throughput with Coarser Feed*

The following is a summarisation of one generated explanation by our pipeline:

Operating Point Change:

- Throughput target: 280 → 270 t/h (−10 t/h)
- Product fineness (D80) target: 1450 → 1460 μm (+10 μm, slightly coarser)
- Feed PSD: coarser across all size classes (1–2% fewer fines near screen cut)

Recommended Adjustments:

- Pressure: 125 → 122 bar (−3 bar); 75% → 70% of range
- Speed: 88% → 87% of max (−1%); 65% → 64% of range

Generated Explanation (condensed):

> *One-sentence summary:* Because the operator wants lower throughput and a slightly coarser product, the system recommends backing off both pressure and roll speed, even though the feed itself is coarser.
>
> *Pressure (−3 bar):* The operator allows the product to be about 10 μm coarser, so the HPGR does not need to crush as hard, this alone explains about 3.5 bar of pressure reduction. The coarser feed (fewer fines near the screen cut) partially offsets this, pushing pressure up by about 0.5 bar. Net result: pressure decreases by 3 bar.
>
> *Speed (−1%):* The operator explicitly wants less tonnage. To achieve that, the rolls should slow down slightly. The lower throughput request is the clear reason for reducing speed by about 1%.
>
> Both changes are small, stable, and well within operating limits.[1]

This explanation demonstrates: (1) leading with a one-sentence summary, (2) explicit baseline-to-new context, (3) SHAP values translated to operational meaning ("this alone explains about 3.5 bar"), and (4) operator-friendly units (percentages, rounded values, screen cut reference).

7. Conclusion

This paper presented the first application of GradientSHAP to explain optimisation recommendations, leveraging the Implicit Function Theorem for efficient sensitivity computation. By exploiting the mathematical structure of optimisation rather than treating it as a black box, we achieve over 40× speedup compared to KernelSHAP while maintaining correlation >0.99 in SHAP attributions. Integration with LLM-based narrative generation provides a complete pipeline from optimisation outputs to operator-friendly explanations, validated through feedback from process engineers and plant operators.

A limitation of the current approach is that it applies only to unconstrained optimisation, where solutions lie in the interior of the feasible region. Extension to constrained problems requires differentiating through the full Karush–Kuhn–Tucker (KKT) system to handle changes in the active set [14]. In addition, both the loss function and process model must be differentiable. SHAP values are also local attributions which means that the generated explanations are specific to a baseline-instance pair. While demonstrated on HPGR, the methodology applies broadly to differentiable optimisation in process control.Future work includes expanding the methodology to constrained optimisation and formal user studies of the generated explanations. Code is available on request due to industrial confidentiality.

[1] **Values modified for confidentiality; relative relationships preserved.*

Acknowledgments

We thank Chris Zerr, Dr Renato Oliveira, and Monica Botha Hansson for their feedback regarding the generated explanations.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5 in order to generate natural language explanations from SHAP attributions as described in the methodology. Claude Opus 4.6 was used to correct grammar and spelling mistakes for the main text. After using Claude, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] J. Liu, K. Marriott, T. Dwyer, G. Tack, Increasing user trust in optimisation through feedback and interaction, ACM Transactions on Computer-Human Interaction 29 (2023).

[2] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.

[3] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.

[4] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 3319–3328.

[5] C. Henkel, T. Gröb, T. Steens, T. Maciol, S. Niessen, Interpretable data-driven model predictive control of building energy systems using SHAP, Energy and AI 16 (2024) 100336.

[6] D. Martens, J. Hinns, C. Dams, M. Vergouwen, T. Evgeniou, Tell me a story! narrative-driven xai with large language models, Decision Support Systems 191 (2025) 114402. doi:https://doi.org/10.1016/j.dss.2025.114402.

[7] K. W. Bock, K. Coussement, A. De Caigny, et al., Explainable AI for operational research: A defining framework, methods, applications, and a research agenda, European Journal of Operational Research 317 (2024) 249–272.

[8] A. Korikov, J. C. Beck, Counterfactual explanations via inverse constraint programming, in: Leibniz International Proceedings in Informatics, volume 210, 2021, pp. 1–17.

[9] B. Biemans, P. Troubil, I. Grau, W. P. Nuijten, Explainable optimization: Leveraging large language models for user-friendly explanations, in: xAI 2025, CCIS, volume 2578, Springer, 2026, pp. 44–67.

[10] B. Amos, J. Z. Kolter, OptNet: Differentiable optimization as a layer in neural networks, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 136–145.

[11] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, Z. Kolter, Differentiable convex optimization layers, in: Advances in Neural Information Processing Systems, 2019.

[12] M. Xu, T. Molloy, S. Gould, Revisiting implicit differentiation for learning problems in optimal control, in: Advances in Neural Information Processing Systems (NeurIPS), volume 36, 2023, pp. 7979–7992.

[13] H. Dundar, H. Benzer, N. Aydogan, Application of population balance model to HPGR crushing, Minerals Engineering 50–51 (2013) 114–120.

[14] A. L. Dontchev, R. T. Rockafellar, Implicit Functions and Solution Mappings: A View from Variational Analysis, Springer Series in Operations Research and Financial Engineering, 2 ed., Springer, 2014.

[15] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-Lopez, F. Pedregosa, J.-P. Vert, Efficient and modular implicit differentiation, in: Advances in Neural Information Processing Systems, volume 35, 2022, pp. 5230–5242.