A Novel Hybrid Modeling and Optimization Approach for Energy Management in Industrial Utility Systems

Von der Fakultät für Maschinenbau, Elektro- und Energiesysteme
der Brandenburgischen Technischen Universität Cottbus–Senftenberg
zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

M.Sc.

Loukas Kyriakidis

geboren am 17.09.1995 in Göttingen

Vorsitzender: Prof. Dr.-Ing. Harvey Arellano-Garcia
Gutachter: Prof. Dr. rer. nat. habil. Uwe Riedel
Gutachter: Prof. Joseph Morlier, PhD
Tag der mündlichen Prüfung: 24.04.2026

DOI: https://doi.org/10.26127/BTUOpen-7420

To my family

Acknowledgments

This research would not have been possible without the valuable contributions and support of many individuals, to whom I am deeply grateful. First, I would like to express my sincere gratitude to Prof. Dr. Uwe Riedel for giving me the opportunity to work at the German Aerospace Center, Institute "Low-Carbon Industrial Processes" and pursue my PhD thesis. I truly appreciate his support and the valuable discussions during the PhD workshops. I would also like to extend my sincere thanks to Dr. Diana Wenzke and Dr. Tom Lorenz for giving me the opportunity to work in the department "Simulation and Virtual Design" and supporting the development of my PhD topic. Their readiness for insightful discussions, as well as their guidance and encouragement, have been invaluable throughout this journey. In addition, I would like to express my sincere gratitude to Prof. Dr. Panagiotis Stathopoulos for his guidance in helping me define the research question of this PhD. His insights and suggestions were instrumental in shaping the direction of my work and providing a strong foundation for this research.

I would like to express my heartfelt gratitude to Prof. Dr. Miguel Alfonso Mendez for the excellent cooperation during the development of the hybrid optimizer "BO-IPOPT". His expertise, insightful guidance, and support have been invaluable. I also deeply appreciate the opportunity to work with him during my stay in Belgium at VKI, where our numerous discussions greatly enriched my research. His mentorship has had a significant impact on my academic development, and I am incredibly thankful for all the advice and encouragement he has provided throughout this process. I am deeply thankful to Dr. Martin Bähr for his generous support, helpful advice, and many insightful discussions throughout my PhD.

Moreover, I would like to thank Dr. Michael Lockan and Jens Gollasch for the valuable discussions on Bayesian optimization and the insightful exchange of ideas on this topic. I am also grateful to Dr. Maria Isabel Roldán Serrano and Rushit Kansara for the excellent cooperation throughout the European project "Sinnogenes". Special thanks also go to Anh Phong Tran for introducing me to Modelica and for providing helpful information about the high-temperature heat pump "CoBra". I am grateful to Dr. Saskia Bublitz as well for carefully reviewing my thesis and providing valuable feedback. Additionally, I extend my gratitude to the rest of my colleagues at the institute "Low-Carbon Industrial Processes" for the interesting discussions, collaboration, and support throughout this process.

Finally, and most of all, I would like to thank my family for their patience, understanding, and endless encouragement. Without their support and sacrifices, this achievement would not have been possible.

Abstract

Industrial energy utility systems are increasingly required to reduce operating costs and CO_2 emissions through effective energy management while integrating variable renewable energy sources. The volatile nature of renewable energies makes accurate prediction of their power generation difficult, requiring the systems to constantly adapt to fluctuations, and necessitating frequent re-optimization to maintain reliable and efficient system operation. The rolling horizon approach addresses these challenges by enabling adaptive decision-making, iteratively solving an optimization problem over consecutive time horizons while regularly updating input data. However, ensuring a reliable, real-time operation of these systems by this approach requires accurate component modeling and advanced optimization methods. On the modeling side, purely physics-based models become extremely complex when applied to systems with many interacting components and detailed dynamics. Simulation-based surrogate models, although more flexible, may not fully capture real-world conditions, while surrogate models relying solely on experimental data often lack extrapolative generalization, particularly under limited data availability or changing operating conditions. To overcome these limitations, this work develops a hybrid modeling methodology that combines simulation and experimental data in a single surrogate model to accurately represent system components. Adaptive weighting prioritizes the most reliable information depending on data quantity and noise. In addition, the model can be retrained in real time as new data become available. The approach is validated on well-known test functions under varying dimensionality, data availability, and noise, as well as on a high-temperature heat pump at pilot scale, demonstrating high accuracy, robustness, scalability, and extrapolative generalization. On the optimization side, the nonlinear, nonconvex, and constrained problems arising in energy management must be solved accurately and within short computational times. State-of-the-art solvers, however, often suffer from high computational costs, convergence to local optima, and slow convergence, particularly as the number of components and constraints increases. To tackle this challenge, this work introduces BO-IPOPT, a hybrid optimization method that integrates the global exploration capabilities of Bayesian optimization (BO) with the fast local convergence of the Interior Point OPTimizer (IPOPT). Key methodological advances include an efficient candidate set strategy to provide IPOPT with high-quality starting points at low computational cost, a unified surrogate modeling approach for effectively handling equality and inequality constraints via an augmented Lagrangian with slack variables, and adaptive trust regions that guide IPOPT within a more focused search space reducing highly suboptimal areas and enhancing the convergence speed of the hybrid method. In addition, a reduced number of user-defined hyperparameters is achieved through parameter studies, improving the global

guidance of BO and reducing the computational resources required by the hybrid method. Applied to test and conceptual use cases with varying sizes, nonconvexity, and nonlinearity, the proposed method achieves high accuracy, robustness, and computational efficiency, particularly in high-dimensional and complex cases where other methods struggle. The results show that it delivers up to 120 % lower objective function values within the same CPU time compared to state-of-the-art optimizers. In the last part of this work, the BO-IPOPT framework is demonstrated on the energy management of both a conceptual utility system for renewable steam generation (minimizing operating costs) and a real-world case study from the food and cosmetics sector in Germany (minimizing both operating costs and CO_2 emissions). In both cases, the method consistently outperforms state-of-the-art solvers in solution accuracy and robustness, achieving up to 35 % lower objective function values within the same CPU time, while ensuring feasible solutions, and highlighting its strong potential as a promising approach for complex energy management systems. Finally, the effects of key parameters – such as CPU time of the optimizer per rolling horizon iteration, forecast uncertainty, and optimization horizon length – are analyzed to evaluate their impact on the system performance and decision quality.

Zusammenfassung

Industrielle Energieversorgungssysteme müssen zunehmend Betriebskosten und CO_2-Emissionen durch ein effektives Energiemanagement reduzieren und gleichzeitig variable erneuerbare Energiequellen integrieren. Die volatile Natur erneuerbarer Energien erschwert genaue Vorhersage ihrer Stromerzeugung, so dass sich die Systeme ständig an die Schwankungen anpassen müssen und eine häufige Re-Optimierung erforderlich ist, um einen zuverlässigen und effizienten Systembetrieb zu gewährleisten. Der Ansatz des rollierenden Horizonts adressiert diese Herausforderungen, indem er eine adaptive Entscheidungsfindung ermöglicht. Dabei wird ein Optimierungsproblem iterativ über aufeinanderfolgende Zeithorizonte gelöst, während die Eingangsdaten regelmäßig aktualisiert werden. Für einen zuverlässigen, echtzeitfähigen Betrieb dieser Systeme mit diesem Ansatz sind jedoch sowohl eine genaue Modellierung der Komponenten als auch fortschrittliche Optimierungsmethoden erforderlich. Auf der Modellierungsebene werden rein physikbasierte Modelle bei Systemen mit vielen interagierenden Komponenten und detaillierten Dynamiken extrem komplex. Simulationsbasierte Ersatzmodelle sind zwar flexibler, erfassen aber häufig nicht vollständig die realen Betriebsbedingungen, während Ersatzmodelle, die sich ausschließlich auf experimentelle Daten stützen, häufig an extrapolativer Generalisierungsfähigkeit mangeln, insbesondere bei begrenzter Datenverfügbarkeit oder wechselnden Betriebsbedingungen. Um diese Einschränkungen zu überwinden, entwickelt diese Arbeit eine hybride Modellierungsmethodik, die Simulationsdaten und experimentelle Daten in einem einzigen Ersatzmodell kombiniert, um die Systemkomponenten genau abzubilden. Adaptive Gewichtung priorisiert die zuverlässigsten Informationen in Abhängigkeit von Datenmenge und Rauschen. Zusätzlich kann das Modell in Echtzeit neu trainiert werden, sobald neue Daten verfügbar sind. Der Ansatz wird anhand bekannter Test-Funktionen unter variierender Dimensionalität, Datenverfügbarkeit und Rauschpegel sowie an einer Hochtemperatur-Wärmepumpe im Pilotmaßstab validiert und zeigt hohe Genauigkeit, Robustheit, Skalierbarkeit und extrapolative Generalisierungsfähigkeit. Auf der Optimierungsebene müssen die nichtlinearen, nichtkonvexen und restringierten Probleme, die im Energiemanagement auftreten, sowohl genau als auch innerhalb kurzer Rechenzeiten gelöst werden. Klassische Lösungsverfahren stoßen häufig auf hohe Rechenkosten, Konvergenz zu lokalen Optima und langsame Konvergenz, insbesondere wenn die Anzahl der Komponenten und Nebenbedingungen zunimmt. Um diese Herausforderungen zu meistern, führt diese Arbeit BO-IPOPT ein, eine hybride Optimierungsmethode, die die globalen Explorationsfähigkeiten der Bayesian-Optimierung (BO) mit der schnellen lokalen Konvergenz des Interior Point OPTimizer (IPOPT) kombiniert. Wichtige methodische Fortschritte umfassen eine effiziente Kandidatenauswahl zur Bereitstellung hochwertiger Startpunkte für IPOPT

bei geringen Rechenkosten, einen vereinheitlichten Ersatzmodellierungsansatz für die effektive Behandlung von Gleichheits- und Ungleichheitsbedingungen über ein augmentiertes LAGRANGE-Funktional mit Schlupfvariablen sowie adaptive Vertrauensbereiche, die IPOPT in einem fokussierteren Suchraum führen. Zusätzlich wird die Anzahl der vom Benutzer zu definierenden Hyperparameter durch Parameterstudien reduziert, wodurch die globale Führung von BO verbessert und die benötigten Rechenressourcen der hybriden Methode verringert werden. Angewendet auf Testprobleme und konzeptionelle Anwendungsfälle mit unterschiedlicher Größe, Nichtkonvexität und Nichtlinearität erreicht die vorgeschlagene Methode hohe Genauigkeit, Robustheit und Recheneffizienz, insbesondere bei hochdimensionalen und komplexen Problemen, bei denen andere Methoden Schwierigkeiten haben. Die Ergebnisse zeigen, dass sie im Vergleich zu den Stand-der-Technik-Optimierern bis zu 120 % niedrigere Zielfunktionswerte bei gleicher CPU-Zeit erzielt. Im letzten Teil dieser Arbeit wird der BO-IPOPT-Ansatz auf das Energiemanagement sowohl eines konzeptionellen Energieversorgungssystems zur Dampferzeugung aus erneuerbaren Energien (Minimierung der Betriebskosten) als auch auf eine reale Fallstudie aus dem Lebensmittel- und Kosmetiksektor in Deutschland (Minimierung der Betriebskosten und CO_2-Emissionen) angewendet. In beiden Fällen übertrifft die Methode konstant den Stand der Technik in Bezug auf Lösungsgenauigkeit und Robustheit, erreicht bis zu 35 % niedrigere Zielfunktionswerte bei gleicher Rechenzeit, wobei zulässige Lösungen gewährleistet werden, und unterstreicht ihr Potenzial als vielversprechender Ansatz für komplexe Energiemanagementsysteme. Abschließend werden die Auswirkungen von Schlüsselparametern – wie die CPU-Zeit des Optimierers pro rollierender Horizont-Iteration, die Prognoseunsicherheit und die Länge des Optimierungshorizonts – analysiert, um deren Einfluss auf die Systemleistung und die Qualität der Entscheidungen zu bewerten.

Related Publications

This thesis is partially based on previously published work. The following sections categorize these contributions into journal articles, conference proceedings, and oral presentations.

Journal Articles

– L. Kyriakidis, M. A. Mendez, and M. Bähr. A hybrid algorithm based on Bayesian optimization and Interior Point OPTimizer for optimal operation of energy conversion systems. *Energy*, 312:1-10, 2024. doi: https://doi.org/10.1016/j.energy.2024.133416.

– L. Kyriakidis, M. Bähr, and M. A. Mendez. Enhanced Hybrid Algorithm based on Bayesian Optimization and Interior Point OPTimizer for Constrained Optimization. *Optimization and Engineering*, pages 1-52, 2025. doi: https://doi.org/10.1007/s11081-025-09975-y.

– L. Kyriakidis, R. Kansara, and M. I. Roldán Serrano. Energy Management of Industrial Energy Systems via Rolling Horizon and Hybrid Optimization: A Real-Plant Application in Germany. *Energies*, 18:1-29, 2025. doi: https://doi.org/10.3390/en18153977.

Conference Papers

– L. Kyriakidis, M. A. Mendez, and M. Bähr. A Hybrid Algorithm Based on Bayesian Optimization and Interior Point OPTimizer for Optimal Operation of Energy Conversion Systems. In *Proceedings of ECOS 2023: 36th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS2023)*, pages 1299-1310, Las Palmas de Gran Canaria, Spain, June 2023. doi: https://doi.org/10.52202/069564-0118.

– L. Kyriakidis and M. Bähr. ENERGY MANAGEMENT OF AN INDUSTRIAL POWER-TO-HEAT SYSTEM USING A HYBRID OPTIMIZATION ALGORITHM. In *Proceedings of ECOS 2025: 38th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS2025)*, Paris, France, June-July 2025. Accepted, https://www.scopus.com/pages/publications/105037454568.

– L. Kyriakidis and A. Thapa. HYBRID APPROACH BASED ON SIMULATION AND EXPERIMENTAL DATA FOR SURROGATE-BASED MODELING: A CASE STUDY OF A HIGH-

TEMPERATURE HEAT PUMP. In *Proceedings of ASME 2025: International Mechanical Engineering Congress and Exposition (IMECE2025)*, pages 1-10, Memphis, USA, November 2025. doi: https://doi.org/10.1115/IMECE2025-167746.

Oral Presentations without Proceedings

– L. Kyriakidis and M. Bähr. A Hybrid Optimization Algorithm for Energy Management of an Industrial Power-to-Heat System. In *ASMO/ISSMO 2024: 13th ASMO-UK/2nd ASMO-EUROPE/ISSMO conference on Engineering Design Optimization*, Cenaero, Belgium, July 2024.

– L. Kyriakidis. Wind and Solar Data Forecasting for Energy Management of Industrial Utility Systems. In *Symposium on Renewable Feed-in Forecasting*, Cottbus, Germany, November 2025.

Contents

Nomenclature

Acronyms

AI	Artificial intelligence
AIC	Akaike information criterion
AL	Augmented Lagrangian
ALBO	Augmented Lagrangian Bayesian optimization
ANN	Artificial neural network
AR	Autoregressive
ARIMA	Autoregressive integrated moving average
ARMA	Autoregressive moving average
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BO	Bayesian optimization
BOwLS	Bayesian optimization with local search
CNN	Convolutional neural networks
COIN-OR	Computational infrastructure for operations research
CPU	Central processing unit
CSO	Competitive swarm optimizers
CV	Cross-validation
DE	Differential evolution
DED	Dynamic economic dispatch
Dir	Direct
DirMO	Direct multiple-output
DirRec	Direct-recursive
DL	Deep learning
EGORSE	Efficient global optimization coupled with random and supervised embedding
EI	Expected improvement
ETR	Extra tree regression

GA	Genetic algorithm
GP	Gaussian process
GPR	Gaussian process regression
GWS	German weather service
HDBO	High-dimensional Bayesian optimization
HESBO	Hashing-enhanced subspace Bayesian optimization
HGBR	Histogram-based gradient boosting regression
HP	Heat pump
HTF	Heat transfer fluid
HTHP	High-temperature heat pump
HTHX	High-temperature heat exchanger
IPOPT	Interior Point OPTimizer
KNN	k-nearest neighbors
KPLS	Kriging with partial least squares
L-BFGS-B	BFGS with limited memory and boundaries
LCB	Lower confidence bound
LogEI	Logarithmic expected improvement
LSTM	Long short-term memory
LTHX	Low-temperature heat exchanger
MA	Moving average
MAE	Mean absolute error
MIMO	Multiple-input multiple-output
ML	Machine learning
MLR	Multiple linear regression
MPR	Multiple polynomial regression
MS	Multi-start
NNCT	Non-negative constraint theory
OLS	Ordinary least squares
OP	Oil pump
PCHIP	Piecewise cubic Hermite interpolating polynomial
PI	Probability of improvement
PID	Proportional-integral-derivative
PSO	Particle swarm optimization

PV	Photovoltaics
q-EI	Multipoint expected improvement
RBF	Radial basis function
Rec	Recursive
REMBO	Random embedding Bayesian optimization
RFR	Random forest regression
RHA	Rolling horizon approach
RMSE	Root mean squared error
RNN	Recurrent neural network
RQ	Research question
RSG	Renewable steam generation
SCBO	Scalable constrained Bayesian optimization
SEGO	Super efficient global optimization
SEGOKPLS	Super efficient global optimization with kriging models and partial least squares
SEGOKPLS+K	SEGOKPLS, extended through additional kriging optimization (+K)
SG	Steam generator
STC	Solar thermal collector
SVR	Support vector regression
TES	Thermal energy storage
TuRBO	Trust region Bayesian optimization
WB2	Watson and Barnes 2nd acquisition criterion
WB2S	Smooth variant of the Watson and Barnes 2nd acquisition criterion
WS	Warm-start
WT	Wind turbine

Greek Symbols

α	Regularization term, various
α	Weighted criterion in BO-IPOPT with linear model, 1 (dimensionless)
$\alpha_{1,l}$	Fraction of STC output directed to layer l of stratified TES 1, 1 (dimensionless)
$\alpha_{2,l}$	Fraction of STC output directed to layer l of stratified TES 2, 1 (dimensionless)
α_i, α_i^*	Positive and negative Lagrange multipliers, 1 (dimensionless)
β	Fluid bypass, 1 (dimensionless)
β, θ, ϕ	Regression coefficients, 1 (dimensionless)

$\beta_{\text{STC},i}$	Coefficients ($i = 1, 2, 3, 4$) for thermal losses in STC, various
γ	Relaxation parameter, 1 (dimensionless)
γ_{PV}	Activation status of PV, 1 (dimensionless)
γ_{STC}	Activation status of STC, 1 (dimensionless)
$\boldsymbol{\delta}$, $\boldsymbol{\tau}$	Penalty weight vectors, various
Δp	Pressure losses, kg/(m s^2)
Δt	Time step, s
$\Delta T_{\text{STC,amb}}$	Temperature difference between STC and ambient, K
$\boldsymbol{\epsilon}$	Threshold for equality constraints in AL, various
ϵ	Residual error of regression model, various
ϵ_{ch}	Effectiveness in TES during charge (use case for RSG), 1 (dimensionless)
ϵ_{dch}	Effectiveness in TES during discharge (use case for RSG), 1 (dimensionless)
ϵ_{s}	Effectiveness in TES model (use case for RSG), 1 (dimensionless)
η_{OP}	Pump efficiency, 1 (dimensionless)
$\eta_{\text{PV,inv}}$	Inverter efficiency in the PV model, 1 (dimensionless)
$\eta_{\text{PV,nom}}$	Nominal efficiency of the PV module, 1 (dimensionless)
$\eta_{\text{STC,opt}}$	Optical efficiency, 1 (dimensionless)
θ_{STC}	Angle of incidence in STC, °
$\boldsymbol{\lambda}$	LAGRANGE multiplier vector, various
λ	Weighting factor in hybrid modeling, 1 (dimensionless)
μ	Barrier parameter in IPOPT, various
ξ	Exploration term, various
ρ	Density, kg/m^3
ρ	Penalty term, various
$\sigma(\cdot)$	Logistic sigmoid function, 1 (dimensionless)
σ^2	Population variance, various
τ_f	Failure tolerance in the trust region method, 1 (dimensionless)
τ_s	Success tolerance in the trust region method, 1 (dimensionless)
$\phi(\cdot)$	Nonlinear mapping function in SVR, various
Ω	Set of $\mathbb{R}^n$, various
$\tilde{\Omega}$	Shifted Ω in IPOPT, various

Latin Symbols

$\boldsymbol{a}$	Vector of coefficients in bicubic interpolation, various
A	Effective heat transfer area in TES, m^2
a, b, c, d	PCHIP coefficients, various
a_i	Quadratic cost coefficient for generator i in DED problem, USD/(W^2 s)
$a_{\mathrm{PV},i}$	Coefficients ($i = 1, 2, 3$) for the part-load performance of PV, various
A_{PV}	PV area, m^2
A_{STC}	STC area, m^2
b	Bias, various
b_i	Linear cost coefficient for generator i in DED problem, USD/J
$\boldsymbol{c}$	Cell input activation vector, 1 (dimensionless)
C	Maximum electricity price of the optimization horizon, €/J
$\boldsymbol{c}'$	Cell state vector, 1 (dimensionless)
c_i	Fixed cost coefficient for generator i in DED problem, USD/s
c_p	Specific heat capacity, J/(kg K)
$c_{p,\mathrm{f}}$	Specific heat capacity of thermal oil, J/(kg K)
$c_{p,\mathrm{s}}$	Specific heat capacity of TES (use case for RSG), J/(kg K)
$c_{p,\mathrm{water}}$	Specific heat capacity of water, J/(kg K)
D	Set of points, various
d	Order of derivative in ARIMA, 1 (dimensionless)
$d(\cdot, \cdot)$	Euclidean distance between points, various
D_0	Initial set of points, various
d_{min}, d_{max}	Minimum and maximum distance between new and already evaluated points in BO-IPOPT with linear model, various
DR_i	Ramp-down limit for generator i in DED problem, W/s
E	Energy, J
e_i	Amplitude of the valve-point effect for generator i in DED problem, USD/s
$\boldsymbol{f}$	Vector of function values and derivatives at the cell corners in bicubic interpolation, various
$\boldsymbol{f}$	Activation vector of the forget gate in LSTM, 1 (dimensionless)
f	Objective function, various
f_i	Frequency coefficient of the valve-point effect for generator i in DED problem, 1/W
$\boldsymbol{g}$	Set of inequality constraints, various
$g_{\mathrm{em,grid}}$	Emission factor, €/J

$g_{\text{pr,grid}}$	Grid electricity price, €/J
$g_{\text{PV,sell}}$	PV feed-in tariff, €/J
$\boldsymbol{h}$	Set of equality constraints, various
$\boldsymbol{h}$	Output vector of the LSTM unit, 1 (dimensionless)
H	Forecasting horizon, s
h	Heat transfer coefficient, W/(m^2 K)
h	Number of independent models in Dir and DirRec strategy, 1 (dimensionless)
h	Specific enthalpy, J/kg
$h_{i,l}$	Specific enthalpy of the incoming stream i at the interconnection node l, J/kg
$h_{j,l}$	Specific enthalpy of the outgoing stream j at the interconnection node l, J/kg
$\boldsymbol{i}$	Activation vector of the input gate in LSTM, 1 (dimensionless)
I	Solar irradiance, W/m^2
$I_{\text{PV,nom}}$	Nominal solar irradiance, W/m^2
K	Number of independent variables (features), 1 (dimensionless)
K	Number of outer iterations, 1 (dimensionless)
k	Discretization points, 1 (dimensionless)
k	Number of closest neighbors in KNN, 1 (dimensionless)
k	Thermal conductivity, W/(m K)
$k(\cdot)$	Kernel function, various
K_{STC}	Incidence correction factor in STC, 1 (dimensionless)
L	Log-likehood estimate, 1 (dimensionless)
L	Side length in trust region method, 1 (dimensionless)
l	Kernel's length scale, various
L_{init}	Initial side length in the trust region method, 1 (dimensionless)
L_{max}	Maximum side length in the trust region method, 1 (dimensionless)
L_{min}	Minimum side length in the trust region method, 1 (dimensionless)
m	Slope in PCHIP, various
m_{s}	Mass of TES (use case for RSG), kg
$\dot{m}$	Mass flow rate, kg/s
$\dot{m}_{\text{I}}$	Inlet mass flow rate of HTHP on the sink, kg/s
$\dot{m}_{\text{II}}$	Outlet mass flow rate of HTHP on the sink, kg/s
$\dot{m}_{\text{III}}$	Inlet mass flow rate of HTHP on the source, kg/s
$\dot{m}_{\text{IV}}$	Outlet mass flow rate of HTHP on the source, kg/s

$\dot{m}_{\text{demand}}$	Mass flow rate demand, kg/s
$\dot{m}_{\text{h,in}}$	Inlet mass flow rate of HP on the sink, kg/s
$\dot{m}_i$	Mass flow rate in node i of the stratified TES, kg/s
$\dot{m}_{\text{in}}$	Mass flow rate of fluid entering the stratified TES, kg/s
$\dot{m}_{i,l}$	Mass flow rate of the incoming stream i at the interconnection node l, kg/s
$\dot{m}_{j,l}$	Mass flow rate of the outgoing stream j at the interconnection node l, kg/s
$\dot{m}_{\text{nom}}$	Nominal mass flow rate, kg/s
$\dot{m}_{\text{out}}$	Mass flow rate of fluid leaving the stratified TES, kg/s
$\dot{m}_{\text{STC}}$	Mass flow rate through the STC, kg/s
$\dot{m}_{\text{TES,3,out}}$	Mass flow rate in the top layer of the stratified TES 3, kg/s
n	Number of dimensions in optimization problems, 1 (dimensionless)
n	Number of outputs of each block in DirMO strategy, 1 (dimensionless)
N_0	Number of initialization points, 1 (dimensionless)
N_{bc}	Number of best candidates, 1 (dimensionless)
N_c	Number of candidates considered in EI, 1 (dimensionless)
N_{exp}	Number of experimental data, 1 (dimensionless)
n_f	Failure counter in the trust region method, 1 (dimensionless)
N_{G}	Total number of generation units in DED problem, 1 (dimensionless)
n_s	Success counter in the trust region method, 1 (dimensionless)
N_{sim}	Number of simulation data, 1 (dimensionless)
$\boldsymbol{o}$	Activation vector of the output gate in LSTM, 1 (dimensionless)
$\boldsymbol{P}$	Vector for electric power, W
p	Number of equality constraints, 1 (dimensionless)
p	Number of independent models in DirMO strategy, 1 (dimensionless)
p	Order of AR component in ARIMA, 1 (dimensionless)
p	Pressure, kg/(m s^2)
$p(\cdot)$	1-D and 2-D cubic polynomial, various
$P_{\text{D}t}$	Total power demand at time t in DED problem, W
P_{el}	Electric power of HP, W
P_{grid}	Electric power from grid, W
$P_i^{\max}$	Maximum power output limit for generator i in DED problem, W
$P_i^{\min}$	Minimum power output limit for generator i in DED problem, W
P_{it}	Power output of generator i at time t in DED problem, W

P_{OP}	Electric power of OP, W
$P_{\text{OP,total}}$	Electric power required by all OPs, W
P_{PV}	PV power, W
$P_{\text{PV,peak}}$	Peak PV power, W
$P_{\text{PV,sell}}$	PV power fed into the grid, W
P_{total}	Total electric power, W
P_{WT}	Power output of WT, W
q	Number of inequality constraints, 1 (dimensionless)
q	Order of MA component in ARIMA, 1 (dimensionless)
$\dot{Q}$	Heat flow, W
$\dot{Q}_{\text{f}}$	Heat transferred from the fluid, W
$\dot{Q}_{\text{loss}}$	Heat losses, W
$\dot{Q}_{l,\text{loss}}$	Heat losses at the interconnection node l, W
$\dot{Q}_{\text{s,ch}}$	TES heat flow during charge (use case for RSG), W
$\dot{Q}_{\text{s,dch}}$	TES heat flow during discharge (use case for RSG), W
R	Motor speed, rad/s
R^2	Coefficient of determination, 1 (dimensionless)
$\boldsymbol{s}$	Slack variable vector, various
s	Number of blocks in DirMO forecasting strategy, 1 (dimensionless)
s	Sample standard deviation, various
s	Secant slope in PCHIP, various
SD	Sunshine duration, s
T	Temperature, K
T	Total number of time intervals in DED problem, 1 (dimensionless)
t	Time, s
T_1	Outlet temperature during charge of TES (use case for RSG), K
T_2	Inlet temperature of SG, K
T_3	Outlet temperature of SG, K
T_4	Outlet temperature during discharge of TES (use case for RSG), K
T_{I}	Inlet temperature of HTHP on the sink, K
T_{II}	Outlet temperature of HTHP on the sink, K
T_{III}	Inlet temperature of HTHP on the source, K
T_{IV}	Outlet temperature of HTHP on the source, K

T_{ambient}	Ambient temperature, K
$T_{\text{c,in}}$	Inlet temperature of HP on the source, K
T_{demand}	Temperature demand, K
$T_{\text{h,in}}$	Inlet temperature of HP on the sink, K
$T_{\text{h,out}}$	Outlet temperature of HP on the sink, K
T_i	Temperature in node i of the stratified TES, K
T_{in}	Temperature of fluid entering the stratified TES, K
T_{in}	Temperature of inlet fluid in TES (use case for RSG), K
T_{out}	Temperature of outlet fluid in TES (use case for RSG), K
$T_{\text{PV,nom}}$	Nominal temperature in the PV model, K
T_{s}	Temperature of TES, K
$T_{\text{s,max}}$	Maximum temperature of TES, K
$T_{\text{s,min}}$	Minimum temperature of TES, K
$T_{\text{STC,in}}$	Inlet temperature of STC, K
$T_{\text{STC,out}}$	Outlet temperature of STC, K
$T_{\text{TES,1},l}$	Temperature in each layer l of the stratified TES 1, K
$T_{\text{TES,2},l}$	Temperature in each layer l of the stratified TES 2, K
$T_{\text{TES,3,out}}$	Temperature in the top layer of the stratified TES 3, K
$\tanh(\cdot)$	Hyperbolic tangent function, 1 (dimensionless)
$\mathbf{tr}_{\text{lb}}$	Low bound in the trust region method, 1 (dimensionless)
$\mathbf{tr}_{\text{ub}}$	Upper bound in the trust region method, 1 (dimensionless)
U	Overall heat transfer coefficient, W/(m^2 K)
U, W	Weight matrices in LSTM, 1 (dimensionless)
u	Augmented objective function, various
$\hat{u}$	GP model of u, various
$\hat{u}_{\text{min}}$, $\hat{u}_{\text{max}}$	Minimum and maximum values of the linear model in BO-IPOPT, various
UR_i	Ramp-up limit for generator i in DED problem, W/s
$\boldsymbol{v}$	Logical vector in AL, 1 (dimensionless)
V	Volume, m^3
v	Wind speed, m/s
v_d	Distance criterion in BO-IPOPT with linear model, 1 (dimensionless)
v_{ref}	Wind speed at the reference height z_{ref}, m/s
v_u	Quality criterion in BO-IPOPT with linear model, 1 (dimensionless)

Var_{exp}	Variance in experimental data, various
Var_{sim}	Variance in simulation data, various
$\dot{V}$	Volume flow, m^3/s
$\dot{V}_{\text{vert,in}}$	Vertical volume inflow from neighboring stratified TES node, m^3/s
$\dot{V}_{\text{vert,out}}$	Vertical volume outflow from neighboring stratified TES node, m^3/s
$\boldsymbol{w}$	Weight vector in SVR, 1 (dimensionless)
w	Stationary time series in ARIMA, various
w	Weights in PCHIP, various
w	Window size in multi-step forecasting strategies, 1 (dimensionless)
w_d	Weight for the distance criterion in BO-IPOPT with linear model, 1 (dimensionless)
w_u	Weight for the quality criterion in BO-IPOPT with linear model, 1 (dimensionless)
w_{cost}	Weight for the operating costs, 1 (dimensionless)
w_{em}	Weight for the CO_2 emissions, 1 (dimensionless)
w_{exp}	Weight for experimental data, 1 (dimensionless)
$w_{\text{exp}}^{\text{norm}}$	Normalized weight for experimental data, 1 (dimensionless)
w_{sim}	Weight for simulation data, 1 (dimensionless)
$w_{\text{sim}}^{\text{norm}}$	Normalized weight for simulation data, 1 (dimensionless)
$\boldsymbol{x}$	Decision variables in optimization problems, various
$\boldsymbol{x}^\star$	$\boldsymbol{x}$-values of the local search, various
$\bar{\boldsymbol{x}}$	$\boldsymbol{x}$-values of sample points in EI, various
$\tilde{\boldsymbol{x}}$	n-dimensional shifted decision variables in IPOPT, various
$\tilde{\boldsymbol{x}}^\star$	$\tilde{\boldsymbol{x}}$-values of the optimal solution, various
X	Constraint matrix relating function data to polynomial coefficients in bicubic interpolation, 1 (dimensionless)
x	Independent variable (feature) of regression, various
x'	Scaled independent variable (feature) of regression, various
$\bar{x}$	Mean value of the independent variable (feature) of regression, various
$\boldsymbol{x}_{\text{center}}$	Center in the trust region method, 1 (dimensionless)
$\boldsymbol{x}_{\text{L}}$	Lower bounds of the decision variables in optimization problems, various
$\boldsymbol{x}_{\text{U}}$	Upper bounds of the decision variables in optimization problems, various
y	Dependent variable of regression, various
$y^\star$	y-value of the local search, various
$\hat{y}_{\text{exp}}$	Output of model for experimental data, various

$\hat{y}_{\text{hyb}}$	Output of the hybrid model, various
$\hat{y}_{\text{sim}}$	Output of model for simulation data, various
y_{exp}	Experimental output value, various
y_{min}	Minimum y-value in BO-IPOPT, MS-IPOPT, and BOwLS, various
y_{sim}	Simulation output value, various
z	Height in WT model, m
z_{ref}	Reference height in WT model, m

1. Introduction

Energy utility systems are an essential part of industrial processes, and their efficient operation is vital for ensuring overall effectiveness. As the transition toward sustainable energy solutions accelerates, these systems are becoming progressively more complex due to the integration of diverse energy sources, storage technologies, and advanced control mechanisms, making optimal energy management more important than ever (Lund et al., 2014). The primary objectives are to lower operating costs, reduce CO_2 emissions, and enhance efficiency (Olatomiwa et al., 2016). At the same time, emerging technologies create new challenges. Energy systems must integrate variable renewable generation, adapt to changing demand patterns (meeting the variable energy requirements of each process), maintain grid stability (avoiding overloads or blackouts), and secure power quality (ensuring electricity meets technical standards) (Khalid, 2024). In addition, forecasting renewable energy output remains highly uncertain, adding significant complexity to the formulation of operational strategies.

Traditionally, industrial systems have relied on simple control strategies, such as proportional-integral-derivative (PID) controllers or rule-based scheduling (Shanmugarjah et al., 2024; Hu, 2025). While effective for maintaining basic operational stability, these methods are limited: PID controllers react only to current errors and do not predict future disturbances or anticipate changes, can not handle multiple objectives or system-wide constraints, and require manual tuning of parameters depending on the system (Hu, 2025). Rule-based control is rigid and non-adaptive, relying on predefined rules that do not adjust to changing operating conditions, uncertainties, or interactions between subsystems (Shanmugarjah et al., 2024). For these reasons, advanced energy management methods that explicitly consider uncertainties, coordinate multiple components, and optimize system-wide objectives have become essential to the effective and sustainable operation of industrial systems.

The role of energy management within the broader framework of real-time optimization is illustrated in Fig. 1.1. In this context, three hierarchical levels are typically distinguished: (i) energy management at the system level, (ii) dynamic control at the component level, and (iii) the real process, including the process plant, instrumentation, and control technology (Kyriakidis et al., 2025a). The first level, which is the focus of this work, corresponds to predictive operational optimization on a time scale ranging from minutes to hours. At this level, the objective is to compute optimal trajectories of setpoints – such as temperatures, mass flows, and power – over consecutive periods in order to minimize the overall operating costs (electricity expenses from grid imports and, where applicable, peak demand charges or revenues from electricity sales) and CO_2 emissions from grid electricity consumption. These optimized setpoints

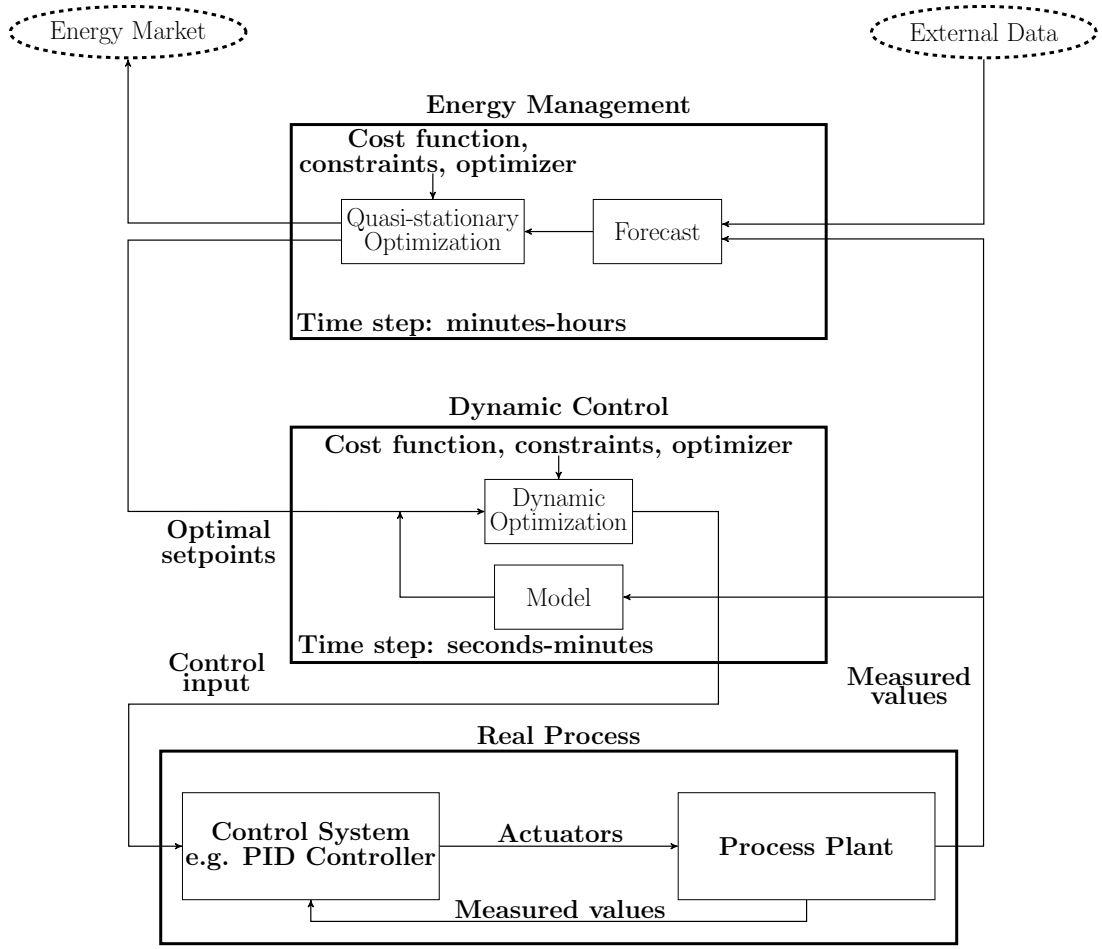

Figure 1.1: Detailed block structure of the different time levels for real-time optimization of a system.

are then passed to the second level, where the component-level dynamic control ensures stable operation by adjusting the system in response to the new targets on a second-to-minute time scale. This interaction between system- and component-level control is particularly important in setups with time-varying operating conditions. Finally, the third level involves real-time execution of control signals, such as those sent to actuators (e.g., valves), in the physical process. However, levels two and three, as well as real-time measurements and disturbances, lie outside the scope of this work. The first level can in principle also interact with energy markets – for instance by incorporating electricity price signals or selling excess energy.

To deal with uncertainties in energy management, a variety of methods have been proposed, with the most well-known being the following: First, the stochastic programming incorporates probability distributions of uncertain variables to represent randomness in the system (Birge and Louveaux, 2011). Second, the robust optimization aims to identify solutions that remain effective across multiple scenarios, often emphasizing resilience under worst-case conditions (Bertsimas and Sim, 2004). Third, the fuzzy logic approaches enable the handling of uncertain or vague information in decision-making contexts (Wu and Xu, 2021). Fourth, the MONTE CARLO simulation relies on repeated random sampling to explore the

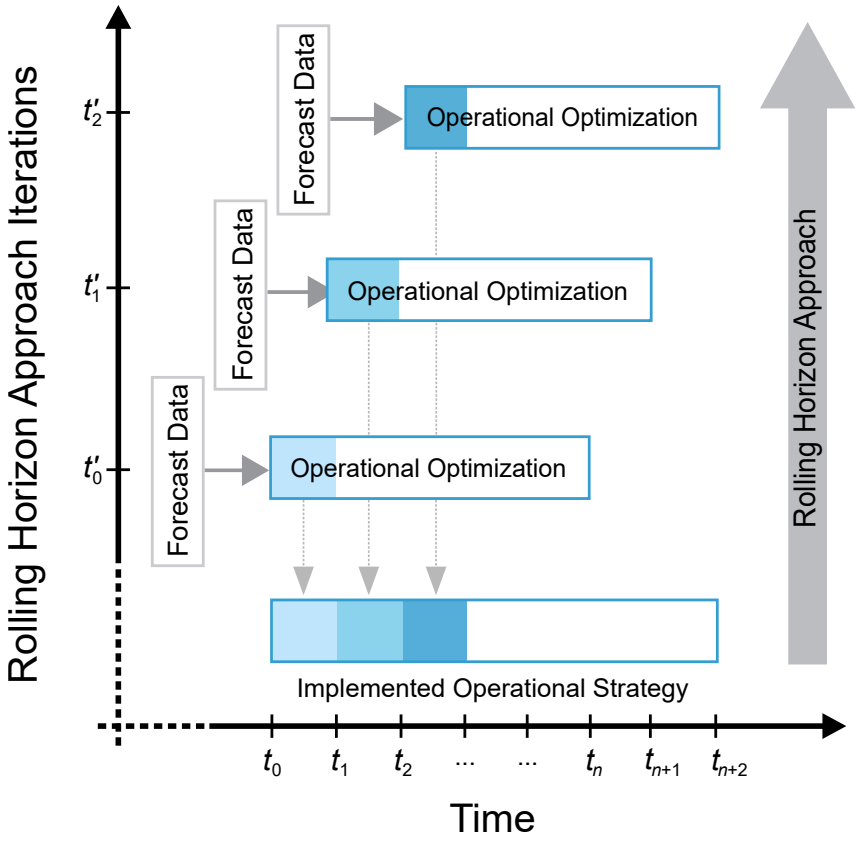

Figure 1.2: Visualization of the rolling horizon approach (RHA).

range of possible outcomes along with their likelihoods (Mavrotas et al., 2010). Despite their usefulness, these techniques generally require significant computational effort, which poses limitations in situations where operational strategies must adapt rapidly to dynamic conditions.

To overcome these challenges, the rolling horizon approach (RHA) has become a widely used strategy, particularly in applications that demand real-time adaptability (Silvente et al., 2015). The method works by solving an optimization problem over a limited future time horizon. From the computed trajectory, only the first control action is implemented, while the remaining steps are discarded. The horizon then shifts one step forward, and the optimization is solved again using newly available information. This rolling procedure enables continuous adjustment of operational strategies as forecasts and real measurements are updated. Such an iterative and forward-looking structure makes RHA especially suitable for renewable energy systems, where fluctuating generation and uncertain conditions require frequent revisions of management decisions. A schematic representation of the approach is shown in Fig. 1.2, highlighting that the input data over the prediction horizon must be forecasted in advance. Within this work, RHA is therefore employed as a practical means to explicitly handle uncertainties in energy management.

Ensuring reliable real-time operation within the RHA framework requires both accurate modeling of the individual system components and advanced optimization techniques capable of handling the complexity of industrial energy utility systems in a limited amount of time. In this work, “complexity”

refers to the challenges arising from the combination of nonlinearities, nonconvexities – which can give rise to multiple local optima in the optimization problem – and high-dimensional decision spaces. Energy management in this context therefore raises two central research questions (RQs):

- **RQ 1**: How can the components of industrial energy utility systems be modeled realistically while adapting to new measurements in real time?
- **RQ 2**: How can complex industrial energy utility systems be optimized in real time with high accuracy?

Addressing RQ 1 requires a careful choice of the modeling approach. A first option is to directly use detailed simulation models and combine them with optimization in energy management. While this allows for high-fidelity representation of systems, the strong coupling between simulation software and optimization algorithms often introduces practical difficulties. These include high computational burden, discontinuities, infeasible operating regions, and convergence issues, which make real-time application challenging (Šturek and Lazarova-Molnar, 2025). For this reason, analytical models formulated directly from physical equations are often preferred. While these models remain the gold standard due to their interpretability and ability to capture system dynamics, they often result in extensive formulations with numerous equations and parameters when applied to systems with many interacting subsystems (Wu et al., 2024). To overcome these challenges, surrogate modeling is commonly applied. Surrogates approximate the input–output behavior of a system using either simulation data or experimental measurements, thereby significantly reducing computational costs while maintaining sufficient predictive accuracy (Šturek and Lazarova-Molnar, 2025). Simulation-based surrogate models benefit from the ability to generate large synthetic datasets without the cost and effort of experimental campaigns (Humbird et al., 2021). This makes them attractive for applications where data requirements are high. However, their main limitation is that the simplifications and assumptions made in the simulation, which neglect some real-world phenomena, can lead to significant deviations from the actual physical system (Humbird et al., 2021). Experiment-based surrogate models, in contrast, are constructed directly from measurement data and therefore better capture real-world phenomena that are often oversimplified in simulations. Yet, they face challenges such as reduced accuracy due to measurement noise, sparse coverage of the input space, and the high cost and logistical difficulty of obtaining high-quality experimental datasets (Letham et al., 2019; Humbird et al., 2021).

To overcome the limitations of relying exclusively on simulation or experimental models, this work introduces a hybrid surrogate modeling methodology that integrates both types of data sources within a single unified model. The approach employs an adaptive weighting mechanism that dynamically prioritizes the more reliable source depending on data quantity and noise level. The proposed method optimizes the integration process directly using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm with limited memory and boundaries (L-BFGS-B) ensuring an efficient parameter estimation of the surrogate model even in high-dimensional settings. Furthermore, the model is designed to be retrainable in real

time, allowing it to continuously adapt as each new measurement becomes available. The methodology is systematically validated on test functions across varying levels of dimensionality, data availability, and noise, showing improved accuracy, robustness, scalability, and extrapolative generalization compared to other methodologies including transfer learning and approaches that maintain separate models for each data source. In addition, the framework is demonstrated on a high-temperature heat pump (HTHP) at pilot scale, confirming its practical relevance and capability to balance predictive fidelity with computational efficiency.

Addressing RQ 2 focuses on how to optimally operate complex energy utility systems within the RHA framework. The effectiveness of rolling horizon strategies strongly depends on the underlying optimization methods, which determine how operational decisions are updated in response to new forecasts and evolving system conditions. Within RHA, a variety of optimization approaches have been employed to handle both linear and nonlinear formulations. For linear problem structures, mixed-integer linear programming has been often applied due to its capability to model both continuous and discrete decision variables efficiently (Silvente et al., 2015; Bischi et al., 2019; Corinaldesi et al., 2020; Étienne Cuisinier et al., 2022; Erdinç, 2023). Standard linear programming, which only accommodates continuous variables, plays a more limited role in the literature (Kallabis, 2020; Bio Gassi and Baysal, 2022). While linear and linearized formulations offer reduced computational times and improved scalability, they often do so at the cost of model realism. Linear and linearized formulations may fail to capture the true system dynamics when strong nonlinearities, wide operating ranges, or complex interactions between components are present (Zhadan et al., 2025). In such cases, linear programming solvers may produce objective values that deviate from those of the true nonlinear system, potentially leading to suboptimal control decisions. In addition, linearization requires careful balance between accuracy and computational efficiency for the grid finess (Zhadan et al., 2025). The introduction of new variables – especially in systems with strong nonlinearities or interacting components – can quickly make the linearization tedious and computationally intensive.

Nonlinear optimization techniques, on the other hand, have seen more limited use in RHA-based energy management, especially when employing deterministic global solvers such as BARON (Tawarmalani and Sahinidis, 2002), which become computationally expensive and impractical for high-dimensional, nonlinear, and nonconvex problems under the strict real-time constraints of energy management. To address nonlinearities in practical applications, some studies have explored global stochastic methods, including genetic algorithm (GA) and particle swarm optimization (PSO) (Li et al., 2019; Mquqwana and Krishnamurthy, 2024), particularly for microgrids and hybrid renewable energy systems. These approaches offer the benefit of escaping local optima, but they generally require a very large number of function evaluations, which can limit their applicability to high-dimensional constrained problems (Kyriakidis and Bähr, 2025; Kyriakidis et al., 2025a).

To mitigate the computational and convergence limitations of conventional stochastic solvers, this work introduces BO-IPOPT, a novel hybrid method designed to efficiently solve the nonlinear, nonconvex, and constrained problems that arise in energy management, particularly in high-dimensional settings

where such characteristics contribute to increased problem complexity. The approach integrates the global exploration of Bayesian optimization (BO) (Jeong and Obayashi, 2005) with the fast local convergence of the Interior Point OPTimizer (IPOPT) (Wächter, 2002). More precisely, BO explores the entire input space on a global scale, capturing smooth variations while overlooking fine details. IPOPT then focuses on the promising regions identified by BO. This integration offers a twofold advantage: IPOPT increases the convergence speed of BO, reducing the number of cost function evaluations, while BO provides high-quality starting points that guide IPOPT toward optimal regions. Since BO is used to guide IPOPT rather than working standalone as it is usually the case, several methodological enhancements are introduced in this work to enable this integration. First, candidate solutions (sample points) are selected at each iteration for the surrogate model of the BO part by considering a finite candidate set rather than the continuous domain, which provides IPOPT with high-quality starting points at low computational cost. Second, constraints are handled through a unified surrogate modeling approach that approximates the objective and constraints within an augmented Lagrangian (AL) framework, introducing slack variables to convert inequalities into equalities (Picheny et al., 2016). This reduces the number of surrogate models and hyperparameters compared to traditional approaches, thereby improving scalability and robustness in high-dimensional problems. Third, a parameter study is conducted to tune the guidance of IPOPT by BO, replacing user-defined hyperparameters with empirically optimized values that balance convergence speed and computational cost. Fourth, adaptive trust regions are employed to guide IPOPT within a more focused search space, reducing highly suboptimal areas and enhancing the convergence speed of the hybrid method.

Building on these methodological developments, this study applies BO-IPOPT to the energy management of two systems – a conceptual and a higher-dimensional real-world industrial energy utility system – both operating with a 15-minute time resolution. The conceptual system for renewable steam generation (RSG), introduced in previous work (Walden et al., 2023), is enhanced here to better reflect realistic behavior, while the real-world system is developed within this thesis. The conceptual system represents a RSG process powered by an on-site wind turbine (WT) and supported by a grid connection to ensure continuous operation during periods of low wind availability. Electrical energy is converted into high-temperature thermal energy through a high-temperature heat pump (HTHP) operating on a closed reverse Brayton cycle. A thermal energy storage (TES) is integrated between the HTHP and the steam generator (SG) to balance variations in supply and provide a constant heat delivery (superheated steam at 13 bar and 215 °C) to the process. To compensate for pressure losses across the TES, HTHP, and SG, oil pumps (OPs) are incorporated into the thermal oil loop, which circulates the heat transfer fluid (HTF) throughout the system. This configuration reflects a typical electrified industrial heat supply, where renewable electricity replaces fossil fuels while maintaining the reliability required by continuous industrial processes.

The real-world system investigated in this work is based on a food and cosmetics production facility in Herzberg (Germany) that has undergone a comprehensive retrofit to decarbonize its energy supply. It

integrates a solar thermal collector (STC), photovoltaic (PV) installation, and three stratified TES units, interconnected through a heat pump (HP). The HP, powered primarily by on-site PV electricity and supported by the grid, upgrades low-temperature solar heat from the STC to process temperatures of up to 90 °C. Two TES units support both the source and sink sides of the HP, while a third TES stores and delivers up to 260 kW of hot water for industrial use. Excess PV electricity can either drive the HP to charge the third TES unit or, if not needed, be exported to the grid. The TES units ensure stable operation during periods of low solar thermal or PV availability. This configuration exemplifies a renewable-based industrial energy system designed for cost-effective and low-emission operation.

The hybrid surrogate modeling is applied specifically to the HTHP of the conceptual system, where experimental data are available, allowing for accurate representation of this key component. To support operational planning, which relies on renewable energy forecasts, various statistical and artificial intelligence (AI)-based forecasting techniques are evaluated for wind (conceptual system) and solar (real-world system) data, exploring different forecasting strategies. For both systems, the total CPU time per RHA iteration is limited to approximately 60 seconds, including forecasting, surrogate model updates if necessary, and optimization for optimal setpoint definition. This ensures that control signals can be sent and system responses received during real-time operation at a 15-minute resolution. Beyond comparing BO-IPOPT to other state-of-the-art optimization techniques (e.g., GA, PSO) within the energy management, this work also investigates how key factors, such as optimizer CPU time (runtime) at each RHA, forecast uncertainty, and optimization horizon length, affect the system operation. This analysis is performed for the conceptual and real-world system, providing comprehensive insights into the practical applicability of the BO-IPOPT framework within a RHA for more effective energy management.

This work is organized as follows: Chapter 2 presents the conceptual energy utility system for RSG and details the modeling of each component involved. It further introduces the hybrid methodology that combines simulation and experimental data for accurately modeling the HTHP, along with the formulation of the resulting optimization problem. Chapter 3 describes several statistical and AI-based methods for forecasting of wind data, comparing their performance in both one-step- and multi-step-ahead forecast, and evaluating various forecasting strategies. Chapter 4 presents the novel hybrid optimization method, BO-IPOPT, including methodological improvements designed to enhance its performance and a comparison with state-of-the-art optimization approaches. Chapter 5 applies BO-IPOPT to the energy management of the conceptual system, comparing its performance with other solvers and analyzing the influence of key parameters such as optimizer's CPU time per RHA iteration, uncertainties in input data, and optimization horizon length on the energy management system. Chapter 6 extends the methodology to a real-world energy utility system, describing the modeling of each component and the formulation of the resulting optimization problem. It further compares several statistical and AI-based models for forecasting of solar data to assess the predictive accuracy and suitability for operational use. Moreover, the chapter compares BO-IPOPT with other state-of-the-art optimization methods within the energy management

framework and evaluates the same performance parameters as in the conceptual case. Finally, Chapter 7 concludes the work with key findings and outlines directions for future research.

2. Conceptual Industrial Utility System for Steam Generation

This chapter introduces a representative industrial energy utility system for RSG considered in this work and outlines the associated optimization problem, which was originally proposed in Walden et al. (2023) and has been further enhanced in this study to better reflect real-world conditions in operational optimization.

2.1. System Description

Electrification of industrial energy utility systems using renewable electricity introduces the challenge of ensuring reliable operation, as the fluctuating nature of renewable energy generation must still meet the heat demand required by industrial processes. To address this, such systems often require both energy storage technologies and supplemental power from the electrical grid to bridge short- and long-term periods of low renewable availability.

The system analyzed in this work, illustrated in Fig. 2.1, is powered by an on-site WT and supported by a grid connection. A HTHP, operating on a closed reverse Brayton cycle with air as the working fluid, converts electrical energy into thermal energy. A TES unit is integrated between the HTHP and the steam generator (SG) to balance variations in supply and provide a constant heat delivery to the process.

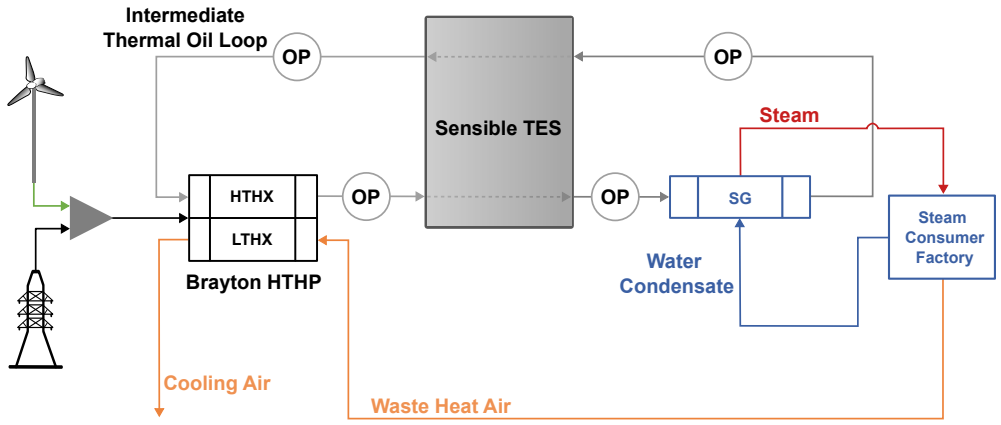

Figure 2.1: Industrial energy utility system for RSG proposed in Walden et al. (2023) and further enhanced in this work.

The case study focuses on meeting a steady heat demand for an industrial process, delivered as superheated steam at 13 bar and 215 °C – parameters chosen to reflect the performance of a typical industrial gas boiler developed by Viessmann Deutschland GmbH (2018).

The HTHP supplies high-temperature thermal energy to an intermediate loop via its high-temperature heat exchanger (HTHX). This loop can either pass through the TES or bypass it. Thermal oil is used as the HTF due to its stability and efficiency across the operating temperature range. During charging, heat from the HTF is stored in the TES before returning to the HTHP. During discharging, the HTF extracts heat from the TES before entering the SG. In idle mode, the HTF bypasses the storage entirely.

As the heat source, the system utilizes waste heat recovered through the low-temperature heat exchanger (LTHX) and modeled as dry air at temperatures between 60 °C and 100 °C, representing typical temperature levels of industrial waste heat (Marina et al., 2021).

Contrary to the assumptions made in Walden et al. (2023), this study accounts for pressure losses after the TES (charging and discharging), HTHP, and SG. To maintain the required pressure levels throughout the system, four OPs after each component are incorporated into the process (Arias et al., 2009).

The following sections describe the modeling of each system component. The system design is based on the following assumptions and simplifications:

- The system is modeled under quasi-stationary conditions, meaning transient dynamics within each time step are neglected, aligning with the system-level energy management presented in Fig. 1.1. While Walden et al. (2023) also consider quasi-stationary behavior and do not include ramp constraints, this study extends that approach by incorporating ramp constraints to account for limitations in how quickly components can adjust their operating states.

- Heat losses are considered only in TES and HTHP.

- No economic benefit from selling excess wind power is considered in this study.

- Operation and maintenance costs of the wind turbine are assumed to be 0.012 €/kWh, as reported in Walden et al. (2023).

- While the Brayton-cycle HTHP also produces cold air, no constraints or utilization strategies are included for this cooling potential in the current scenario.

- The inlet mass flow and inlet temperature at each OP are assumed to be equal to the outlet mass flow and outlet temperature. This simplification is justified because the temperature rise across a properly operating thermal oil pump is negligible (typically less than 1 °C) and the mass flow is conserved (Karassik et al., 2007).

Finally, it should be noted that this study focuses on the energy management of the system under given component sizes and process structure. The design optimization, including sizing of components, is beyond the scope of this work.

2.2. Component Modeling

This section presents the modeling of each individual component in the industrial energy utility system described in Section 2.1. The objective is to develop representations of each component that accurately capture their physical behavior, while being simple enough for efficient integration into the overall optimization framework.

The modeling of the HTHP is presented in detail in the following section (see Section 2.3). The reason for this is the availability of experimental data for the HTHP, which allows for the development of a hybrid surrogate modeling methodology that combines simulation results with real measurements. Since such experimental data are not currently available for the other components of the system, they are not considered in this work. However, the proposed methodology can be applied to these components in future work once corresponding measurement data become accessible.

To enable efficient integration of the component models with the PYTHON-based optimization framework (see Chapter 5) and to avoid direct coupling between modeling softwares, such as EBSILON (STEAG System Technologies, 2025), and PYTHON, simplified physical equations or surrogate models are employed. These approaches are necessary to mitigate issues such as simulation discontinuities, infeasible operating regions, and convergence failures that would arise from direct simulation-based optimization (Šturek and Lazarova-Molnar, 2025). By replacing the full simulation models with continuous, differentiable surrogate functions or simplified physical models, the optimization process benefits from greater numerical stability and faster evaluation times, as the underlying equations provide known gradients that enable robust and efficient convergence in high-dimensional energy management problems.

The following subsections present the modeling approaches for the WT, TES, SG, and OP components, which are integrated into the overall optimization framework in Section 2.4. While these models are sufficient for capturing the essential system behavior, they can be further refined in future work to incorporate more detailed physical dynamics. However, this is not the focus of the present study, which concentrates on the HTHP, for which experimental data is available.

2.2.1. Wind Turbine

The WT model used in this study relies on a manufacturer-provided power curve that directly relates the wind speed to the electrical power output, as done in Walden et al. (2023). This curve-based approach enables an efficient estimation of the WT performance without requiring a detailed representation of turbine aerodynamics, control strategy, or drivetrain dynamics.

The power output is modeled only as a function of the wind speed, i.e., $P_{\mathrm{WT}} = f(v)$. Although, according to IEC standards, all power curves are normalized to a reference air density and the actual power output should ideally be corrected for site-specific air density using the barometric height formula (Floors and Nielsen, 2019), this correction is neglected here. Including air density as an input would introduce an additional variable that must be forecasted within the energy management system (see Chapter 5),

increasing the effort associated with developing an additional forecasting model. Moreover, Walden et al. (2023) has stated that the influence of the air density correction on the power output estimation is relatively small in this context.

Since the wind speed measurements are typically taken at a reference height different from the turbine hub height, an extrapolation method is employed. Specifically, the logarithmic wind profile (Bauelos-Ruedas et al., 2011) is used to estimate the wind speed at the hub height based on the following equation:

$$v(z) = v_{\text{ref}} \frac{\ln(z/z_0)}{\ln(z_{\text{ref}}/z_0)}, \tag{2.1}$$

where $v(z)$ denotes the wind speed at height z and the variable v_{ref} represents the wind speed at the reference height z_{ref}. The term z_0 refers to the surface roughness length and the variable z indicates the hub height.

The extrapolated wind speed $v(z)$ is then used to determine the power output from the turbine using the predefined power curve. This modeling approach ensures computational simplicity while remaining sufficiently realistic to capture the key physical behavior relevant to the optimization tasks described in this work.

2.2.2. Thermal Energy Storage

In this work, a reduced-order TES model is considered, based on the approach presented and verified in Walden et al. (2023), to enable efficient integration into the optimization framework. While high-fidelity TES models using spatial discretization (e.g., finite element methods) offer detailed insight, they lead to larger optimization problems that are more computationally intensive to solve. To overcome this, a lumped capacitance approach is applied, assuming a spatially uniform TES temperature. This simplification maintains the essential thermal dynamics of the storage system while significantly reducing computational effort. Notably, the TES is the only component in the system modeled dynamically, while all other components are treated as stationary. This modeling approach results in a quasi-stationary system, consistent with the system-level energy management framework. The TES in this study is concrete-based, selected for its high specific heat capacity, material availability, low investment and maintenance cost, and proven durability in high-temperature applications (100-400 °C).

The model consists of two equations. First, the heat transfer between the HTF and the TES is represented through an effectiveness-based formulation. The thermal effectiveness ϵ_{s} characterizes the efficiency of heat exchange between the HTF and the TES. Assuming that the storage mass is sufficiently large such that its temperature changes slowly compared to the HTF temperature, as it is commonly the case in industrial applications, the outlet temperature of the HTF T_{out} is computed from the inlet fluid temperature T_{in} and the current TES temperature T_{s} as:

$$\epsilon_{\text{s}} = \frac{T_{\text{in}} - T_{\text{out}}}{T_{\text{in}} - T_{\text{s}}}. \tag{2.2}$$

Second, the evolution of the TES temperature over time accounts for the energy exchange and thermal losses, and is given by:

$$\frac{\mathrm{d}T_\mathrm{s}}{\mathrm{d}t} = \frac{\dot{Q}_\mathrm{f} - \dot{Q}_\mathrm{loss}}{m_\mathrm{s} c_{p,\mathrm{s}}(T_\mathrm{s})}, \tag{2.3}$$

where $\dot{Q}_\mathrm{f}$ is the heat transferred from the fluid, m_s denotes the mass of the TES, $c_{p,\mathrm{s}}(T_\mathrm{s})$ is the temperature-dependent specific heat capacity of the storage material, and $\dot{Q}_\mathrm{loss}$ represents heat losses to the environment. Unlike the original formulation in Walden et al. (2023), where thermal losses are neglected, this study explicitly includes heat losses in the model to better reflect real operating conditions. The heat losses to the ambient are described by:

$$\dot{Q}_\mathrm{loss} = hA\big(T_\mathrm{s} - T_\mathrm{amb}\big), \tag{2.4}$$

where h denotes the heat transfer coefficient, A is the effective heat transfer surface area, and T_amb is the ambient temperature, assuming that the surface temperature of the storage is the same as the temperature inside the storage (Kyriakidis and Bähr, 2025). Under ideal conditions, the heat rejected by the fluid $\dot{Q}_\mathrm{f}$ equals the heat absorbed by the storage $\dot{Q}_\mathrm{s}$, which allows the calculation of the resulting temperature change of the storage medium as shown above. This reduced model, verified in Walden et al. (2023), offers a good balance between physical accuracy and computational effort, making it suitable for integration into energy management and optimization workflows.

2.2.3. Steam Generator

The modeling of the SG is based on the following energy balance that captures the thermal behavior of the component:

$$T_\mathrm{out} = T_\mathrm{in} - \frac{\dot{Q}}{\dot{m} c_{p,\mathrm{f}}}, \tag{2.5}$$

where T_out is the outlet temperature and T_in the inlet temperature of the HTF. The symbol $\dot{Q}$ represents the heat transferred to the steam. Moreover, the variable $\dot{m}$ denotes the mass flow rate of the HTF and $c_{p,\mathrm{f}}$ is the specific heat capacity of the HTF.

The SG was modeled using EBSILON and a dataset of steady-state operating points was generated. This dataset forms the basis for the derivation of surrogate models that express the inlet and outlet temperatures of the HTF as functions of the mass flow rate:

$$T_\mathrm{out} = a_0 + \frac{a_1}{\dot{m}}, \quad T_\mathrm{in} = \beta_0 + \frac{\beta_1}{\dot{m}}. \tag{2.6}$$

The parameters a_0, a_1, β_0, and β_1 are estimated by fitting the equations to the simulation data using ordinary least squares (OLS) regression, which minimizes the sum of squared differences between the surrogate model predictions and the simulation results. The resulting coefficients for the surrogate model are:

$$T_\mathrm{out} = 196.3 - \frac{188.4}{\dot{m}}, \quad T_\mathrm{in} = 201.92 + \frac{1{,}819.32}{\dot{m}}. \tag{2.7}$$

These models provide fast and continuous approximations of the SG thermal performance and are used within the optimization framework described later in this work.

2.2.4. Oil Pump

In the energy utility system considered in this thesis, it is essential to maintain the required and pressure levels throughout the system. This is achieved using four OPs placed along the oil circuit: one after the HTHP, one after the SG, and two associated with the TES – one for charging and one for discharging (see Fig. 2.1). Each pump is responsible for overcoming the pressure losses encountered in the subsequent component. The electric power of each OP is given by:

$$P_{\text{OP}} = \frac{1}{\eta_{\text{OP}}} \left(\frac{\dot{m}}{\dot{m}_{\text{nom}}} \right)^2 \frac{\dot{m} \Delta p}{\rho(T)}, \tag{2.8}$$

where $\dot{m}$ is the actual mass flow rate, Δp the pressure losses, and $\rho(T)$ the density of the thermal oil (Arias et al., 2009). Moreover η_{OP} denotes the pump efficiency and $\dot{m}_{\text{nom}}$ the nominal mass flow rate.

2.3. High-Temperature Heat Pump Modeling

This study examines a HTHP called CoBra operating on a closed-loop reverse Brayton cycle with air as the working fluid (Tran and Stathopoulos, 2025). The system, installed at a testing facility of the German Aerospace Center (DLR) in Cottbus, Germany, is designed to deliver supply temperatures of up to 260 °C. As seen in Fig. 2.2, it consists of a combination of compression, heat exchange, expansion, and recuperation. Key components include a two-stage radial compressor, a HTHX, a recuperator for internal heat recovery, a turbine, and a LTHX. Flow regulation is achieved using a three-way valve, which routes the working fluid through the recuperator when required, and a turbine bypass valve, which adjusts the flow resistance within the loop, allowing control over the operating points of the compressors.

The cycle starts with the compression of air, increasing its temperature. The heated air then passes through the HTHX, where thermal energy is extracted, followed by partial internal heat recovery via the recuperator. The air next expands through the turbine, resulting in a drop in both pressure and temperature. The cooled air then absorbs thermal energy in the LTHX, is preheated again in the recuperator, and returns to the compressor to complete the cycle. This configuration enables efficient thermal energy utilization, making the system well-suited for industrial applications demanding high supply temperatures.

It is important to note that, for steam generation, heat pumps based on the Rankine cycle with water as working fluid typically achieve higher efficiencies compared to reverse Brayton cycle systems. This is primarily because a Rankine cycle with water is specifically designed for producing and utilizing steam, efficiently leveraging phase change, whereas the Brayton cycle is optimized for gas-phase processes and requires an additional heat exchanger for steam production (Müller and Frechette, 2002). However, in this study, a Brayton-based heat pump is employed because detailed experimental data from the test facility

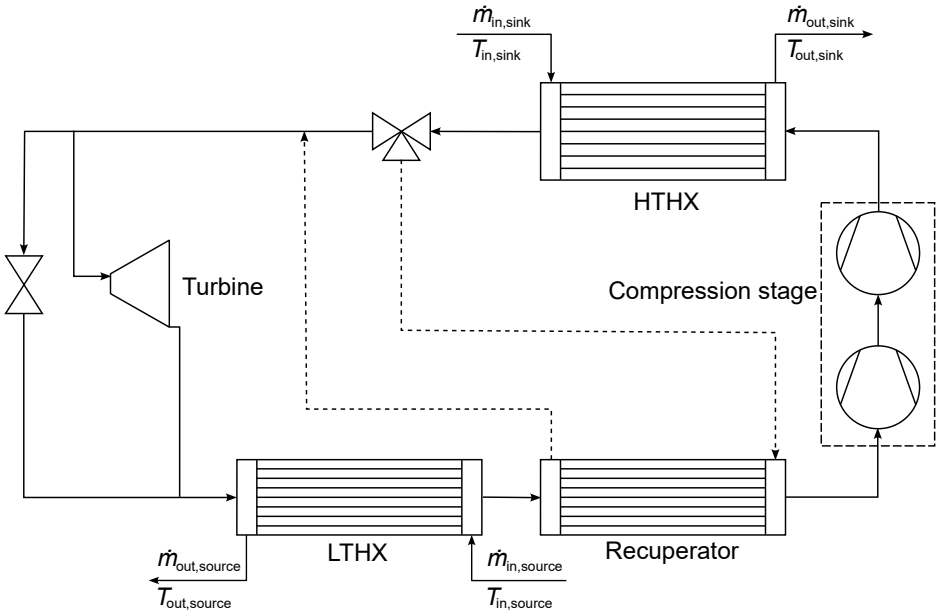

Figure 2.2: Flow sheet diagram of the recuperated HTHP.

are available, which is essential for developing and validating the proposed hybrid modeling methodology. The primary goal of this section is to develop a hybrid modeling methodology that combines simulation with experimental data, rather than to determine which heat pump cycle offers the greatest efficiency for a given application. The methodology developed here can be applied to RANKINE-based heat pumps in future work, provided that suitable experimental datasets become available. It should also be mentioned that the models are designed to predict key output-side metrics of the HTHP, such as the outlet temperatures on the sink and source side (i.e., $T_{\text{out,sink}}$ and $T_{\text{out,source}}$) and the corresponding mass flow rates (i.e., $\dot{m}_{\text{out,sink}}$ and $\dot{m}_{\text{out,source}}$), as functions of, for example, the inlet temperatures (i.e., $T_{\text{in,sink}}$ and $T_{\text{in,source}}$) and mass flow rates (i.e., $\dot{m}_{\text{in,sink}}$ and $\dot{m}_{\text{in,source}}$) on both sides.

One common strategy for modeling thermodynamic systems like the HTHP investigated in this study is to rely entirely on physical modeling. This approach remains the benchmark for accurately capturing the system behavior. However, as the component complexity increases, consisting of multiple subcomponents and governed by complex thermodynamic and control interactions, purely physics-based modeling becomes impractical. A simplified thermodynamic model for estimating the coefficient of performance (COP), often used in the literature, is useful for basic analysis, but such a model can not capture the full range of operating conditions, particularly the part-load behavior (Pieper et al., 2019). This limitation makes it difficult to use purely physical models for accurate modeling of the HTHP.

To address this limitation, surrogate modeling offers an appealing alternative. Rather than deriving detailed physics-based equations for each component, surrogate models approximate the component's behavior based on data. These models can be broadly categorized based on the data used to train them. Simulation-based surrogate models are widely used when experimental data is limited or expensive. These models are built on data generated from simulations and enable broad exploration of input conditions.

Popular techniques include Gaussian process regression (GPR), radial basis functions (RBF), multiple polynomial regression (MPR), artificial neural networks (ANNs), support vector regression (SVR), and random forest regression (RFR) (McBride and Sundmacher, 2019). For black-box optimization, nonparametric models (i.e., they do not assume any predefined relationship between input and output variables) such as GPR, RBF, and ANNs are often preferred due to their flexibility and high predictive accuracy. For applications that require equation-based outputs, simpler models like MPR or other nonlinear transformations (e.g., logarithmic, exponential) are more appropriate (Cozad et al., 2014). Despite their flexibility, simulation-based surrogate models have some limitations. Simulations rely on simplifying assumptions that may not hold under real operating conditions, leading to prediction errors between the surrogate predictions and actual system responses (Humbird et al., 2021).

On the other hand, experiment-based surrogate models rely on actual system measurements and can more accurately capture complex physical effects that simulations might overlook. However, this strategy introduces several challenges. Experimental data often contain noise, which can reduce the surrogate model's predictive performance if not properly managed (Letham et al., 2019). Moreover, collecting high-quality experimental data is generally costly, time-intensive, and may be constrained by logistical limitations – particularly in scenarios requiring high-fidelity measurements, extreme conditions, or specialized infrastructure (Humbird et al., 2021). The typically limited number of data points also results in sparse input space coverage (Humbird et al., 2021). These factors often make purely experimental surrogate modeling less feasible for many real-world applications.

To overcome the individual weaknesses of simulation- and experiment-based approaches, hybrid surrogate modeling combines the strengths of both. This strategy leverages the efficiency and scalability of simulations as well as the realism and high fidelity of experimental data, resulting in more robust surrogate models. By integrating both data sources, hybrid models address the limitations inherent in relying solely on simulations or experiments. A notable example of this technique is the use of neural networks with transfer learning, where a model initially trained on simulation data is fine-tuned using experimental observations (Humbird et al., 2021).

In this study, we introduce and evaluate several hybrid modeling techniques, including transfer learning and adaptive weighting. The latter method allows the model to dynamically adjust the influence of experimental and simulated data during training and prediction. We specifically examine how factors such as data availability, problem dimensionality, and discrepancies between simulation and experimental results affect the performance of each hybrid strategy. These approaches are first evaluated on two analytical benchmark functions to assess their accuracy and robustness across problems with varying degrees of nonlinearity. Then, they are applied to the HTHP, providing further insight into their practical effectiveness and overall robustness.

2.3.1. Hybrid Surrogate Modeling Strategies

This section presents the hybrid surrogate modeling strategies explored in this thesis, designed to combine the complementary advantages of simulation-based models and data-driven experimental approaches. These strategies build upon earlier results presented in Kyriakidis and Thapa (2025), with several improvements incorporated in the current work.

ANN with Transfer Learning: This strategy employs a transfer learning-based ANN framework to merge simulation and experimental data effectively (Humbird et al., 2021). In this approach, a base neural network is initially trained on a large, low-cost dataset generated via simulations. This pretraining stage helps the network capture general behaviors and patterns from the simulated system. Once this foundational knowledge is established, the model undergoes fine-tuning using a smaller, high-fidelity experimental dataset. This second phase adjusts the network's weights to better match the real-world system behavior, thereby mitigating discrepancies between simulated and actual responses.

Weighted Surrogate Models: This hybrid approach proposed creates a composite model by combining two separate surrogates: one trained on simulation data and the other on experimental data. The final output of the hybrid model, denoted as $\hat{y}_{\text{hyb}}$, is computed as follows:

$$\hat{y}_{\text{hyb}} = w_{\text{sim}}^{\text{norm}} \hat{y}_{\text{sim}} + w_{\text{exp}}^{\text{norm}} \hat{y}_{\text{exp}}, \tag{2.9}$$

where $w_{\text{sim}}^{\text{norm}} + w_{\text{exp}}^{\text{norm}} = 1$. These normalized weights represent the degree of confidence or availability associated with each data source. Both $\hat{y}_{\text{sim}}$ and $\hat{y}_{\text{exp}}$ can take the form of ANNs or MPR models, with the polynomial degree selected based on the problem's characteristics. A key challenge in this method is the appropriate setting of the weights. Simple hyperparameter tuning often proves insufficient, especially in cases where a model performs well on experimental validation but poorly on simulation data, or vice versa. This can occur due to differences in noise levels, sample sizes, or underlying biases in the data. To address this, we introduce an adaptive weighting scheme, developed in this thesis, that dynamically adjusts the weights based on the dataset characteristics, specifically the number of samples and observed variance. The weights are calculated as:

$$w_{\text{sim}} = \frac{N_{\text{sim}} \text{Var}_{\text{exp}}}{(N_{\text{sim}} + N_{\text{exp}})(\text{Var}_{\text{sim}} + \text{Var}_{\text{exp}})}, \quad w_{\text{exp}} = \frac{N_{\text{exp}} \text{Var}_{\text{sim}}}{(N_{\text{sim}} + N_{\text{exp}})(\text{Var}_{\text{sim}} + \text{Var}_{\text{exp}})}. \tag{2.10}$$

Here, N_{sim} and N_{exp} denote the number of simulation and experimental samples, while Var_{sim} and Var_{exp} represent the variance in each respective dataset. The normalized weights are determined as follows:

$$w_{\text{sim}}^{\text{norm}} = \frac{w_{\text{sim}}}{w_{\text{sim}} + w_{\text{exp}}}, \quad w_{\text{exp}}^{\text{norm}} = \frac{w_{\text{exp}}}{w_{\text{sim}} + w_{\text{exp}}}. \tag{2.11}$$

This formulation favors the model trained on the data source that is both more abundant and more reliable (i.e., less noisy), while still ensuring that both sources contribute proportionally to the final prediction. This

adaptive scheme enhances the robustness of the surrogate models and mitigates risks such as overfitting to noisy experimental data or ignoring useful information from simulations.

Unified Surrogate Model via Weighted Cost Function Minimization: This hybrid approach avoids the need for two separate surrogate models by training a single one that minimizes a weighted loss function over both simulation and experimental datasets. The loss function is defined as:

$$J(\hat{\boldsymbol{y}}) = \frac{1}{N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} (\hat{y}_i - y_{\text{sim},i})^2 + \lambda \frac{1}{N_{\text{exp}}} \sum_{j=1}^{N_{\text{exp}}} (\hat{y}_j - y_{\text{exp},j})^2. \tag{2.12}$$

In this formulation, $\hat{y}_i$ and $\hat{y}_j$ represent the model's predicted outputs for the i-th simulation sample and the j-th experimental sample, respectively, while $y_{\text{sim},i}$ and $y_{\text{exp},j}$ are the associated target values from the simulation and experimental datasets. The weighting factor λ controls the influence of the experimental data during training. To ensure a fair treatment of both data sources, λ is defined through an adaptive weighting scheme developed in this work:

$$\lambda = \frac{N_{\text{exp}} \text{Var}_{\text{sim}}}{(N_{\text{sim}} + N_{\text{exp}})(\text{Var}_{\text{sim}} + \text{Var}_{\text{exp}})}. \tag{2.13}$$

Its formulation considers both dataset size and variance. This hybrid model is conceptually inspired by the physics-informed neural networks (Raissi et al., 2019), which involve physical laws into the learning process. In contrast, our approach leverages simulation data for domain knowledge and refines the model using experimental observations through data-driven regularization, rather than explicit equation-based constraints. Another key distinction from physics-informed frameworks lies in the treatment of the weighting term. While physics-informed models typically use an optimized hyperparameter to balance data and physical losses, our approach introduces an adaptive weight that automatically adjusts based on the statistical properties of the datasets. By accounting for both the sample size and variability of each dataset, the method avoids bias toward the more dominant data source supporting balanced learning. The unified framework is especially advantageous when a single, compact model is desired for practical deployment or ease of interpretation.

2.3.2. Hybrid Modeling Validation on Test Functions

Before applying the proposed hybrid surrogate modeling approaches to the HTHP, their performance is first evaluated on two standard benchmark functions. This preliminary study examines the predictive accuracy, robustness (i.e., the ability to maintain high accuracy across varying dataset sizes and noise levels), and scalability (i.e., the ability to maintain high accuracy and robustness across vayring input dimensions) of the models on two test functions with varying nonlinearity and across different input dimensionalities. It also allows for a systematic investigation of how the surrogate performance is influenced by the quality and quantity of the available data, which are two key considerations in real-world applications.

The benchmark functions considered in this work are the ROSENBROCK and GOLDSTEIN–PRICE functions, selected for their distinct nonlinear characteristics (Hao et al., 2021). While the ROSENBROCK function exhibits moderate nonlinearity, the GOLDSTEIN–PRICE function is substantially more nonlinear according to Hao et al. (2021), posing a more difficult approximation task. To evaluate the scalability of the surrogate models, the input dimensionality is varied as $d = 2,\ 5,\ 10,\ 15,\ 20$. This range enables the assessment of how well the methods generalize as the input space dimensionality increases.

The d-dimensional ROSENBROCK function is defined as:

$$f_{\text{Rosen}}(\boldsymbol{x}) = \sum_{n=1}^{d-1} \left[100(x_{n+1} - x_n^2)^2 + (1 - x_n)^2\right], \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{2.14}$$

where $\boldsymbol{x}$ is the input vector of decision variables. The ROSENBROCK function features a narrow, curved valley, making it challenging for regression models, particularly as the dimensionality increases.

The original GOLDSTEIN–PRICE function is limited to two dimensions. To extend it to higher-dimensional settings, we define a composite formulation by applying the standard 2-D GOLDSTEIN–PRICE function to overlapping input pairs and summing the results. This captures the complex pairwise interactions across the input space and increases the function's nonlinearity with dimension. The generalized form is given as:

$$\begin{aligned} f_{\text{Gold}}(\boldsymbol{x}) = \sum_{n=1}^{d-1} &\left[1 + (x_n + x_{n+1} + 1)^2\Big(19 - 14x_n + 3x_n^2 - 14x_{n+1} + 6x_n x_{n+1} + 3x_{n+1}^2\Big)\right] \\ &\left[30 + (2x_n - 3x_{n+1})^2\Big(18 - 32x_n + 12x_n^2 + 48x_{n+1} - 36x_n x_{n+1} + 27x_{n+1}^2\Big)\right], \end{aligned} \tag{2.15}$$

with $\boldsymbol{x} \in \mathbb{R}^d$. To simulate realistic hybrid modeling scenarios, two types of data are generated: The experimental data, treated as the ground truth, are computed directly from the benchmark functions:

$$y_{\text{exp}} = f(\boldsymbol{x}). \tag{2.16}$$

The simulation data are constructed by adding zero-mean GAUSSIAN noise to the true function values to represent typical modeling inaccuracies:

$$y_{\text{sim}} = f(\boldsymbol{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{2.17}$$

where σ denotes the standard deviation of the noise and serves as a parameter for controlling the simulation fidelity. In this study, $f(\boldsymbol{x})$ denotes the benchmark function, which can be either the ROSENBROCK function $f_{\text{Rosen}}(\boldsymbol{x})$ or the generalized GOLDSTEIN–PRICE function $f_{\text{Gold}}(\boldsymbol{x})$. Another key variable across the test scenarios is the number of available simulation and experimental data points. By systematically varying both the quantity and quality (via noise level) of the training data, the evaluation framework enables an

in-depth analysis of the robustness and adaptability of the hybrid surrogate modeling approaches under diverse conditions.

Based on the two aforementioned benchmark functions, we now assess and compare the performance of the proposed hybrid surrogate modeling approaches across the defined scenarios. This analysis focuses on how each model performs under varying input dimensionalities, data availability, and noise levels using the Rosenbrock and Goldstein–Price functions. Hybrid model 1 utilizes ANN with transfer learning. Hybrid model 2 combines two separately trained surrogate models using a weighted combination. Two variants are employed in this hybrid modeling strategy. In the first variant, MPR serves as the base model for both surrogates, trained individually on simulation and experimental data, respectively. In the second variant, ANN is used as the base model for both surrogates. Hybrid model 3, in contrast, integrates both simulation and experimental data directly via a unified, weighted cost function. For both benchmark functions, third-degree polynomial regressions are used for hybrid model 2 (MPR variant) and hybrid model 3. In the case of hybrid model 3, this means that polynomial features up to third degree, including interaction terms, are employed in the weighted cost function (2.12), and the corresponding coefficients are estimated through optimization. The resulting surrogate model can thus be interpreted as a third-degree polynomial. The choice of third-degree polynomials for the surrogate models is motivated by the need to accurately capture the nonlinear relationships in the data while keeping the model size (i.e., the number of polynomial coefficients) and optimization time manageable. Preliminary results (not shown here) indicated that second-degree polynomials in both hybrid model 2 and hybrid model 3 were insufficient to represent the nonlinear patterns. Fourth-degree polynomials did not improve accuracy in either model but increased the number of polynomial coefficients. For both models, having more coefficients mainly raises the risk of overfitting. In hybrid model 3, the increased number of coefficients in the weighted cost function (2.12) results in longer optimization times.

Although analytical expressions for surrogate models are considered in this work in Chapters 5 and 6, making MPR more suitable for such purposes due to its ability to produce analytical equations, ANN-based models are also included in this benchmark comparison. This is motivated by their widespread use in the literature and their known ability to model complex nonlinear relationships, despite their limited analytical tractability.

Each hybrid model is evaluated in terms of prediction accuracy (measured by R^2), robustness (i.e., the ability to maintain high accuracy across varying experimental and simulation dataset sizes and noise levels), and scalability (i.e., the ability to maintain high accuracy and robustness across vayring input dimensions). The coefficient of determination R^2 is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2}, \tag{2.18}$$

where y_i and $\hat{y}_i$ denote the measured and predicted values, respectively, $\overline{y}$ represents the mean value and N is the number of data points considered in the performance evaluation (Tao et al., 2021). R^2 quantifies how well the model explains the variance in the data, with values closer to 1 (or 100 %) indicating a good fit and values close to 0 (or 0 %) a poor fit. For visualization, the left plots in Figs. 2.3-2.8 show the accuracy on experimental test data, and the right plots on simulation data. The unified objective function in hybrid model 3 is optimized using the local deterministic L-BFGS-B algorithm from SciPy (Virtanen et al., 2020) with default settings, as it provides fast convergence for differentiable functions by efficiently utilizing gradient information while supporting bound constraints. Although the stochastic differential evolution (DE) algorithm was also evaluated, it did not yield improved accuracy and significantly increased the computation time, so it is not further considered in this work. All models are implemented in Python 3.12 (Van Rossum and Drake, 2009) using TensorFlow (Abadi et al., 2015) for ANN and Sklearn (Pedregosa et al., 2011) for MPR. The ANN models used in hybrid models 1 and 2 consist of three hidden layers and underwent hyperparameter tuning via RandomizedSearchCV from Sklearn to achieve optimal performance. Moreover, the computations in this study run on an Intel(R) Core(TM) i7-8665U CPU. To ensure stable and efficient training, all input and output variables are normalized to the [0, 1] range across the models considered.

To ensure the reliability of the results and to mitigate overfitting (i.e., when models perform well on training data but poorly on unseen data), cross-validation (CV) is employed during model evaluation. Among the most widely used CV techniques is the k-fold CV method (Berrar, 2019). In this approach, the dataset is divided into k equally sized subsets, or folds. The model is trained on $k-1$ folds and validated on the remaining one, with this process repeated k times such that each fold serves as the validation set exactly once. The final performance metrics are then obtained by averaging the results across all folds. In this study, a 5-fold CV procedure is implemented using Sklearn. The results presented in this section and in Section 2.3.3 are based on this CV strategy.

Figs. 2.3 and 2.4 display the R^2 scores across different input dimensions d with 5,000 simulation data points and a varying number of experimental data for both benchmark functions. In the first part of the analysis, the number of simulation data points was fixed at 5,000 to provide a sufficiently rich representation of the underlying functions while keeping computational requirements manageable. Hybrid model 1 and hybrid model 2 with ANN achieve high accuracy (up to 99.99 % accuracy) in low-dimensional settings (i.e., $d \leq 5$), regardless of the number of experimental data points. However, in higher dimensions (i.e., $d > 5$), both models exhibit reduced accuracy unless more experimental data are available. For the Rosenbrock function, the accuracy on the experimental dataset (based on 50 experimental samples) drops to 76.97 % for hybrid model 1 and 81.71 % for hybrid model 2 with ANN, while on the simulation dataset it decreases to 85.93 % and 81.78 %, respectively. Similarly, for the Goldstein-Price function, the accuracy on the experimental dataset falls to 50.55 % for model 1 and 47.82 % for model 2 with ANN, whereas the accuracies on the simulation dataset remain higher at 76.23 % and 77.63 %, respectively. The lower accuracies observed in the Goldstein–Price function are primarily due to its higher nonlinearity

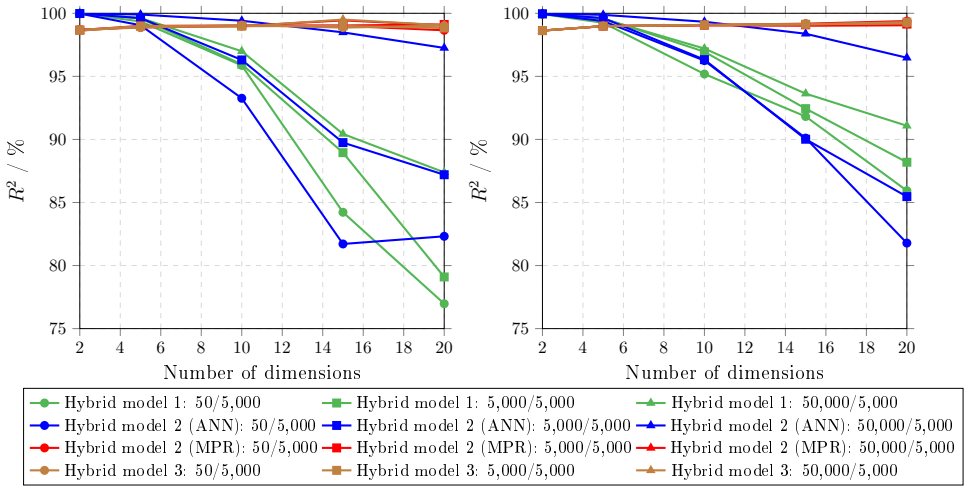

Figure 2.3: R^2 values of the hybrid models evaluated on experimental (**left**) and simulation (**right**) datasets as a function of the number of dimensions for different number of experimental points, exemplified for the Rosenbrock function with a noise level of 0.8. The legend indicates the number of data points considered: the first number refers to experimental data and the second to simulation data.

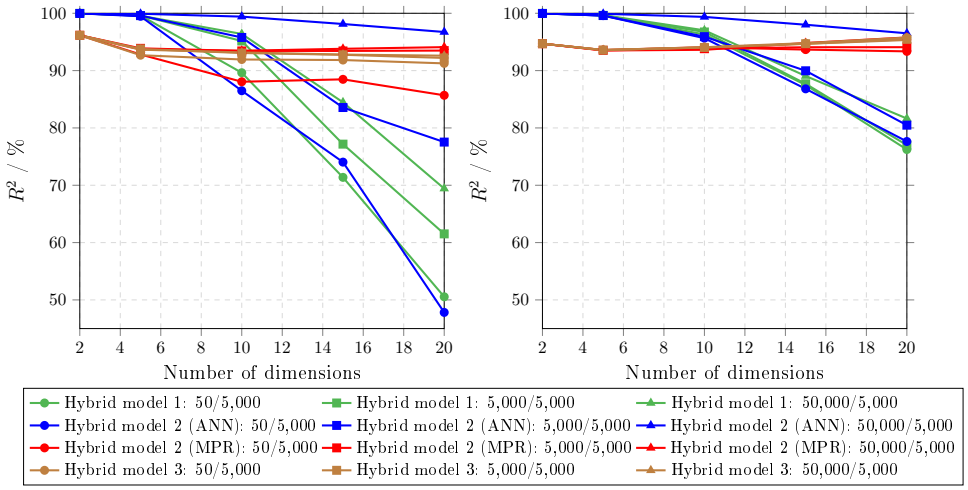

Figure 2.4: R^2 values of the hybrid models evaluated on experimental (**left**) and simulation (**right**) datasets as a function of the number of dimensions for different number of experimental points, exemplified for the Goldstein-Price function with a noise level of 5. The legend indicates the number of data points considered: the first number refers to experimental data and the second to simulation data.

compared to the Rosenbrock function, making it more challenging for the surrogate models to approximate effectively with limited data. Hybrid model 2 with MPR shows stable performance ($\approx$ 99 % accuracy on average) in both experimental and simulation dataset across the dimensions for the Rosenbrock function, but for the more nonlinear Goldstein-Price function, its accuracy in the experimental test data drops below 90 % when only 50 experimental samples are available, starting from 10 dimensions. With 5,000 or more experimental points, the model maintains high accuracy (94.24 % accuracy on average). In the simulation set, however, the accuracy of hybrid model 2 for the Goldstein-Price function remains higher, at 94.49 %. Hybrid model 2 with MPR outperforms the hybrid model 2 with ANN when only

limited experimental data are available because ANNs, having many parameters, require more data to accurately learn the nonlinear mappings, whereas MPR, having fewer parameters (polynomial coefficients), can capture the relationships with fewer data points. It should be mentioned that in the results of this section, the models are often observed to perform better on the simulation test data than on the experimental data, in cases where the number of simulation points is larger than that of experimental points. In such scenarios, the simulation data dominate the training process and help stabilize the model performance.

Hybrid model 3 exhibits a similar accuracy trend to hybrid model 2 with MPR for the Rosenbrock function in both the experimental and simulation sets, as well as for the Goldstein-Price function in the simulation set (see Figs. 2.3 and 2.4). However, it outperforms hybrid model 2 with MPR in terms of robustness in the experimental set of the Goldstein-Price function, particularly when experimental data is scarce (e.g., 50 samples). More precisely, hybrid model 3 achieves an average accuracy of approximately 93 % across the considered dimensions, with values ranging from 91.26 % to 96.16 % depending on the number of experimental data points and dimensionality. In comparison, hybrid model 2 reaches around 90 % accuracy under the same conditions, with values varying between 85.69 % and 96.16 %. This indicates that the integrated training strategy of hybrid model 3, where simulation and experimental data are jointly optimized via a weighted cost function, better handles the strong nonlinearities of the Goldstein-Price function, particularly when limited experimental data is available, as it is often the case in real-world scenarios. Unlike hybrid model 2, which combines two independently trained models, hybrid model 3 learns complementary patterns through a unified loss, leading to improved extrapolative generalization. It is worth noticing that both hybrid model 2 with MPR and hybrid model 3 show improved accuracy with increasing dimensionality, suggesting that higher-dimensional input spaces make the function easier to approximate and enable the models to leverage a richer feature space for improved learning.

Figs. 2.5 and 2.6 display the R^2 values across varying input dimensions d for both benchmark functions, using a fixed set of 50 experimental samples while varying the number of simulation data points. Using 50 experimental samples, rather than a much larger number, reflects realistic scenarios where only a limited number of costly or time-consuming experiments are feasible, and is consistent with the sample size considered in CoBra (see Section 2.3.3). As expected, hybrid models 1 and 2 with ANN benefit significantly from an increased number of simulation data. For example, the accuracy on the simulation data rises from 78.14 % for model 1 and 69.15 % for model 2 with ANN on the 15-D Rosenbrock function, and from 34.16 % for model 1 and 40.85 % for model 2 with ANN on the 20-D Goldstein-Price function, to 99.99 %. This highlights their ability to generalize better when more simulation data is available. In addition, hybrid model 3 consistently demonstrates higher accuracy and robustness than hybrid model 2 with MPR in the experimental test set, even under limited experimental data conditions. More precisely, in the 20-D case, the accuracy of hybrid model 3 across varying amounts of simulation data ranges from 99.02 % to 99.98 % in the Rosenbrock function, and from 91.19 % to 99.99 % in the Goldstein–Price function. In contrast, hybrid model 2 with MPR achieves a lower and wider accuracy range – from

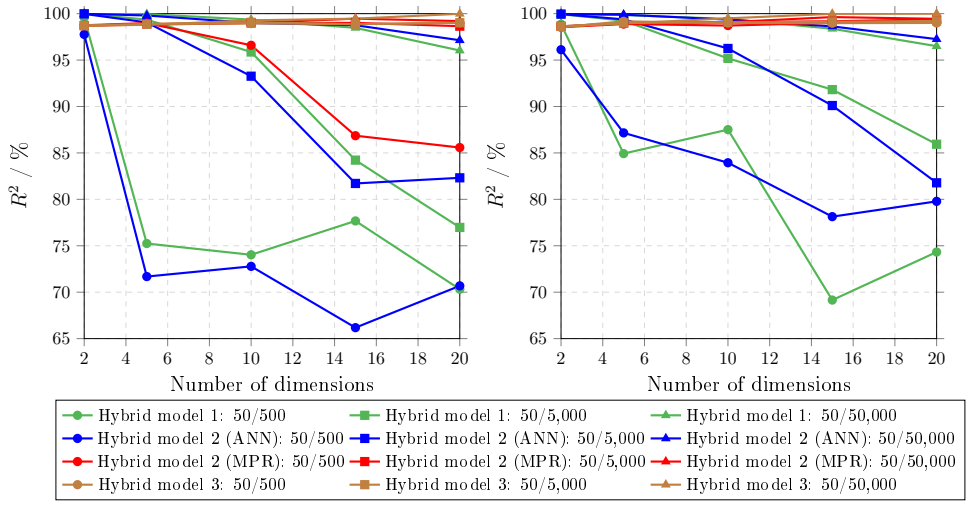

Figure 2.5: R^2 values of the hybrid models evaluated on experimental (**left**) and simulation (**right**) datasets as a function of the number of dimensions for different number of simulation points, exemplified for the Rosenbrock function with a noise level of 0.8. The legend indicates the number of data points considered: the first number refers to experimental data and the second to simulation data.

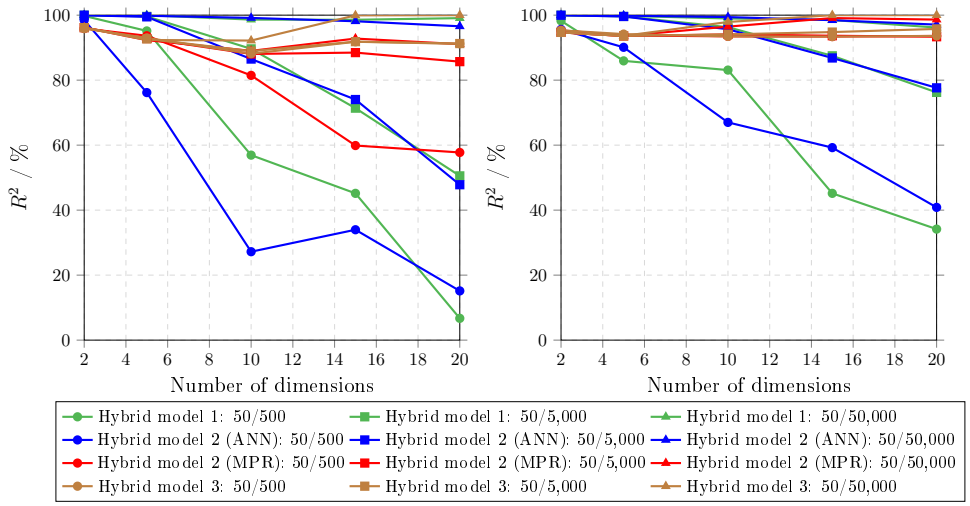

Figure 2.6: R^2 values of the hybrid models evaluated on experimental (**left**) and simulation (**right**) datasets as a function of the number of dimensions for different number of simulation points, exemplified for the Goldstein-Price function with a noise level of 5. The legend indicates the number of data points considered: the first number refers to experimental data and the second to simulation data.

85.58 % to 99.19 % for the Rosenbrock function, and from 57.75 % to 91.08 % for the Goldstein–Price function. In the simulation test set, hybrid model 3 achieves comparable accuracy to hybrid model 2 with MPR. Specifically, for the Rosenbrock function, the accuracy of both models ranges from 98.54 % to 99.99 %, and for the Goldstein–Price function, from 93.40 % to 99.99 %. These results highlight the stable performance and scalability of hybrid model 3 across different simulation data sizes.

Figs. 2.7 and 2.8 explore the impact of different noise levels between simulation and experimental data, using a dataset of 50 experimental and 5,000 simulation points. It should be mentioned that different noise

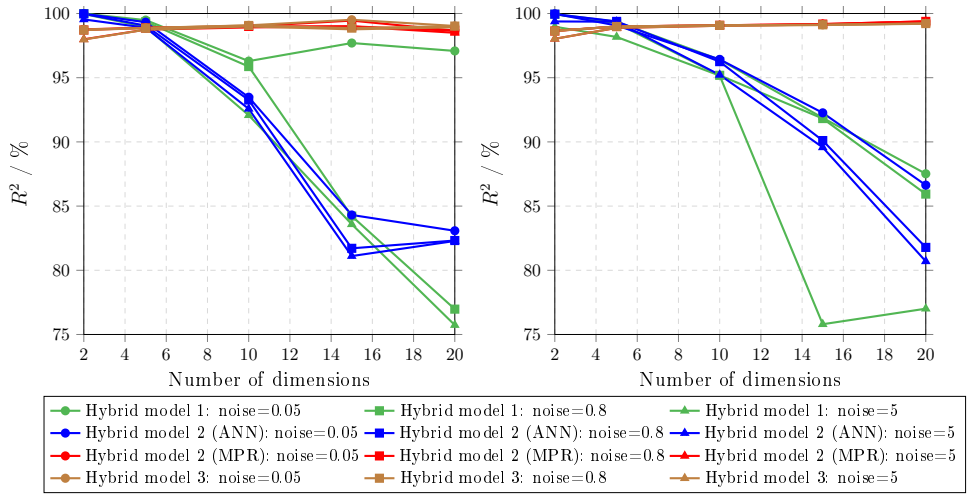

Figure 2.7: R^2 values of the hybrid models evaluated on experimental (**left**) and simulation (**right**) datasets as a function of the number of dimensions for different noise levels, exemplified for the Rosenbrock function with 50 experimental and 5,000 simulation points.

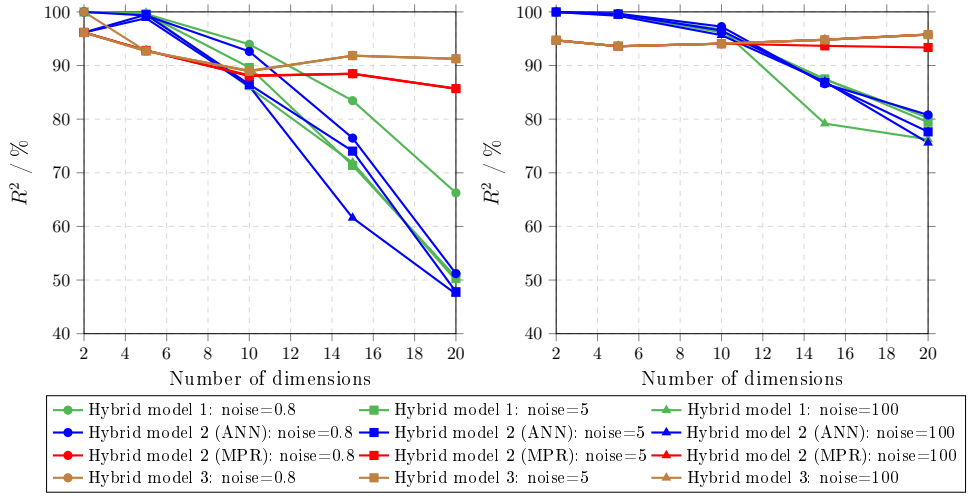

Figure 2.8: R^2 values of the hybrid models evaluated on experimental (**left**) and simulation (**right**) datasets as a function of the number of dimensions for different noise levels, exemplified for the Goldstein-Price function with 50 experimental and 5,000 simulation points.

scales were applied to the Rosenbrock and Goldstein–Price functions, with smaller noise magnitudes chosen for the Rosenbrock case due to its much narrower range of y-values. This adjustment ensures that the relative influence of noise remains comparable across both benchmarks. As seen in these figures, hybrid model 3 exhibits strong robustness to noise, maintaining high prediction accuracy across all noise levels tested. On average, it achieves 98.88 % and 99.01 % accuracy in the Rosenbrock function in the experiment and simulation test sets, respectively, and 92.56 % and 94.59 % in the Goldstein–Price function. In contrast, hybrid models 1 and 2 with ANN show notable performance degradation as the noise increases, especially in higher-dimensional settings. More precisely, they achieve performances in the Rosenbrock function down to 75.73 % in the experimental set and 77.01 % in the simulation

set, and for the Goldstein-Price function down to 47.82 % in the experimental set and 75.60 % in the simulation set. Notably, hybrid model 2 using MPR achieves performance close to that of model 3, further demonstrating its reliability under noisy conditions. The largest deviations are observed for the Goldstein–Price function in higher dimensions, with errors reaching up to 5.58 % in the experimental test set and 2.41 % in the simulation test set.

In addition to accuracy, robustness, and scalability, the computational effort of the models was also evaluated using the two functions with 50 experimental and 5,000 simulation data points as an example. In low-dimensional settings such as 2-D, hybrid model 2 with MPR and hybrid model 3 exhibit very low CPU time due to the simplicity of the polynomial regression and the efficiency of the optimization process of (2.12) ($\approx$ 0.04 seconds for both models in the Rosenbrock and 0.12 seconds in the Goldstein-Price function). Neural networks, on the other hand, require longer training times even in these small dimensions (107 seconds for hybrid model 1 and 65 seconds for hybrid model 2 with ANN in the Rosenbrock, and 119 and 77 seconds, respectively, in the Goldstein-Price function excluding the CPU time for hyperparameter tuning). However, as the dimensionality increases, particularly beyond 10-D (e.g., 20-D), the CPU time of hybrid model 3 increases significantly and can exceed that of the ANN-based models (without considering the CPU time for hyperparameter tuning in the ANN-based models), leading to a training time of 598 seconds in the Rosenbrock function and of 753 seconds in the Goldstein-Price function. Meanwhile, hybrid model 1 and hybrid model 2 with ANN and MPR require 149, 115, and 2 seconds, respectively, in the Rosenbrock function, and 193, 157, and 3 seconds, respectively, in the Goldstein-Price function. This is primarily due to the increased computational effort required to optimize the weighted cost function in higher-dimensional input spaces using the L-BFGS-B algorithm. While this model offers strong accuracy and robustness, its computational efficiency may be improved in future work by exploring more efficient optimizers, implementing parallelization, or leveraging more powerful computing hardware. For the ANN-based models, the CPU time does not increase as significantly with dimensionality as it does for hybrid model 3. This is mainly due to the use of early stopping, which limits the number of training iterations once convergence is reached.

Another key point is how the hybrid models with self-adaptive weights perform compared to cases where the weights are manually fine-tuned as hyperparameters. This comparison is displayed in Figs. 2.9 and 2.10, which show the performance in the Rosenbrock and Goldstein–Price functions, respectively. Each figure presents the results for hybrid model 3 in both experimental and simulation test sets across a range of manually chosen weight values, alongside the performance achieved with the self-adaptive weight calculated using (2.13). The three subplots correspond to different scenarios of data availability: 50 experimental with 500,000 simulation data points (left), 5,000 experimental with 5,000 simulation data points (middle), and 500,000 experimental with 50 simulation data points (right). In all scenarios, the self-adaptive weighting approach yields balanced and consistent performance across both test sets. It effectively avoids regions where the performance drastically diverges, especially where the excessive reliance on either simulation or experimental data harms the extrapolative generalization. The adaptive

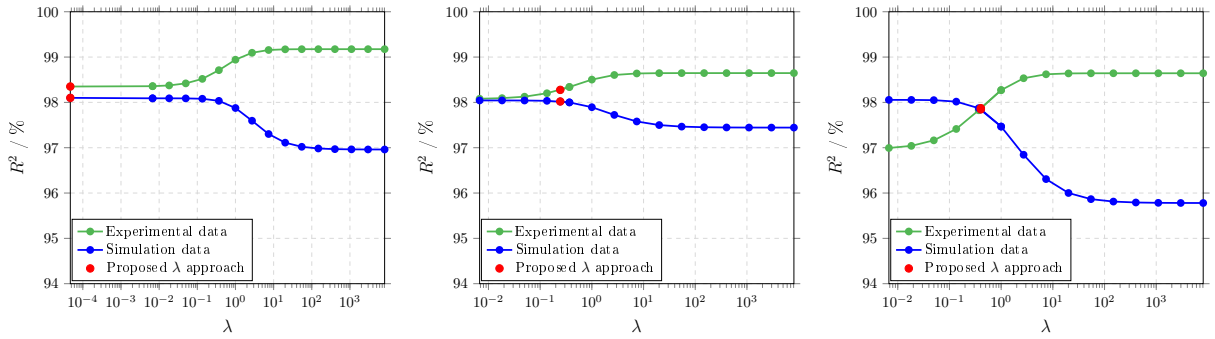

Figure 2.9: R^2 values of the hybrid model 3 evaluated on simulation and experimental datasets as a function of the weighting parameter λ, exemplified for the 2-D Rosenbrock function with a noise level of 5. Each subplot corresponds to a different data distribution scenario: (**left**) 50 experimental vs. 500,000 simulation points, (**middle**) 5,000 experimental vs. 5,000 simulation points, and (**right**) 500,000 experimental vs. 50 simulation points. The red point indicates the λ value computed using the proposed formula (2.13).

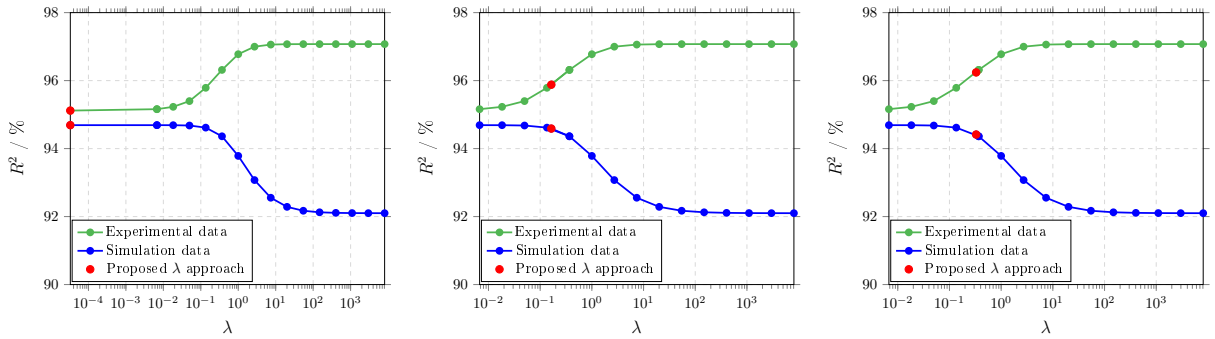

Figure 2.10: R^2 values of the hybrid model 3 evaluated on simulation and experimental datasets as a function of the weighting parameter λ, exemplified for the 2-D Goldstein-Price function with a noise level of 5. Each subplot corresponds to a different data distribution scenario: (**left**) 50 experimental vs. 500,000 simulation points, (**middle**) 5,000 experimental vs. 5,000 simulation points, and (**right**) 500,000 experimental vs. 50 simulation points. The red point indicates the λ value computed using the proposed formula (2.13).

method adjusts the weighting dynamically, based on both data variance and sample size. A similar trend is also observed for hybrid model 2 using adaptive weighting, which employs the same underlying concept but extends it to two self-adaptive weights instead of one. However, these results are not shown here, as they follow the same pattern and do not provide additional insights beyond those already illustrated for hybrid model 3, and are therefore omitted for conciseness.

Tables 2.1 and 2.2 further highlight the effectiveness of hybrid modeling by comparing the performance of hybrid model 3 to that of the conventional MPR model trained exclusively on simulation data. The results, presented for the 20-D Goldstein–Price function with a noise level of 5, underscore the drawbacks of relying on a single data source. Specifically, MPR trained only on simulation data achieves high accuracy on the simulation test set (95.81 %) but generalizes poorly to experimental data, reaching only 85.32 % accuracy with 50 experimental samples and 89.44 % with 500. In contrast, hybrid model 3 achieves balanced and consistently high performance across both test sets, with 91.33 % on experimental and 95.81 % on simulation data with 50 experimental samples (Table 2.1), and improving to 93.73 %

Table 2.1: Comparison of R^2 for MPR trained solely on simulation data and the hybrid model 3. The models are evaluated on simulation and experimental datasets for the 20-D GOLDSTEIN-PRICE function with 50 experimental and 5,000 simulation data points (noise=5).

Models	R^2 (exp)	R^2 (sim)
MPR (sim)	85.32 %	95.81 %
Hybrid model 3	91.33 %	95.81 %

Table 2.2: Comparison of R^2 for MPR trained solely on simulation data and the hybrid model 3. The models are evaluated on simulation and experimental datasets for the 20-D GOLDSTEIN-PRICE function with 500 experimental and 5,000 simulation data points (noise=5).

Models	R^2 (exp)	R^2 (sim)
MPR (sim)	89.44 %	95.81 %
Hybrid model 3	93.73 %	95.81 %

on experimental data when 500 experimental points are used (Table 2.2). These results demonstrate the robustness of hybrid model 3 and its capacity to effectively integrate complementary information from both data sources. The increase in MPR accuracy on experimental data (from 85.32 % with 50 points to 89.44 % with 500 points) arises because the larger experimental dataset covers a broader portion of the input space, including regions that reflect patterns captured in the simulation training data. This allows the model, although trained solely on simulation data, to demonstrate higher accuracy on experimental samples that lie within these regions.

2.3.3. Hybrid Modeling Validation on HTHP

This section evaluates the performance of the proposed hybrid surrogate modeling methods applied to the HTHP. Each hybrid approach involves developing two individual surrogate models: one estimates the outlet temperature on the sink side, which indicates the temperature available for industrial processes, based on four input variables: inlet temperature on the source side, inlet mass flow and temperature on the sink side, and motor speed of the two-stage compressor. The second model predicts the electric power consumption using the same set of inputs. These two output variables – outlet temperature and electric power usage – were also considered in the HTHP of Walden et al. (2023), along with the same input variables, except for the motor speed. In their work, the HTHP employed a one-stage compressor, and the rotational shaft speed was used as the fourth input variable. In contrast, the HTHP investigated in this study incorporates a two-stage compressor, and the motor speed of this compressor is used as the corresponding input variable. Additionally, Walden et al. (2023) built their surrogate models based solely on simulation data, whereas our surrogate models are built based on both experimental and simulated data. Furthermore, Walden et al. (2023) included a third surrogate model for the outlet temperature on the source side, also based on the same four inputs as the other two surrogate models. However, since no

utilization strategy is currently defined for the cold side in this work, that surrogate model is reused, and no new hybrid model is developed for it. For consistency, we adopt this surrogate model from Walden et al. (2023), replacing the rotational shaft speed with the motor speed as the fourth input variable.

To evaluate the hybrid surrogate models, both experimental and simulated data are utilized. A total number of 26 experimental data points were obtained from the test facility, while simulation data was produced using a detailed MODELICA 4.0.0 model (Modelica, 2025). The simulation dataset was built through a one-at-a-time sensitivity analysis, where each input parameter was varied independently across a specified range, while the other four inputs remained constant. This method allows the system's response to individual parameter changes and captures nonlinear relationships between inputs and outputs. All output data correspond to steady-state conditions, meaning the variables have reached equilibrium and do not change over time. Although MODELICA is primarily used for transient simulations, only the steady-state results are considered here, following the same approach as in the experiments. To assess how the quantity of data affects the model performance, simulation dataset sizes of 50, 500, 5,000, and 25,000 samples were considered. Similar to the test function analysis, all input and output variables were normalized to the [0, 1] interval to ensure stable and efficient training. For hybrid model 2 using MPR and for hybrid model 3, third-degree MPR models were trained for both outlet temperature and electric power predictions.

Figs. 2.11 and 2.12 present the R^2 scores for the outlet temperature and electric power predictions, respectively, across varying the number of simulation data. The models were evaluated on both the simulated and experimental datasets. In both figures, hybrid model 3 consistently shows strong predictive performance in the simulated test set (99.90 % accuracy on average) – even when only 50 simulation samples are used. This demonstrates the effectiveness of its unified training approach, which integrates both data sources using a single weighted loss function. In contrast, the other hybrid models need more simulation data to reach comparable accuracy; at least 500 samples for the outlet temperature surrogate model and 5,000 for the electric power model. This also applies to hybrid model 2, which trains the experimental and simulation surrogates separately, and if one performs poorly (e.g., the experiment-based surrogate), it decreases the overall prediction accuracy.

When evaluated on the experimental data, hybrid model 3 again maintains high accuracy across all simulation dataset sizes (see left side of Figs. 2.11 and 2.12), achieving 92.19 % and 91.96 % accuracy on average for the outlet teperature and electric power, respectively, and confirming its robustness and ability to generalize. The ANN-based methods – hybrid model 1 and hybrid model 2 with ANN – show a gradual improvement in accuracy for predicting the outlet temperature as more simulation data become available, starting from 11.97 % and 17.80 % and reaching 91.57 % and 91.38 % accuracy, respectively, at 25,000 simulation samples. For the electric power prediction, the models start from 10.47 % and 9.76 % and achieve accuracies of 90.10 % and 89.70 %, respectively, at 25,000 simulation samples. This reflects their strong ability to learn from large datasets but also their dependency on the size of the dataset. Hybrid model 2 with MPR follows a similar pattern to hybrid model 3 over the number of simulation data points, but with lower accuracy (85.14 % and 90.63 % accuracy on average for the outlet temperature and

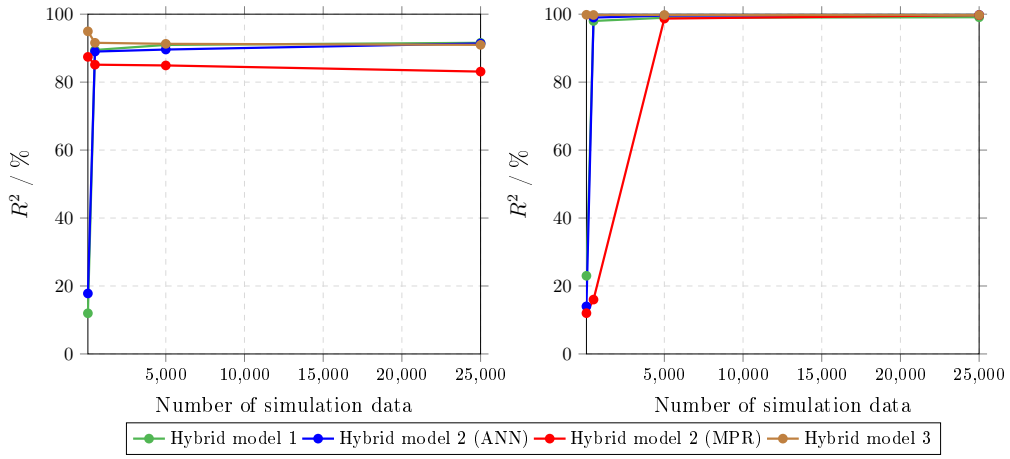

Figure 2.11: R^2 values of the hybrid models for predicting the outlet temperature of the HTHP as a function of the number of simulation points. The models are evaluated on experimental (**left**, 26 points) and simulation (**right**) datasets.

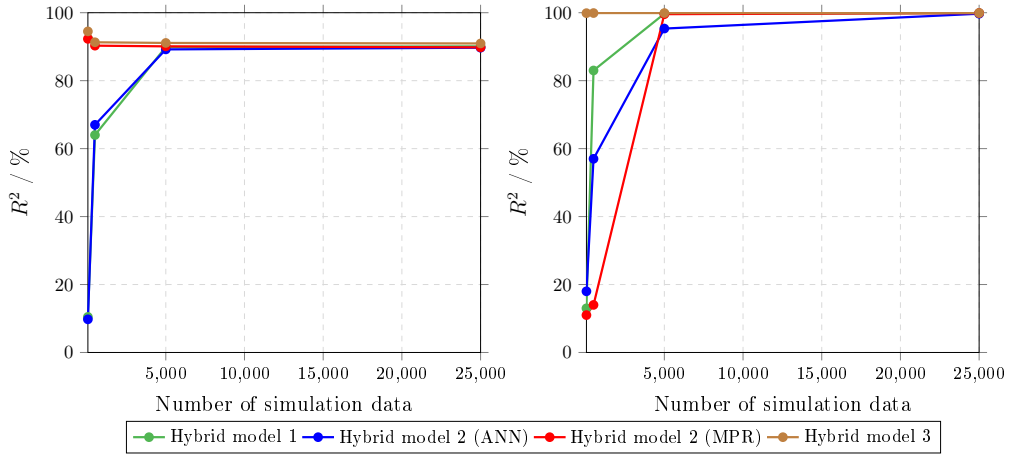

Figure 2.12: R^2 values of the hybrid models for predicting the electric power consumption of the HTHP as a function of the number of simulation points. The models are evaluated on experimental (**left**, 26 points) and simulation (**right**) datasets.

electric power, respectively). Although its accuracy in the simulation test set is low for small data sizes, its performance in the experimental test set is higher since both surrogate models (for simulation and experiments) are well trained in this setting. This contrast highlights that the separate training approach used in hybrid model 2 can be especially sensitive when one of the surrogate models performs poorly. For hybrid model 2 with MPR and hybrid model 3, the highest experimental accuracy for outlet temperature and power prediction is achieved with only 50 simulation samples. As the number of simulation points increases, the accuracy of these two models evaluated on the experimental test set (see left side of Figs. 2.11 and 2.12) decreases compared to using 50 simulation points. This happens due to the adaptive weighting, which assigns more importance to simulation data as its volume grows. Since discrepancies exist between the simulation and experimental datasets, this shift in weighting leads to reduced accuracy when tested

Table 2.3: Comparison of R^2 for MPR trained solely on simulation data and hybrid model 3. The models are evaluated on simulation and experimental datasets for the outlet temperature prediction of the HTHP with 26 experimental and 50 simulation data points.

Models	R^2 (exp)	R^2 (sim)
MPR (sim)	83.2 %	99.8 %
Hybrid model 3	95.0 %	99.8 %

on the experimental data. Again, as shown in the test cases, Table 2.3 illustrates the advantage of using a hybrid model rather than relying solely on a simulation-based model, as it can better capture relevant patterns from both experimental and simulation datasets.

In terms of computation time, hybrid model 3 takes 0.38 seconds to train with 50 simulation samples, increasing to 8 seconds for 25,000 samples. By comparison, for 50 to 25,000 simulation samples, hybrid model 1 requires 43–110 seconds, hybrid model 2 with ANN needs 25–75 seconds (excluding ANN hyperparameter tuning time), and hybrid model 2 with MPR takes 0.02–0.30 seconds. Across all scenarios, hybrid model 2 with MPR consistently delivers the fastest training times. However, the computational effort of hybrid method 3 observed in the HTHP case is within the acceptable time limits for energy management applications in this thesis, particularly when online updates are required for each new experimental data point. As already noted in Chapter 1, a maximum total CPU time of approximately 60 seconds per RHA iteration is defined for determining an optimal setpoint. Nonetheless, there is potential for further reduction in computational effort, as discussed in Section 2.3.2, which will be addressed in future studies.

Based on these findings, hybrid model 3 is considered for developing surrogate models for both outlet temperature and electric power of the HTHP. Across all investigations, hybrid model 3 consistently achieved high accuracy, robustness, and extrapolative generalization – especially in higher-dimensional input spaces or when simulation and experimental data differ. Another advantage of hybrid model 3 is that the analytical equations can be easily derived. After the optimization of (2.12), where the model coefficients are determined, the resulting model can be explicitly expressed in mathematical form, which is required in this work. Since the HTHP used in this study operates at the kW scale, smaller than the MW-scale system investigated in Walden et al. (2023), the HTHP here has been scaled accordingly, assuming that the system's performance characteristics remain unchanged at higher power levels. This is a common assumption in surrogate modeling and preliminary system analysis; however, experimental studies have shown that scale-up can affect component efficiencies and heat losses, potentially leading to deviations in absolute performance at industrial scale (Jeßberger et al., 2024). While this effect is acknowledged, it is not the focus of this thesis and will be investigated in future work.

2.4. Resulting Optimization Problem

After introducing the individual components and their models, this section integrates all elements into a unified system model. The optimal operation of the complete renewable energy utility system is formulated as a discrete, multi-period optimization problem by discretizing the continuous time interval $[t_0, t_n]$. Therefore, a uniformly distributed time grid with the time step $\Delta t = t_k - t_{k-1}$ for $k = 1, \dots, n$ is introduced, so that the piecewise constant functions are controlled only at the discrete time points, i.e., $T_I^k := T_I(t_k)$. Here, t_n denotes the last time point of the optimization horizon, while n corresponds to the number of discrete time intervals considered in the model. The total duration of the horizon is given by $t_n - t_0$. The optimization problem in discrete setting, which is initially introduced in Walden et al. (2023) and further extended in this work, is defined as follows based on Fig. 2.13:

$$\min \; f(\boldsymbol{P}_{\text{total}}) = \sum_{k=1}^{n} P_{\text{total}}^k g_{\text{pr,grid}}^k \Delta t + \max(P_{\text{total}}^k) C, \tag{2.19}$$

subject to:

$$P_{\text{grid}}^k + P_{\text{WT}}^k = 3F_{\text{HTHP}}\big(T_{\text{I}}^k, \dot{m}_{\text{I}}^k, T_{\text{III}}^k, R^k\big), \tag{2.20}$$

$$T_{\text{II}}^k = F_{\text{HTHX}}\big(T_{\text{I}}^k, \dot{m}_{\text{I}}^k, T_{\text{III}}^k, R^k\big), \quad T_{\text{IV}}^k = F_{\text{LTHX}}\big(T_{\text{I}}^k, \dot{m}_{\text{I}}^k, T_{\text{III}}^k, R^k\big), \tag{2.21}$$

$$T_2^k = T_{\text{II}}^k\big(1-\beta_1^k\big) + T_1^k \beta_1^k, \quad T_{\text{I}}^k = T_3^k\big(1-\beta_2^k\big) + T_4^k \beta_2^k, \tag{2.22}$$

$$T_2^k = 201.92 + \tfrac{1{,}819.32}{3\dot{m}_{\text{II}}^k}, \quad T_3^k = 196.3 - \tfrac{188.4}{3\dot{m}_{\text{I}}^k}, \tag{2.23}$$

$$\dot{Q}_{\text{s,ch}}^k = 3\dot{m}_{\text{II}}^k c_{p,\text{f}}\big(T_{\text{II}}^k, T_1^k\big)\big(T_{\text{II}}^k - T_1^k\big)\beta_1^k, \quad \dot{Q}_{\text{s,dch}}^k = 3\dot{m}_{\text{I}}^k c_{p,\text{f}}\big(T_3^k, T_4^k\big)\big(T_4^k - T_3^k\big)\beta_2^k, \tag{2.24}$$

$$\dot{Q}_{\text{loss}}^k = hA\big(T_{\text{s}}^k - T_{\text{ambient}}^k\big), \quad \beta_1\beta_2 \le \gamma, \tag{2.25}$$

$$T_1^k = T_{\text{II}}^k - \epsilon_{\text{ch}}\big(T_{\text{II}}^k - T_{\text{s}}^{k-1}\big), \quad T_4^k = T_3^k - \epsilon_{\text{dch}}\big(T_3^k - T_{\text{s}}^{k-1}\big), \tag{2.26}$$

$$T_{\text{s}}^k = T_{\text{s}}^{k-1} + \tfrac{\dot{Q}_{\text{s,ch}}^k - \dot{Q}_{\text{s,dch}}^k - \dot{Q}_{\text{loss}}^k}{m_{\text{s}} c_{p,\text{s}}(T_{\text{s}}^k, T_{\text{s}}^{k-1})} \Delta t, \tag{2.27}$$

$$P_{\text{total}}^k = P_{\text{grid}}^k + P_{\text{OP,total}}^k, \quad P_{\text{OP,total}}^k = \frac{1}{\eta_{\text{OP}}} \sum_{i=1}^{4} \left(\frac{\dot{m}_i^k}{\dot{m}_{\text{nom},i}^k}\right)^2 \frac{\Delta p_i \dot{m}_i^k}{\rho_i(T_i^k)}, \tag{2.28}$$

$$T_{\text{s}}^0 = \widetilde{T}_0, \quad \frac{T_{\text{s}}^n - T_{\text{s,min}}}{T_{\text{s,max}} - T_{\text{s,min}}} \le 0.8, \quad \frac{T_{\text{s}}^n - T_{\text{s,min}}}{T_{\text{s,max}} - T_{\text{s,min}}} \ge 0.2, \tag{2.29}$$

$$|T_{\text{II}}^k - T_{\text{II}}^{k-1}| \le 15, \quad |\dot{Q}_{\text{s,ch}}^k - \dot{Q}_{\text{s,ch}}^{k-1}| \le 1{,}000, \quad |\dot{Q}_{\text{s,dch}}^k - \dot{Q}_{\text{s,dch}}^{k-1}| \le 1{,}000, \tag{2.30}$$

$$T_{\text{I}}^k = T_{\text{I}}^{k-1}, \quad T_{\text{II}}^k = T_{\text{II}}^{k-1}, \quad T_2^k = T_2^{k-1}, \quad T_3^k = T_3^{k-1}, \quad \forall k \notin \{1, 5, 9, \dots\}, \tag{2.31}$$

$$\dot{m}_{\text{I}}^k = \dot{m}_{\text{I}}^{k-1}, \quad \dot{m}_{\text{II}}^k = \dot{m}_{\text{II}}^{k-1}, \quad \forall k \notin \{1, 5, 9, \dots\}, \tag{2.32}$$

$$T_{\text{I}}^k \in [177, 250], \quad \dot{m}_{\text{I}}^k \in [5, 16], \quad T_{\text{III}}^k \in [60, 100], \quad R^k \in [30, 240]. \tag{2.33}$$

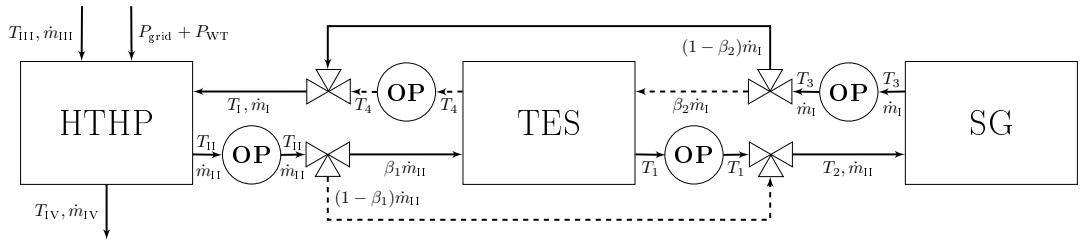

Figure 2.13: Detailed flow chart (Walden et al., 2023) of the studied system (see Fig. 2.1) including HTHP (powered by WT and supported by a grid connection), TES, SG, and OPs. The *solid* and *dashed* lines indicate the charging and discharging mode in a simplified way.

The objective function (2.19) is based on the system's operating costs, which depend on the grid power consumed by the seven pumps (three HTHPs and four OPs). In contrast to Walden et al. (2023), this study also incorporates a demand charge that penalizes high electricity consumption with increased electricity costs defined by the parameter C. In this work, C denotes the maximum electricity price of the optimization horizon n.

The power balance and outlet temperatures of the HTHP are governed by (2.20) and (2.21), where F_{HTHP}, F_{HTHX}, and F_{LTHX} represent the respective surrogate models as functions of inlet temperatures, mass flow rates, and motor speed. The factor of 3 in these equations originates from the fact that the surrogate models were developed for a single HTHP, while three HTHPs are operated in parallel to maintain the component dimensions at a manageable size. To supply the required electrical input for the HTHPs, a VESTAS V150 WT model (Vestas, 2025), with a rated capacity of 4.2 MW and a hub height of 166 m, is considered as a key electricity source alongside grid electricity. The bypass behavior is captured by (2.22), while the SG surrogate models are described by (2.23). The charging and discharging heat flows, detailed in (2.24), are dependent on the HTF mass flow, temperature levels, and fluid flow bypasses. Additionally, the heat losses in (2.25) are influenced by the heat transfer coefficient h, the surface A, the ambient temperature T^k_{ambient}, and the temperature of the storage surface T^k_{s}, assuming that the surface temperature of the storage is the same as the temperature inside the storage. Here, h is fitted to approximate the daily energy losses, which vary depending on the operating conditions and can reach up to 2 % as reported by the manufacturer (Energynest, 2025). Simultaneous charging and discharging is not permitted (see (2.24)).

The constraints in (2.26) and (2.27) correspond to the TES effectiveness model and the variation of the storage temperature over time, with a charging and discharging effectiveness of $\epsilon_{\mathrm{ch}} = \epsilon_{\mathrm{dch}} = 0.9$ and a storage mass of $m_{\mathrm{s}} = 600\,\mathrm{t}$, as in Walden et al. (2023). A design optimization could be addressed in future work to determine a more optimal storage size, as it lies beyond the scope of the present study. The heat capacity of thermal oil $c_{p,\mathrm{f}}$ and concrete $c_{p,\mathrm{s}}$ is modeled using a linear function and third-degree polynomial, respectively, based on the data from Eastman Chemical Company (2025) and Thiel and

Klapperich (2019), with the arithmetic mean temperature difference between the specific inlet and outlet conditions used. The total electric power, consisting of the power supplied from the electrical grid to the three HTHPs and the total power of the four OPs, is described by (2.28). The power of each OP depends on its efficiency η_{OP} as well as the mass flow $\dot{m}_i^k$, nominal mass flow $\dot{m}_{\text{nom},i}^k$, pressure losses Δp_i, and temperature-dependent density $\rho_i(T_i^k)$ of each component involved in the energy utility system, with $i = 1$ for TES charging, $i = 2$ for HTHP, $i = 3$ for SG, and $i = 4$ for TES discharging. In addtion, the following relationshsips hold for the model of the OPs:

$$T_2^k = T_{\text{II}}^k, \quad \dot{m}_1^k = 3\dot{m}_{\text{II}}^k \beta_1^k, \quad \dot{m}_2^k = 3\dot{m}_{\text{II}}^k, \quad \dot{m}_3^k = 3\dot{m}_{\text{I}}^k, \quad \dot{m}_4^k = 3\dot{m}_{\text{I}}^k \beta_2^k. \tag{2.34}$$

In this study, η_{OP} is set to 0.8 for all OPs. This value reflects the performance of commercial OPs (Saiken Pumps, 2025), which are designed for thermal oil applications with operating temperatures up to 380 °C. Moreover, the density $\rho(T)$ is modeled as a second-degree polynomial function of temperature (Eastman Chemical Company, 2025). The nominal mass flow rate is set to 18 kg/s for each component, corresponding to 6 kg/s per unit for the three HTHPs, based on the design specifications described in Section 2.3. The pressure losses are defined for each component as follows: $\Delta p_2 = \Delta p_3 = 4$ bar and $\Delta p_1 = \Delta p_4 = 12$ bar. The total pressure losses for the three HTHPs are based on experimental data conducted for CoBra. The SG is assumed to have the same pressure losses to the three HTHPs in this study. For the TES, the pressure losses during charging and discharging are assumed to be three times higher than those of the HTHPs and SG. This is attributed to the significantly longer tubing used for thermal storage, which results in greater frictional resistance and consequently higher pressure losses. Nonetheless, the assumed values for the SG and TES can be further refined in future work once experimental data become available to better reflect real-world conditions.

In (2.29), a condition is imposed on the storage temperature at the end of the optimization horizon, ensuring that the TES is charged to at least 20 % but not more than 80 %. To limit rapid changes in the outlet temperature of the HTHP as well as the charging and discharging heat flow of the TES, ramp conditions are defined in (2.30). The ramp limits follow the specifications defined by Tran and Stathopoulos (2025) and Energynest (2025), ensuring realistic changes in operating conditions between the time steps.

It is worth noticing that, in contrast to Walden et al. (2023), the optimization problem formulated in this study utilizes two different time resolutions to align with the characteristics of the available data. Specifically, wind power and electricity price inputs are updated every 15 minutes to accurately reflect their high temporal variability. In contrast, the models for the HTHP and the SG operate on an hourly time scale to prevent rapid thermal changes that may negatively affect the system performance and equipment lifespan. This constraint is implemented by enforcing additional equality constraints that maintain constant inlet and outlet temperatures and mass flow rates for both the HTHP and SG within each hour (see (2.31) and (2.32)). The upper limit of the set $\{1, 5, 9, \dots\}$ depends on the total horizon length n of the optimization problem and thus adapts according to the particular use case setup. The 15-minute resolution is commonly

used for trading on the intraday spot market, as noted by Corinaldesi et al. (2020). Last, the surrogate models for the HTHP (see (2.20) and (2.21)) are valid within the box constraints in (4.35).

In summary, we built upon the system introduced in Walden et al. (2023) and extended it to better reflect realistic operating conditions. Key enhancements include the incorporation of temperature-dependent heat capacities, thermal losses, and ramp constraints to capture the dynamic behavior of the system components. Additionally, four OPs were modeled to maintain the required pressure levels throughout the thermal oil loop. Peak demand charges were also considered, penalizing the highest power drawn from the grid during a predefined interval with a high electricity price. In contrast to Walden et al. (2023), where the HTHP models were developed solely based on simulation data, the models in this work were created using a hybrid modeling strategy that combines both simulation and experimental data. This approach enhances model accuracy and extrapolative generalization under real-world conditions.

The effective energy management of the underlying system (see Chapter 5) relies on two key components: first, accurate short-term forecasts of input variables such as wind power, electricity prices, ambient temperature, and heat demand, as addressed in Chapter 3; and second, an accurate, robust, and computationally efficient optimization algorithm capable of solving the resulting nonlinear, nonconvex, constrained optimization problem, as presented in Chapter 4.

3. Input Uncertainties in Conceptual System

The optimization problem introduced in Section 2.4 relies on several key input parameters, including wind power generation, electricity prices, ambient temperature, and heat demand. These parameters play a crucial role in the energy management and typically require accurate forecasting for effective decision-making.

On the one hand, forecasting electricity prices is a particularly complex task, as it depends on numerous variables such as market dynamics, fuel costs, weather conditions, and grid constraints (Singh and Mohanty, 2015; Laitsos et al., 2024). This complexity makes it challenging to incorporate accurate price forecasts within the scope of this study. On the other hand, heat demand forecasting typically requires extensive historical data to capture seasonal patterns, daily variations, and other influencing factors (Hietaharju et al., 2019). Since such data are unavailable for the conceptual system investigated in this work, the heat demand is assumed to be constant and without uncertainties, as in Walden et al. (2023).

Uncertainties in ambient temperature, however, are not explicitly modeled in this study. Although ambient temperature influences the performance of thermal components such as TES, highly reliable short-term forecasts are available from national meteorological platforms like the german weather service (GWS) (Deutscher Wetterdienst, 2025). These forecasts are typically provided on an hourly basis. However, the optimization problem considered in this study requires input data at 15-minute intervals. Since ambient temperature tends to remain relatively stable over short periods, the interpolation from hourly to 15-minute intervals results in only minor inaccuracies. Consequently, treating ambient temperature as deterministic eliminates the need for an additional forecasting model without compromising accuracy.

In contrast, this study explicitly considers uncertainties in wind power data. Wind power forecasting is more uncertain than the ambient temperature due to complex atmospheric dynamics and provides a meaningful way to incorporate uncertainty into the energy management process. Although wind speed forecasts, which directly determine the wind power output (see (2.1)), are publicly available from platforms such as GWS, they are typically provided at hourly intervals, and it is not easy to find high-resolution forecasting data for every location. Wind conditions, however, are known to fluctuate much more rapidly than temperature. Interpolating hourly wind data to finer time resolutions (e.g., 15 minutes) may fail to capture these high-frequency variations, leading to significant inaccuracies. Therefore, instead of relying on coarse wind speed forecasts and imperfect interpolation, we propose to predict wind power directly at the required temporal resolution. This is feasible due to the increasing availability of high-resolution

historical wind speed data, which we can use to train forecasting models. By doing so, we aim to improve forecast accuracy and enhance the robustness of the energy management strategy against uncertainties.

Various approaches have been proposed for the prediction of the wind power and the associated wind speed in recent decades. The approaches can be mainly classified into two main categories: physical and data-based methods. Data-based techniques include persistence, standard statistical, machine learning (ML), and deep learning (DL) methods, as well as hybrid AI-based models (Jiang and Huang, 2017; Santhosh et al., 2019; He et al., 2024).

Physical models rely on fundamental laws of physics and site conditions, including meteorology (e.g., pressure, temperature, humidity), and terrain effects (Carvalho et al., 2012; Su et al., 2014). However, solving the related governing equations requires substantial computational resources, resulting in time-consuming simulations. In contrast, data-based methods use time series or black-box models based on historical data to make predictions. The persistence method uses the most recent known value as the forecast for the next time step, while the other approaches assume that the future data have a linear or nonlinear relationship with the previous data. Compared to the physical methods, they require less computational resources. Physical methods have advantages in medium- and long-term predictions (>6 h ahead), while the data-based methods perform well in short-term predictions (≤ 6 h ahead) (Santhosh et al. (2019)). In this work, we employ data-based models for wind power forecasting due to their computational efficiency and high accuracy in short-term predictions.

The GWS serves as the platform providing wind data for this study. Since this platform offers historical data for wind speed, but not for wind power, we predict wind speed in this section and subsequently determine wind power as a function of the wind speed, as detailed in Section 2.2.

In what follows, this chapter compares several data-driven forecasting models using both one-step and multi-step-ahead strategies across four locations and two seasons in Germany. The goal is to identify the most promising approaches in terms of accuracy and CPU training time.

3.1. Data-Based Methods

Wind speed forecasting has been addressed using a variety of data-based methods. Among these, the persistence model (Santhosh et al., 2019) is one of the simplest and fastest approaches. It predicts the wind speed by assuming that its value at the previous time step will remain unchanged. While this method is computationally efficient and straightforward to implement, its main disadvantage lies in its inability to account for dynamic changes in wind patterns, making it less accurate over longer forecasting horizons or in highly variable conditions. Therefore, it is not further considered in this thesis.

Other commonly employed approaches include statistical models such as multiple linear regression (MLR), MPR, and autoregressive integrated moving average (ARIMA). MLR (Barhmi et al., 2020) and ARIMA (Aasim et al., 2019) are used to model linear relationships or time-dependent patterns, while MPR (Koller et al., 2019) can capture nonlinear trends by fitting higher-order terms to the data. These models

are effective in capturing basic relationships but may struggle with more complex dynamics because their fixed functional forms (predefined equation structure) can not fully capture the nonlinear and time-varying characteristics often present in wind data.

To overcome the limitations of traditional statistical models, ML and DL techniques are increasingly used in the literature. Algorithms like RFR, extra trees regression (ETR), and histogram-based gradient boosting regression (HGBR) (Alkesaiberi et al., 2022) to model nonlinear interactions, while k-nearest neighbors (KNN) (Domingo et al., 2018) works by predicting based on the similarity of nearby data points, making it effective for capturing local patterns. SVR (Wang et al., 2015b) is powerful for handling high-dimensional data and ANNs (Barhmi et al., 2020) are well-suited for modeling intricate, nonlinear relationships through layered architectures.

Decomposition and reconstruction methods, while not a separate category, play a crucial supporting role in statistical, ML, and DL models. Decomposition techniques, such as empirical mode decomposition (Sareen et al., 2023), wavelet transform (Yang et al., 2023), and singular spectrum analysis (Yang and Dong, 2023), help break down complex time series data – which exhibit trends, seasonal cycles, and irregular fluctuations – into simpler components for easier modeling. This enhances the ability of models to focus on specific aspects of the data, improving overall the forecasting accuracy. Reconstruction methods, such as phase space reconstruction (He et al., 2024), then reassemble these components to generate the final prediction. However, decomposition and reconstruction techniques are not considered in this work, as the focus remains on simpler models for the sake of computational efficiency and clarity.

To further enhance forecasting performance, hybrid AI methods, that combine convolutional neural networks (CNN) and long short-term memory (LSTM), are considered to capture both short-term, detailed features (with CNN) and long-term dependencies (with LSTM) in the data (Sari et al., 2020; Wu et al., 2021). More advanced hybrid AI methods based on the non-negative constraint theory (NNCT) strategy couple various ML and DL techniques and integrate methods like genetic algorithms or simulated annealing, to optimize the weight coefficients of each single prediction model (Lv et al., 2021; Wang et al., 2022; He et al., 2024). Despite the advantages of hybrid AI models based on the NNCT strategy, this work focuses on simpler hybrid approaches, like the CNN-LSTM model, without the need for additional optimization algorithms. The models considered in this work are briefly presented below and have been selected due to their wide application in the literature.

3.1.1. Multiple Linear Regression

The MLR is one of the classical statistical tools to describe linear relationships between a dependent and more independent variables (Chong et al., 2015; Azadi and Karimi-Jashni, 2016). The model for making predictions is shown in the following equation:

$$y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + \ldots + \beta_K x_{Kt} + \epsilon_t, \tag{3.1}$$

where y_t, i.e., $y_t = y(t)$, is the dependent variable and x_{it} ($i = 1, 2, 3, \ldots, K$) denotes the independent variables at every time step t. The parameter β_i ($i = 0, 1, 2, 3, \ldots, K$) describes the regression coefficients, while ϵ_t is the residual error of the regression model at every time step. The parameters of the MLR are determined using the OLS, which minimizes the sum of the squared differences between the values being predicted by the MLR and the given values from the dataset (Ghysels and Qian, 2019).

3.1.2. Multiple Polynomial Regression

MPR is a statistical technique to fit a nonlinear equation to a dataset by employing polynomial functions of the independent variables and can be described as following (Imran et al., 2022):

$$y_t = \beta_0 + \sum_{i=1}^{K} \beta_i x_{it} + \sum_{i=1}^{K} \sum_{j=i}^{K} \beta_{ij} x_{it} x_{jt} + \sum_{i=1}^{K} \beta_{ii} x_{it}^2 + ... + \epsilon_t. \tag{3.2}$$

Compared to (3.1), this formulation includes interaction effects between the independent variables (represented by $x_{it}x_{jt}$ terms) and nonlinear terms (such as quadratic terms x_{it}^2 and potentially higher-order terms).

3.1.3. Autoregressive Integrated Moving Average

ARIMA, developed by Box and Jenkins, was inspired by their work on forecasting problems in business, economics, and control engineering. It builds a linear relationship between future and previously observed values (Box and Jenkins, 1970).

The ARIMA model consists of the autoregressive (AR) model combined with the moving average (MA) model. The AR model of order p, known as an AR(p) model, is described as follows:

$$w_t = \phi_1 w_{t-1} + \phi_2 w_{t-2} + \ldots + \phi_p w_{t-p} + \epsilon_t, \tag{3.3}$$

while the MA(q) model is expressed by the following equation:

$$w_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}. \tag{3.4}$$

The resulting ARIMA(p, d, q) model is then defined as:

$$w_t = \phi_1 w_{t-1} + \phi_2 w_{t-2} + \ldots + \phi_p w_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}, \tag{3.5}$$

where w_t describes the stationary time series obtained by taking the d^{th} difference of y_t, ϕ_i denotes coefficients associated with each previously observed value and w_{t-i} the observed previous values. In addition, θ_i represents the coefficients associated with previous white noises, ϵ_t is a normal white noise process with zero mean and variance σ^2 and ϵ_{t-i} stands for the previous noise terms. If w_t is replaced

by y_t, which means that $d = 0$, (3.5) indicates the presence of an autoregressive moving average model, abbreviated as ARMA(p, q). The ARIMA model and its theoretical background are described in detail in Box et al. (1994).

3.1.4. k-Nearest Neighbors

KNN is a simple, nonparametric (i.e., it does not assume any predefined relationship between input and output variables) algorithm used in supervised learning for regression and classification tasks (Kramer, 2013). It works by predicting the value of a target point based on the values of its k-closest neighbors in the feature space. The closeness, or similarity, between data points is usually measured using distance metrics like EUCLIDEAN distance. The predicted value for a target point is typically the average of the values of its k-nearest neighbors, which is expressed by the following equation:

$$y_t = \frac{1}{k} \sum_{i=1}^{k} y_i, \tag{3.6}$$

where k describes the number of closest neighbors and y_i their corresponding target values. The choice of k affects the model's performance: a small k makes the model more sensitive to noise (overfitting), while a large k smooths out predictions, potentially leading to underfitting.

3.1.5. Support Vector Regression

SVR is an extension of the support vector machine, proposed by Cortes and Vapnik (2004), in regression. The basic principle of SVR is to map the original data into a high-dimensional feature space through nonlinear mapping (Hu et al., 2013). A linear regression is then performed in the high-dimensional feature space to fit the target variable. SVR gives the flexibility to define how much error is acceptable in the model with the aim of finding an appropriate line (or hyperplane in higher dimensions) to fit the data. The regression formula of SVR is expressed as follows:

$$y_t = \boldsymbol{w}^\top \phi(\boldsymbol{x}_t) + b_t, \tag{3.7}$$

where $\phi(\cdot)$ is a nonlinear mapping function, $\boldsymbol{w}$ denotes the weight vector, and b describes the bias. In contrast to OLS, the objective function of SVR is to minimize the L_2-norm of the weight vector and not the squared error.

To apply linear techniques to nonlinear problems, SVR uses appropriate kernels and (3.7) is thus modified to:

$$y_t = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) k(\boldsymbol{x}_i, \boldsymbol{x}_t) + b_t, \tag{3.8}$$

where α_i and α_i^* represent positive and negative LAGRANGE multipliers, respectively, N denotes the number of support vectors determined based on the training data and the choice of hyperparameters, and

$k(\boldsymbol{x}_i, \boldsymbol{x}_t)$ is the kernel function, measuring the similarity of $\boldsymbol{x}_t$ to each support vector $\boldsymbol{x}_i$. The typical kernel functions include polynomial kernel functions, GAUSSIAN RBF, and sigmoid functions (Wang et al., 2015a). In this study, the RBF kernel is applied due to its computational simplicity, robustness in dealing with nonlinear data, and widespread application (Wang et al., 2015a; Jiang et al., 2015; Tao et al., 2021).

3.1.6. Ensemble Techniques

Ensemble methods (Dietterich, 2000) combine several individual models, often called base learners, to create a more powerful and reliable predictive model. The core idea behind these techniques is that by merging multiple models, overfitting and bias can be minimized, leading to improved generalization on new, unseen data. The outputs of the different models are aggregated in various ways, such as averaging for regression tasks or voting for classification tasks, to produce the final prediction. Some of the most commonly used ensemble methods are:

Random Forest Regression: RFR is a nonparametric ensemble technique, which is used for regression problems. This technique is first introduced by Breiman (2001) and it is an extended version of the decision tree algorithm, which forms a tree structure working on a set of rules and on the possible outcomes. RFR generates a forest of N trees from a given dataset (Drisya et al., 2017). Each node in the forest has a splitting condition based on only the randomly selected predictor attributes. These attributes reduce the error rate by avoiding the correlation among the trees. The split of nodes continues until a maximum tree depth or stopping criterion is reached. For each tree, there is a certain output. The response from the RFR is determined calculating the average of all individual regression tree outputs.

Extra Tree Regression: ETR is an ensemble learning technique used for regression that operates by constructing multiple decision trees during training (Geurts et al., 2006). Unlike traditional decision trees, ETR introduces additional randomness by selecting random splits for each feature, rather than choosing the most optimal split. This randomness helps to reduce variance, leading to more robust predictions, especially on noisy datasets. The final prediction is typically the average of the outputs from all the trees in the ensemble.

Histogram-Based Gradient Boosting Regression: HGBR is an advanced algorithm based on gradient boosting, designed for large datasets (Pedregosa et al., 2011). Unlike traditional Gradient Boosting, which uses exact values of features for splits, HGBR first bins continuous features into discrete buckets (or histograms), which speeds up the training process by reducing the number of split points that need to be evaluated. This method is computationally efficient and works particularly well with high-dimensional data, while still maintaining the flexibility and predictive power of gradient boosting. It iteratively fits new models to correct the residuals of previous models, improving accuracy with each step.

3.1.7. Artificial Neural Network

ANN is a nonlinear method for forecasting, which offers a black-box model to simulate the nonlinear relationship between the input and the output layers. It consists of an input layer, one or more hidden layers and an output layer. Moreover, a number of artificial neurons appears in each layer. A nonlinear mapping in the network is possible using activation functions between neurons of different layers (Tao et al., 2021). This section summarizes two widely used networks, namely the convolutional neural network and long short-term memory.

Convolutional Neural Network: CNN is a type of neural network with convolution structure to automatically extract spatial features from input data and is commonly trained using supervised learning (LeCun et al., 1989). As seen in Fig. 3.1, a CNN consists of an input layer, one or more convolutional, pooling, and fully connected layers, and an output layer. The input layer receives the raw data, while the convolutional layer applies a set of filters (kernels) that slide over the input layer to produce feature maps, capturing local patterns. These maps are passed to the pooling layer, which reduces their spatial dimensions, improving computational efficiency and robustness to small spatial variations (Lecun et al., 1998). The fully connected layer integrates the extracted features into a one-dimensional representation, and the output layer generates the final prediction, such as the next-step wind speed in a regression task.

Long Short-Term Memory: LSTM is a type of recurrent neural network (RNN) designed to handle sequential data effectively (Hochreiter and Schmidhuber, 1997). One of the key advantages of LSTM over traditional RNN models is its ability to mitigate the vanishing gradient problem, which allows it to retain information over longer time intervals. Unlike conventional RNNs that struggle with long-term dependencies, LSTM networks incorporate a gating mechanism that regulates the flow of information through the network. LSTMs are capable of processing entire sequences of data rather than just individual data points. They achieve this by using cyclic connections to capture temporal dependencies and long-term relationships within the data. Fig. 3.2 illustrates the structure of a memory cell of an LSTM network.

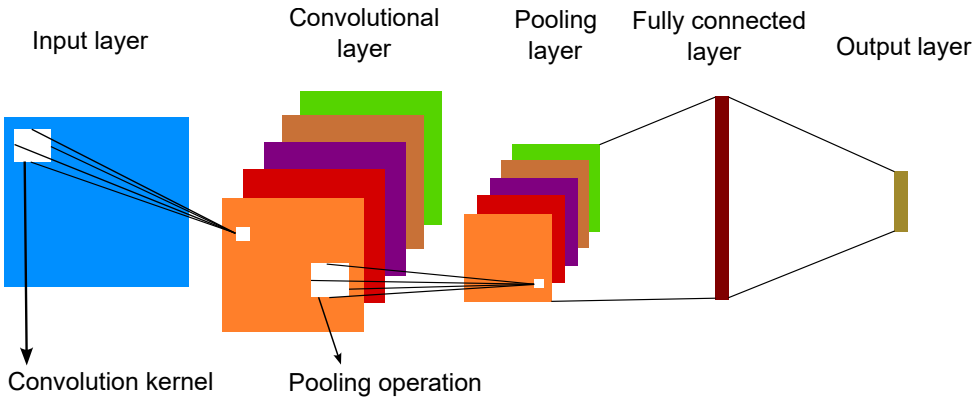

Figure 3.1: Architecture of a CNN, consisting of an input layer (receives the raw data), a convolutional layer (for spatial feature extraction via filters), a pooling layer (for dimensionality reduction), a fully connected layer (for feature integration and transformation into one-dimensional vectors), and an output layer (for the final prediction).

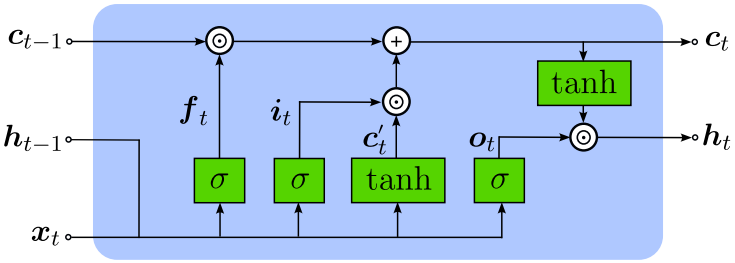

Figure 3.2: Structure of a LSTM memory cell.

A standard LSTM unit consists of a cell state and three gates: the input, output, and forget gate. The cell state acts as a memory unit that retains information across time steps, while the gates control how information is updated and passed through the network. More precisely, the input and the output gate control the transfer of input and output activations and the forget gate updates the interior state adaptively. The processes of updating the cell state and calculating the output of the LSTM are described by:

$$\boldsymbol{f}_t = \sigma\left(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + \boldsymbol{b_f}\right), \tag{3.9}$$

$$\boldsymbol{i}_t = \sigma\left(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + \boldsymbol{b_i}\right), \tag{3.10}$$

$$\boldsymbol{o}_t = \sigma\left(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + \boldsymbol{b_o}\right), \tag{3.11}$$

$$\boldsymbol{c}'_t = \tanh\left(W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1} + \boldsymbol{b_c}\right), \tag{3.12}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{c}'_t, \tag{3.13}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{3.14}$$

where $\boldsymbol{x}_t$ is the input vector of the LSTM cell. The vectors $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, and $\boldsymbol{o}_t$ denote the activation vectors of the input, forget, and output gate, respectively. Moreover, the hidden state vector, also known as the output vector of the LSTM unit, is represented by $\boldsymbol{h}_t$ and the cell input activation vector as well as the cell state vector are described by $\boldsymbol{c}'_t$ and $\boldsymbol{c}_t$, respectively. In addition, W and U stand for the weight matrices, $\boldsymbol{b}$ is the bias vector, σ describes the logistic sigmoid function and tanh the hyperbolic tangent function. The symbol $\odot$ denotes the HADAMARD product (element-wise product).

Hybrid CNN-LSTM Model: The hybrid CNN-LSTM model leverages the strengths of both CNN and LSTM (Sari et al., 2020; Wu et al., 2021). The CNN component is responsible for extracting spatial features from the input data, while the LSTM component captures temporal patterns, making it well-suited for time-series modeling.

3.2. Data Processing & Model Evaluation

The data considered in the present study to train and test the models are obtained from the GWS platform for the period 2010 to 2021 (i.e., 12 years) with a temporal resolution of 10 minutes. GWS

provides the air temperature at a height of two meters above ground, air pressure at station level, air pressure gradients between two locations at station level, historical wind speed at ten meters, wind direction at ten meters, and relative humidity at two meters above ground. In addition, a pseudo parameter is derived from the GWS, taking the value of 1 for January, 2 for February, and increasing incrementally for the following months.

The regions studied are Cottbus in eastern Germany, Düsseldorf in western Germany, Hamburg in northern Germany, and Augsburg in southern Germany. Moreover, the models are evaluated in this study in two different seasons, in winter and summer 2021, in order to test their robustness across different locations and seasonal conditions, where wind speeds vary significantly.

In the GWS data, wrong measurements marked with -999 and missing data often appear in the datasets. These values must be replaced so forecasting methods can make accurate predictions based on these data. This has been done using 1-D monotonic cubic interpolation in this work, with the details of the interpolation model provided in the Appendix A.1.1. This interpolation is also used to resample the wind data from a 10-minute to a 15-minute resolution, as required for the optimization problem presented in Section 2.4.

The selection of the model input parameters is of crucial importance to enhance the output forecasting performance. In this study, our aim is to select the variables, which influence the wind speed the most. To achieve this, two complementary approaches are employed, which ensure a robust variable selection process that improves the accuracy and reliability of wind speed forecasting.

First, a p-value-based selection method (Murtaugh, 2014) is used that represents the probability that the observed effect could have occurred under the null hypothesis, which typically assumes no effect or relationship. This approach is to fit a full multiple regression model and iteratively removes terms one at a time, starting with the term with the highest p-value. This process continues until only terms with p-values below 0.05 remain, a commonly used threshold indicating that there is less than a 5 % probability that the observed relationship occurred by chance (i.e., these terms have a statistically significant relationship with the dependent variable y).

Second, the Akaike information criterion (AIC) (Murtaugh, 2014) is applied to assess the model quality. This method is used to compare different possible models and determine which one is the best fit for the data. The formula for AIC is the following:

$$\text{AIC} = 2K - 2\ln(L), \tag{3.15}$$

where K is the number of independent variables and L is the log-likelihood estimate. Lower AIC scores indicate better models than higher AIC values.

The variables showing the strongest influence on the wind speed determination, as identified through the p-value analysis and the AIC method, are the wind speed from the previous five time steps, the

temperature from one year ago and the pressure from one day ago, considering the seasonality of the temperature and the pressure. The resulting wind speed model is expressed as follows:

$$v_t = f(v_{t-1}, v_{t-2}, v_{t-3}, v_{t-4}, v_{t-5}, T_{t-\text{year}}, p_{t-\text{day}}), \tag{3.16}$$

where v denotes the wind speed, T the air temperature, and p the air pressure. The function f represents the relationship learned by the model between the input features and the predicted wind speed, which can be linear or nonlinear depending on the specific model used. The independent variables (features) in (3.16) have been selected for the MLR and are considered in every model presented in Section 3.1 except for the ARIMA model, which only takes into account the wind speed at previous time steps. The MPR model additionally accounts for interactions between wind speed values at previous time steps. In this work, a second-degree MPR is considered, as preliminary (not reported) tests with a third-degree MPR, based on the scenarios described in Section 3.3, did not improve the forecasting accuracy, but it considerably increased the number of polynomial terms (features) and thus the model complexity. The resulting model for the MPR, derived based on the p-value approach and the AIC method, is expressed as follows:

$$\begin{aligned} v_t = f(&v_{t-1}, v_{t-2}, v_{t-3}, v_{t-4}, v_{t-5}, T_{t-\text{year}}, p_{t-\text{day}}, v_{t-1}^2, v_{t-1}v_{t-2}, v_{t-1}v_{t-3}, v_{t-1}v_{t-5}, \\ &v_{t-2}^2, v_{t-2}v_{t-3}, v_{t-2}v_{t-4}, v_{t-2}v_{t-5}, v_{t-3}^2, v_{t-3}v_{t-5}, v_{t-4}^2, v_{t-4}v_{t-5}, v_{t-5}^2). \end{aligned} \tag{3.17}$$

All the terms listed above represent the features selected for their strongest influence in the MPR model.

After collecting data, data processing – and more specifically, feature scaling – is crucial for the performance of algorithms. Feature scaling is a crucial technique that transforms features to a consistent range, ensuring each variable contributes proportionally to the model's learning process. By bringing features to a comparable range of values, scaling prevents larger magnitude features from excessively influencing model predictions and helps often algorithms converge faster during training. Two common feature scaling techniques are the standardization and normalization described by the following equations (Wan, 2019):

$$x_i' = \frac{x_i - \bar{x}}{s} \quad \text{(standardization)}, \tag{3.18}$$

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \quad \text{(normalization)}, \tag{3.19}$$

where $\bar{x}$ and s are the mean value and the standard deviation, respectively. Standardization centers the data to have a mean of zero and a variance of one, while normalization transforms the values into the interval [0,1].

To evaluate the forecasting performance of the algorithms, certain metrics are used in the literature. Three widely used metrics, which are considered in this work, are the mean absolute error (MAE), root

mean squared error (RMSE), and coefficient of determination R^2 (introduced in (2.18)), and are defined as follows (Tao et al., 2021):

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \quad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{3.20}$$

where y_i and $\hat{y}_i$ denote the measured and predicted values, respectively, and N is the number of data points considered in the performance evaluation. MAE measures the average magnitude of errors between predicted and actual values, providing a straightforward measure of prediction accuracy. RMSE emphasizes larger errors by squaring them, offering insight into the variance of errors, while R^2 quantifies how well the model explains the variance in the data, with values closer to 1 (or 100 %) indicating a good fit and values close to 0 (or 0 %) a poor fit.

However, models may sometimes suffer from overfitting, where they perform well on training data but poorly on unseen data. To mitigate this issue, CV techniques (Berrar, 2019) are often employed to ensure the model's robustness and generalizability.

As already mentioned in Section 2.3.2, one of the most known CV methods is the k-fold CV, where the dataset is divided into k equally sized folds (Berrar, 2019). Then, the model is trained on $k-1$ of these folds and tested on the remaining fold, repeating this process k times with each fold serving as the test set exactly once, and the final metrics are determined as the average of scores obtained in every fold.

In the case of time series, CV is not trivial. It is illogical to choose random samples and assign them to either the test set or the train set because future values are used to predict past values in this way. For this reason, blocked CV based on the function TimeSeriesSplit of the Sklearn tool (Pedregosa et al., 2011) and modified according to Mustafa Qamaruddin (2019) is considered in this work. As seen in Fig. 3.3, margins are added at two positions. The first margin is positioned between the training and the testing folds so the model does not use values twice. The second margin appears between the folds with the aim of preventing the model from memorizing patterns from an iteration to the next.

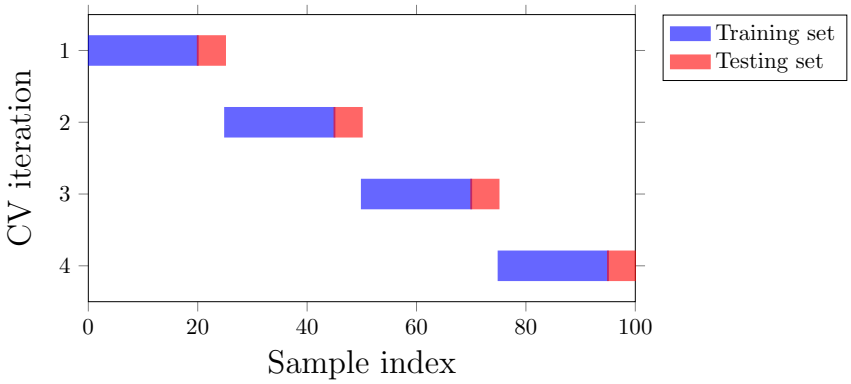

Figure 3.3: Schematic structure of blocked CV for time series split.

3.3. Short-term Wind Speed Forecasting

The methods introduced in Section 3.1 have been tested to evaluate their performance for one-step-ahead (forecasting the next time step) and multi-step-ahead (forecasting several future time steps) predictions, using Cottbus, Düsseldorf, Hamburg, and Augsburg as example locations in winter and summer 2021. The results of these experiments are presented in this section. In this study, an ARIMA(0,2,2) model derived from the autocorrelation and partial autocorrelation figure (not shown in this thesis) has been considered, while all ML and DL methods underwent hyperparameter tuning utilizing RandomizedSearchCV from Sklearn in Python to achieve optimal performance. To prevent overfitting, all results presented in the tables are obtained using blocked CV with 5 folds, where each fold is split into 90 % training and 10 % test data. Moreover, all models are implemented in Python 3.12. More precisely, ANN models are implemented with the Tensorflow library, with early stopping considered to prevent overfitting. ARIMA is employed through Statsmodels (Skipper Seabold and Josef Perktold, 2010), and all the other forecasting methods are applied using Sklearn. To enhance model performance, feature scaling is applied where necessary. Specifically, KNN and ANN require normalization, while SVR benefits from standardization.

3.3.1. One-Step-Ahead Forecast

This section evaluates the performance of the models presented in Section 3.1 for a one-step-ahead (15-minutes) forecast. In order to further quantify the forecasting errors and to compare the methods to each other, the metrics R^2, RMSE, and MAE are calculated and listed in Table 3.1 for Cottbus in winter 2021 and in Tables A.1-A.7 (Appendix A.2.2) for Cottbus in summer 2021 as well as for the other three locations in both seasons. Moreover, the CPU time for training every method, referring to Cottbus in 2021, is listed in Table 3.1. For the other locations and seasons, this information has not been included as it is similar to that of Cottbus in winter 2021. Computations for all forecasting methods are performed in this study with an Intel(R) Xeon(R) Gold 5220 CPU @ 2.20 GHZ.

As seen in Fig. 3.4, the forecasting results by the proposed methods have slight differences and can generally resemble the measured wind speeds for Cottbus in winter 2021. However, not all the peaks of the measured wind speed are forecasted by the models. This happens because the statistical methods based on the accumulated historical data are able to capture the trend of the time series, but not the noise which appears in the time series. Similar phenomena appear in Fig. A.2 in Appendix A.2.2 for the location "Düsseldorf" in summer 2021. The two figures do not include all three ensemble techniques, as Tables 3.1 and A.1-A.7 do, but only HGBR, since their accuracy is very similar. However, HGBR was chosen for its significantly faster computational speed.

Most of the models have an accuracy of approximately 90 % on average in the investigated locations and seasons. The differences between the models for one-step-ahead wind speed forecast can be clearly seen in Table 3.1. According to this table, MLR has one of the highest accuracy metrics and the lowest

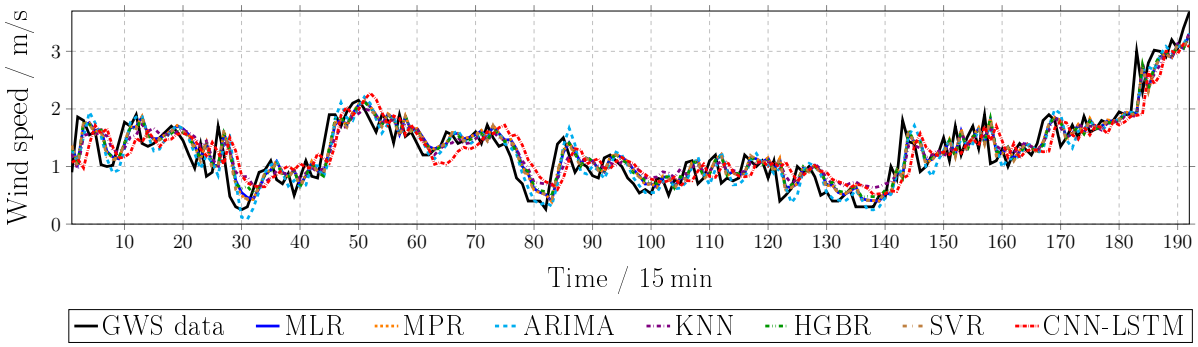

Figure 3.4: Results of one-step-ahead wind speed forecast, exemplified for Cottbus in winter 2021.

computational time (less than 1 second) required to be trained. In addition, the interactions between the independent variables using MPR do not drastically improve the forecasting accuracy in comparison to MLR, but they slightly increase the CPU time of the model (from 0.12 up to 0.20 seconds). This can also be observed in the results for Cottbus in summer 2021 and for the other three locations in both seasons, except for the results in Düsseldorf in summer 2021, where no improvement in the accuracy metrics appears in MPR (see Tables A.1-A.7). The high accuracy of the MLR model indicates that the relationships between the selected features and the dependent variable (wind speed) are largely linear and well captured by the model. This also suggests that, for the given datasets, introducing nonlinearities or interaction terms provides only marginal improvements, as the essential dynamics are already represented in the linear formulation. To examine potential overfitting in MLR and MPR, regularization was applied using Ridge and Lasso regression from Sklearn. However, as the models were already evaluated using blocked CV, no signs of overfitting were observed, and the regularization did not lead to any improvement in forecasting accuracy. Therefore, these results are not included in the aforementioned tables and regularization is not further considered in this work.

Table 3.1: Comparative analysis (R^2, RMSE, MAE, CPU training time) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Cottbus in winter 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)	CPU time (s)
MLR	89.13	0.47	0.34	0.12
MPR	89.18	0.47	0.34	0.20
ARIMA	88.07	0.49	0.35	79
KNN	87.84	0.50	0.36	4
RFR	89.11	0.47	0.34	212
ETR	89.13	0.47	0.34	22
HGBR	89.09	0.47	0.34	3
SVR	89.20	0.47	0.34	586
LSTM	85.53	0.54	0.41	610
CNN-LSTM	86.65	0.53	0.40	627

The models RFR, ETR, HGBR, and SVR achieve similar accuracy metrics to MLR and MPR ($\approx 90\,\%$ on average across the four locations and the two seasons), showing a high robustness in their results. However, they need much more time to be trained (212, 22, and 586 seconds on average for RFR, ETR, and SVR, respectively), except for the HGBR, which requires three seconds on average. The statistical model "ARIMA" does not demonstrate robust behavior in the results, as its performance fluctuates significantly – sometimes achieving the highest accuracy, while at other times yielding the lowest accuracy compared to the other methods. On average, its accuracy is approximately 89 %. An ARIMA(0,2,2) model considered in this thesis applies second-order differencing to make the wind speed series stationary and includes two MA terms, which rely on past forecast errors from the differenced series. Since there are no AR terms ($p = 0$) and no additional explanatory variables (features), the model can not directly account for influences such as pressure, temperature, or past wind speeds beyond the differenced wind speed series. While differencing and the MA terms can help capture short-term trends, the model remains sensitive to nonstationary behavior in wind speed, which is not explicitly modeled. In contrast, MLR incorporates multiple explanatory variables – including past wind speeds, air pressure, and air temperature – allowing it to capture broader relationships and maintain more consistent performance across different locations and seasons. Additionally, ARIMA requires a high amount of CPU time for training (79 seconds on average), further limiting its practicality.

KNN shows good performance, ranking close to the top-performing methods, although not among the highest ($\approx 88\,\%$ accuarcy on average). Moreover, its training time is relatively low (4 seconds on average), making it an efficient option in terms of computational resources. The slightly lower accuracy of KNN compared to MLR, MPR, RFR, ETR, HGBR, and SVR is expected, because KNN relies on local averaging over nearby training points in feature space (see (3.6)) rather than explicitly learning feature-target relationships. This makes it more sensitive to noisy data and less capable of capturing complex or global patterns in the wind speed time series.

Last, the hybrid CNN-LSTM model, whose architecture is presented in Appendix A.2.1, consistently has one of the lowest accuracy metrics across all cases ($\approx 86\,\%$ accuracy on average). This lower accuracy is likely due to the neural network's sensitivity to the noisy nature of the wind speed data and the difficulty of optimally tuning the numerous hyperparameters of the neural network. Despite this, the model's accuracy remains comparable to the other models, though it incurs higher computational costs (627 seconds on average). However, the CNN-LSTM model shows better performance than the LSTM model introduced in A.2.1, which achieves an average accuracy of approximately 84 % and a similar training time of 610 seconds. This improvement in the accuracy occurs because the CNN component of the hybrid model extracts relevant spatial features from the input data, providing richer representations that the LSTM can then process to better capture temporal patterns, which the standalone LSTM may miss. For this reason, only the hybrid model is considered in the figures presented in this and next section.

In future work, filtering techniques like Kalman filter, moving average, and Savitzky-Golay filter (Durbin and Koopman, 2012) can be employed to reduce noise in the data and assess whether they

improve the performance of the models, especially of the neural networks. It is also worth noticing that the predictions of the models in winter have higher accuracy than in summer. This is likely due to the higher wind speeds in winter, which provide a clearer signal for forecasting models to detect and analyze, reducing the impact of noise compared to summer.

3.3.2. Multi-Step-Ahead Forecast

After presenting the results of the one-step-ahead forecast, we now turn to the multi-step-ahead forecast to evaluate the models' performance over a longer prediction horizon. Unlike one-step-ahead forecasting, which only predicts the next time point, multi-step-ahead forecasting aims to predict several steps into the future. This aspect is particularly important for energy management applications, where decisions are made over a prediction horizon within each iteration of the RHA. Therefore, both the accuracy and computational efficiency of multi-step-ahead forecasts directly influence the quality of the subsequent optimization and operational planning.

Forecasting in multiple steps in the future is considerably more difficult than a single-step forecast. This happens due to the accumulation of errors, reduced accuracy, and increased uncertainty (An and Anh, 2015). This work considers the following strategies for multi-step forecasting (Rodriguez et al., 2020), which are compared to each other for different models:

Recursive (Rec) Strategy: In this strategy, the model f is trained to forecast a single time step, as seen in the following equation:

$$y_t = f(y_{t-1}, \ldots, y_{t-w}) + \epsilon, \tag{3.21}$$

where w is the size of the sliding temporal window (i.e., the number of past observations used as input to predict the next value), ϵ denotes the residual error, $t \in \{w+1, \ldots, N\}$, and N is the total number of observations in the dataset. To predict H steps, the forecasted value from the previous step is recursively fed back into the model as input to predict the next step, without retraining the model at each iteration. This process is repeated until the H horizon is fully forecasted.

Direct (Dir) Strategy: In the Dir strategy, H separate models are trained, each to forecast a specific prediction horizon h. Unlike the Rec strategy, the forecasts produced at previous steps are not used as inputs for subsequent predictions. The Dir strategy is described by (3.22) as follows:

$$y_{t+h-1} = f_h(y_{t-1}, \ldots, y_{t-w}) + \epsilon, \tag{3.22}$$

with $t \in \{w+1, \ldots, N-H+1\}$ and $h \in \{1, \ldots, H\}$.

Multiple-Input Multiple-Output (MIMO) Strategy: The MIMO strategy produces multiple outputs and each output corresponds to a different forecasting horizon. A single multi-output prediction model F is used in this strategy, as seen in the following equation:

$$[y_t, \ldots, y_{t+H-1}] = F(y_{t-1}, \ldots, y_{t-w}) + \epsilon, \tag{3.23}$$

with $t \in \{w+1, \ldots, N-H+1\}$.

Direct-Recursive (DirRec) Strategy: The DirRec strategy combines both the Dir and the Rec strategy. As seen in (3.24), each forecasted time step is computed separately using a different model for each horizon, as in the Dir strategy, while also incorporating previous forecasts into the input window, following the Rec approach:

$$y_{t+h-1} = f_h(y_{t+h-2}, \ldots, y_{t+h-w-1}) + \epsilon, \tag{3.24}$$

with $t \in \{w+1, \ldots, N-H+1\}$ and $h \in \{1, \ldots, H\}$.

Direct Multiple-Output (DirMO) Strategy: The DirMO strategy combines the advantages of both the Dir and the MIMO strategy. This strategy divides the horizon H into s blocks and each block represents an independent model that forecasts n outlets. The number of outputs that each block forecasts is described by the following equation:

$$n = \frac{H}{s}. \tag{3.25}$$

The DirMO strategy is defined by (3.26) as follows:

$$[y_{t+(p-1)n}, \ldots, y_{t+pn-1}] = F_p(y_{t-1}, \ldots, y_{t-w}) + \epsilon, \tag{3.26}$$

with $t \in \{w+1, \ldots, N-H+1\}$ and $p \in \{1, \ldots, s\}$.

The results of the multi-step-ahead forecast are presented in Figs. 3.5-3.7, which show a comparison of multi-step predicted wind speeds over several forecasting steps by the proposed models with the measured data for Cottbus in summer 2021. These results are based on the Rec strategy, which is widely used in multi-step forecasting due to its simplicity and recursive structure. However, all five forecasting strategies are compared later in this section to evaluate their performance. Since all methods, except for MLR, MPR, KNN, and HGBR, are computationally expensive, it is important to emphasize that all models were trained only once at the beginning of the forecasting process to allow for a fair comparison. After each forecasting horizon, they used the actual measured values for the next predictions without undergoing retraining. Nevertheless, a study carried out in this thesis shows in Tables A.8-A.9 (Appendix A.2.2) that retraining MLR, MPR, KNN, and HGBR after each prediction horizon, following the Rec strategy, has only a minor effect on MAE. The observed differences occur mainly in the fourth decimal place, with the largest differences corresponding to relative changes of up to approximately 0.1 %, indicating that retraining does not substantially influence the forecasts.

The results in Figs. 3.5-3.7 support the fact that the deviations between forecasted and measured wind speeds become more and more prominent as the time horizon expands, since the predicted value of the current step is considered in the model for the prediction of the wind speed in the next step.

As seen in Figs. 3.5-3.7, ARIMA has a worse performance than the other methods as the forecasting step increases. Since the real values in a certain forecasting horizon are not known, the errors of the previous time steps appearing in the MA part (see (3.4)) of ARIMA can not be determined and they

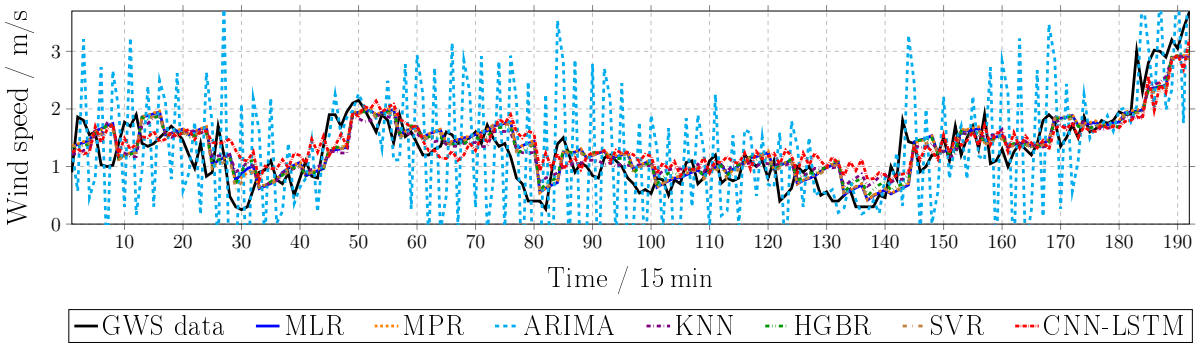

Figure 3.5: Results of four-step-ahead wind speed forecast, exemplified for Cottbus in winter 2021.

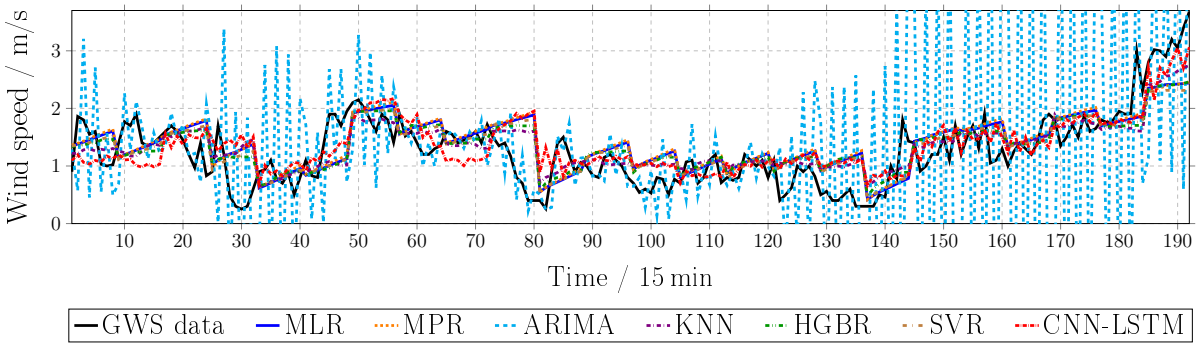

Figure 3.6: Results of eight-step-ahead wind speed forecast, exemplified for Cottbus in winter 2021.

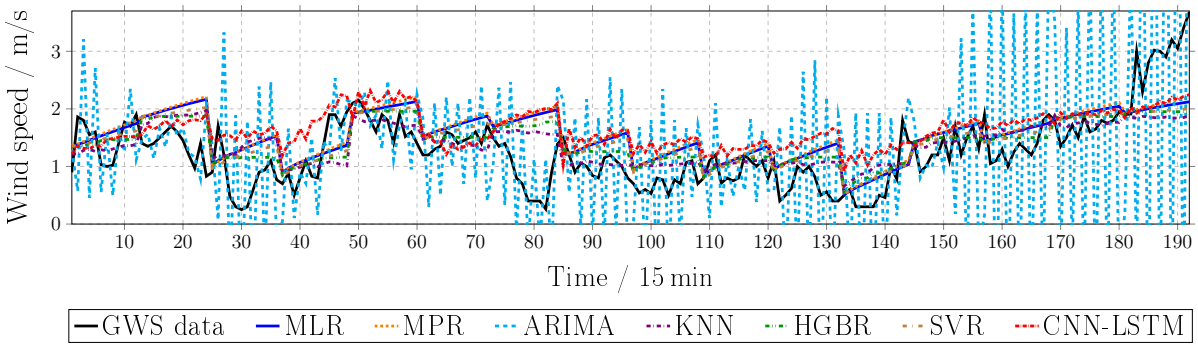

Figure 3.7: Results of twelve-step-ahead wind speed forecast, exemplified for Cottbus in winter 2021.

are assumed to be zero. For this reason, the predictions using ARIMA have high accuracy in the first forecasting step, while their accuracy drastically decreases after the first step as the deviations between the predicted and actual values increase (see Figs. 3.5-3.7). All the other methods have small differences to each other. A similar trend is observed in Figs. A.3-A.5 in Appendix A.2.2, which display the results for Düsseldorf during summer 2021. The differences in the performance of the models over the forecasting steps can be seen in Fig. 3.8. This figure shows the prediction error of different models with respect to the forecasting steps using MAE as performance evaluation criterion, providing a clear view of average error magnitudes. It should be underlined at this point that the data for the following figure has been generated after training the models only once at the beginning of the two-day dataset. The location, which is taken

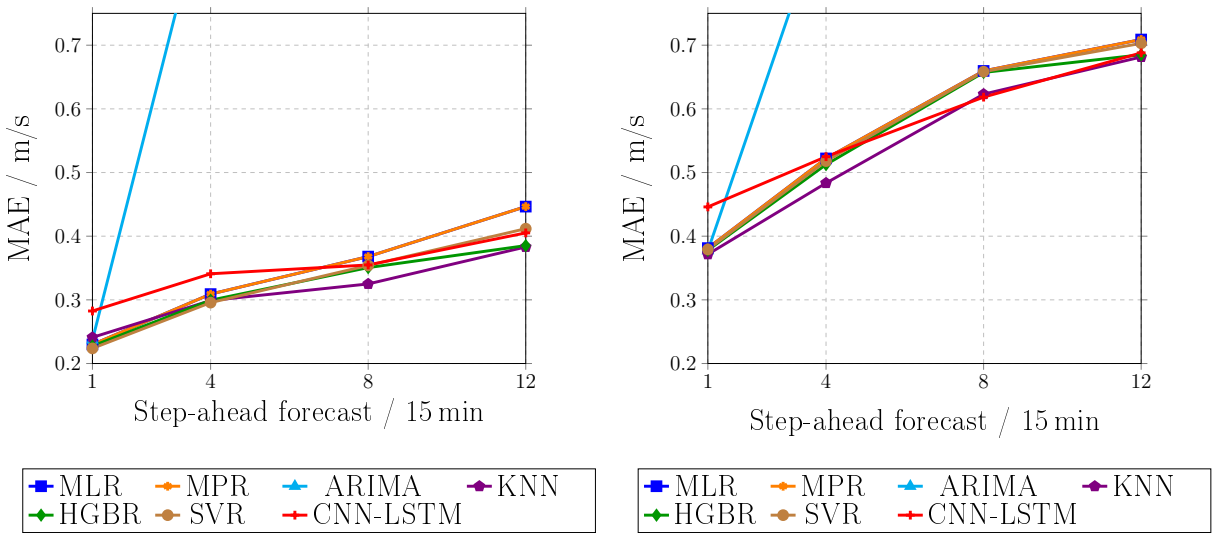

Figure 3.8: MAE comparison of wind speed forecast results for the presented methods with respect to different multi-step-ahead horizons, exemplified for Cottbus in winter 2021 (**left**) and Düsseldorf in summer 2021 (**right**). From the fourth step-ahead forecast onwards, ARIMA reaches MAE values up to 1.53 m/s, which exceed the y-axis limit; for clarity, only the range 0.2–0.75 m/s is displayed to allow detailed comparison of the other methods.

into consideration in this figure, is Cottbus in winter 2021 as well as Düsseldorf in summer 2021. The forecasting horizon ranges from 1 to 12 steps ahead, corresponding to up to 3 hours into the future.

As expected, the prediction error increases over the forecasting steps in Fig. 3.8 on the left, which corresponds to the location "Cottbus". All models have almost the same MAE in the first step (0.23 m/s on average), with CNN-LSTM showing a slighlty worse performance (0.28 m/s). After this step, ARIMA produces larger errors in comparison to the other models, with MAE values up to 1.53 m/s. In contrast to that, the error increase of CNN-LSTM over the next steps is relatively small compared to the other forecasting models. More precisely, CNN-LSTM exhibits an error increase of 43.43 % from 1 to 12 steps ahead, while the other models – except for ARIMA – have an average increase of 76.65 %. MLR, MPR, HGBR, KNN, and SVR have almost the same MAE over the forecasting horizons, with KNN and HGBR achieving a slightly better performance in the forecasting step 12 (0.38 m/s and 0.39 m/s, respectively). Similar observations appear in Fig. 3.8 on the right, which presents the results for Düsseldorf in summer 2021. However, there is an overall increase in error. For example, the error in the first step rises by 64.37 % on average when moving from Cottbus in winter to Düsseldorf in summer. This increase is mainly due to the change in location from Cottbus to Düsseldorf, which raises the error by approximately 32 % (see Tables 3.1 and A.3). To a lesser extent, the seasonal difference from winter to summer, as already mentioned in Section 3.3.1, contributes an additional 2 % increase.

Since MLR, MPR, HGBR, and KNN do not require extensive training time (see Section 3.3.1), several multi-step-ahead forecasting strategies presented at the beginning of this section have been applied in this work to these methods in order to investigate which strategy performs better for them in terms of accuracy and computational time. The results of this investigation are displayed in Tables A.10-A.15 in

Appendix A.2.2 for Cottbus (winter 2021) and Düsseldorf (summer 2021). All the strategies show the same accuracy metrics for all forecasting methods. However, the Rec strategy requires fewer computational resources than the other strategies, as clearly seen in the tables in Appendix. Specifically, Rec requires up to 134 seconds total CPU time – including forecasting model training and executing the strategy – on a two-day dataset, depending on the forecasting model, while the other strategies require up to 560 seconds.

3.4. Selected Forecasting Models for Energy Management

Based on the results of Section 3.3, MLR, HGBR, and KNN offer a combination of high accuracy and low computational costs, making them well-suited for energy management applications where efficiency is a key consideration. These models demonstrate reliable performance in both one-step- and multi-step-ahead forecasting while ensuring low CPU training time (see Table 3.1). As already mentioned in Chapter 1, a maximum total CPU time of approximately 60 seconds is defined for each RHA iteration in this thesis, including forecasting, surrogate model updates if necessary, and optimization for optimal setpoint definition. These forecasting models fit well within this time constraint, as they enable rapid retraining and leave sufficient margin for the subsequent optimization step, which typically requires a larger share of the computational budget. Therefore, they are considered in the energy management of the conceptual utility system in Chapter 5, where their performance is investigated within this dynamic system. To forecast multiple steps ahead, the Rec strategy is employed, which demonstrates high accuracy and low computational resources.

MPR is excluded from this investigation because it requires additional polynomial features compared to MLR, which increase the model complexity without providing significant improvement in the accuracy over the simple linear model. This indicates that the dependencies of the target variable on the predictors (features) are approximately linear in the considered datasets. ARIMA is also not considered due to its high CPU time during the training process and poor performance in multi-step-ahead forecasting. Moreover, SVR is excluded solely because of its high computational costs.

The last forecasting model considered in this study is the CNN-LSTM. Despite the limitation of high training time, it demonstrates strong performance in multi-step-ahead forecasting. For this reason, it is also considered in the energy management in Chapter 5 to investigate the performance of a DL model in the dynamic system. However, it is trained only at the beginning of the energy management process and not re-trained during each iteration due to its high computational cost.

4. Hybrid Optimization Method

In this work, we consider nonlinear, nonconvex, constrained optimization problems, such as the one introduced in Section 2.4, which have the following form:

$$\min_{\boldsymbol{x}\in\Omega}\Big\{f(\boldsymbol{x})\ \text{ s. t. }\ \boldsymbol{h}(\boldsymbol{x})=\boldsymbol{0}, \boldsymbol{g}(\boldsymbol{x})\leq\boldsymbol{0}, \boldsymbol{x}_{\mathrm{L}}\leq\boldsymbol{x}\leq\boldsymbol{x}_{\mathrm{U}}\Big\}, \tag{4.1}$$

with $\boldsymbol{x}\in\Omega\subseteq\mathbb{R}^n$ the n-dimensional decision variable contained in the set Ω, $\boldsymbol{x}_{\mathrm{L}}$ and $\boldsymbol{x}_{\mathrm{U}}$ the lower and upper bounds of the variables $\boldsymbol{x}$, $f:\mathbb{R}^n\to\mathbb{R}$ the objective function to be minimized, $\boldsymbol{h}:\mathbb{R}^n\to\mathbb{R}^p$ the set of equality constraints, and $\boldsymbol{g}:\mathbb{R}^n\to\mathbb{R}^q$ the set of inequality constraints. A key challenge in numerical optimization is the increasing complexity of the search space as the number of variables grows and the nonlinearity and nonconvexity of the optimization problem increase (Cao et al., 2018; Zhan et al., 2022), making it difficult to find the global optimum. To address these challenges, this chapter presents several state-of-the-art methods and discusses their limitations. To overcome these drawbacks, the novel hybrid optimization method BO-IPOPT, developed in this thesis, is introduced in this chapter, and its performance is evaluated on both test and conceptual use cases.

4.1. State-of-the-Art Methods

A variety of methods have been developed to solve nonlinear, nonconvex optimization problems described by (4.1), which are broadly classified into global and local approaches. Global optimization methods can be further categorized into deterministic and stochastic approaches. Deterministic global methods, as implemented in advanced solvers like BARON (Tawarmalani and Sahinidis, 2002), employ techniques such as enumeration, cutting planes, and bounding regions to systematically eliminate areas that can not contain optimal solutions. However, these methods require substantial computational resources, particularly for nonlinear, nonconvex problems, restricting their applicability to problems with approximately 100 variables (Neumaier et al., 2005). One way to overcome these limitations is by focusing on either linear (Parisio et al., 2014; Ma et al., 2017; Dowling et al., 2017) or linearized (Bischi et al., 2014; Pan et al., 2020; Xu et al., 2020) formulations of the optimization problem. While these approaches enable the use of more efficient linear optimization solvers, they sacrifice realism and require a careful trade-off between accuracy and computational efficiency in the linearization process.

Stochastic global optimization, in contrast, introduces randomness to explore the solution space. Based on how the solution approximation is updated, these methods can be divided into metaheuristics

and surrogate-based techniques. Metaheuristic algorithms, such as GA, simulated annealing, and PSO (Long and Wu, 2014; Ngo et al., 2016; Zapata et al., 2019; Wang et al., 2020a), draw inspiration from natural phenomena to guide the search process. Surrogate-based methods, including BO (Jeong and Obayashi, 2005; Brochu et al., 2010; Berk et al., 2019; Lan et al., 2022; Roberts et al., 2024), consider metamodels to iteratively refine approximations of the objective function and constraints. While stochastic approaches are less prone to getting stuck in local minima, they generally demand a high number of function evaluations (total number of evaluations for f, $\boldsymbol{g}$, and $\boldsymbol{h}$ together). Competitive swarm optimizers (CSO), an advancement of particle swarm optimization, have been introduced to improve the efficiency of metaheuristic methods in high-dimensional optimization. These enhancements include competitive mechanisms for accelerated convergence, a two-stage particle update process, and strategies designed to prevent premature convergence to local optima (Cheng and Jin, 2015; Zhang et al., 2018; Tian et al., 2020; He et al., 2021; Ming et al., 2023). However, further refinements are necessary to improve CSO for tackling complex optimization problems.

Local methods, also known as local search algorithms, focus on identifying the optimal solution within the vicinity of a given starting point. These approaches can be categorized into gradient-free (Larson et al., 2019) (e.g., Nelder-Mead) and gradient-based (Andrei, 2017) (e.g., quasi-Newton methods) techniques, depending on whether they utilize gradient information to update iterates (Nocedal and Wright, 2006). Gradient-free methods are particularly useful for nonsmooth or noisy functions, where derivative information is unreliable, whereas gradient-based approaches are more suited for well-conditioned, high-dimensional problems. Compared to global optimization techniques, local search methods typically achieve faster convergence, requiring fewer function evaluations, especially when the initial point is well-chosen (close to the region of the optimum) or the objective function is locally convex. However, their primary limitation is the tendency to become trapped in local minima, reducing their effectiveness in solving nonlinear, nonconvex optimization problems.

A conceptual extension of local methods into global optimization is the multi-start (MS) approach (Martí et al., 2018), where multiple local searches are initiated from different starting points, and the best local solution – based on the objective function value – is selected as the estimated global optimum. The primary drawback of MS strategies is that multiple starting points may lead to the same local minimum, resulting in redundant computations without providing additional insights. To address this issue, methods such as tunneling, clustering, scatter search, and memory structures (Arora et al., 1995; Ugray et al., 2007; Lasdon and Plummer, 2008; Martí et al., 2019) are employed to enhance the selection of initial points. However, most heuristic MS techniques struggle with high-dimensional problems, making random initialization a common practical choice.

Another promising approach involves combining global stochastic methods with local optimization techniques (Ulder et al., 1990; Chelouah and Siarry, 2003; Kelner et al., 2008; Long and Wu, 2014; Asim et al., 2018), leveraging the strengths of both strategies. These hybrid approaches are typically categorized into sequential and integrative types (Cherki et al., 2019), though classifications may vary

across literature. In sequential methods (Attaviriyanupap et al., 2002; Sivasubramani and Swarup, 2010), a stochastic algorithm is first applied, and its approximate solution serves as the starting point for a local search. This strategy often requires a high number of function evaluations during the stochastic phase to generate a solution sufficiently close to the global optimum. In contrast, integrative methods (Ulder et al., 1990; Kelner et al., 2008; Long and Wu, 2014) apply both stochastic and local techniques simultaneously in an iterative manner, fostering better complementarity between the two. Consequently, integrative approaches generally achieve greater sample efficiency and improved performance. The most widely used hybrid techniques in this category combine metaheuristics with local search (Ulder et al., 1990; Chelouah and Siarry, 2003; Long and Wu, 2014), whereas recent research has increasingly focused on integrating surrogate-based models with gradient-based optimization (Gao et al., 2020; Luo et al., 2022). Surrogate-assisted hybrids offer a more computationally efficient alternative by progressively approximating objective functions, thereby reducing costs compared to metaheuristic-based hybrids. However, their effectiveness in high-dimensional settings remains a challenge due to the growing complexity with an increasing number of samples. To address this issue, recent advancements in high-dimensional Bayesian optimization (HDBO) have introduced strategies that efficiently handle large search spaces.

To improve the efficiency of HDBO, Wang et al. (2016) introduced random embedding Bayesian optimization (REMBO) that projects the high-dimensional search space into a lower-dimensional subspace using random linear embeddings. Another approach, called hashing-enhanced subspace Bayesian optimization (HESBO) (Nayebi et al., 2019), optimizes functions in reduced-dimensional spaces identified through random embeddings and structure learning. Eriksson et al. (2019) proposed trust region Bayesian optimization (TuRBO), a scalable method that partitions the search space into smaller, more manageable regions, enabling efficient high-dimensional optimization. Expanding on the concept of dimensionality reduction, Binois et al. (2020) emphasized the significance of selecting an appropriate low-dimensional domain for global optimization using random embeddings. More recently, Priem et al. (2023) introduced efficient global optimization coupled with random and supervised embedding (EGORSE), a high-dimensional approach that combines random embeddings and supervised learning to enhance search efficiency. However, all these BO techniques have been designed specifically for unconstrained optimization.

Incorporating constraints in high-dimensional settings is both challenging and crucial for many real-world applications. Constraints introduce additional complexity by requiring a careful trade-off between optimization and feasibility, making their formulation and handling in high-dimensional spaces more difficult. While unconstrained methods serve as a solid foundation, recent advancements in GPR have led to promising developments in constrained optimization. Bouhlel et al. (2018) proposed an approach, SEGOKPLS+K, which integrates the SEGO algorithm (super efficient global optimization, Sasena et al. (2001)) with kriging models and partial least squares (KPLS) to address the challenges of high-dimensional constrained problems by effectively reducing dimensionality. The "+K" denotes an optional step consisting of a local optimization of the kriging model to enhance its accuracy. Additionally, Eriksson and Poloczek

(2021) extended the TuRBO framework to handle constrained optimization problems, introducing scalable constrained BAYESIAN optimization (SCBO).

4.2. Novel Hybrid Method

This work presents BO-IPOPT, a novel hybrid optimization method that integrates the surrogate-based BO with the Interior Point OPTimizer (IPOPT) for local refinement. BO-IPOPT combines the global exploration capabilities of BO with the local efficiency of IPOPT, ensuring that all constraints are satisfied throughout the optimization process. More precisely, BO explores the entire input space on a global scale, taking into account smooth variations and ignoring details (local variations). Then, IPOPT focuses on the regions of interest provided by the BO part. The goal is to develop a method that combines high accuracy, robustness (i.e., consistent performance across repeated runs), and computational efficiency, especially for problems of higher dimensions ($\gtrsim 100$ decision variables), where conventional approaches face difficulties.

Unlike state-of-the-art BO approaches that construct precise surrogate models for both the objective function and constraints, our method employs a single surrogate model to provide a coarse approximation of the cost function. Additionally, unlike traditional methods that incur extra computational costs by learning their hyperparameters during optimization, our approach avoids this overhead by fixing them to predefined values. These aspects – using a single surrogate model and avoiding costly parameter learning – support the global guidance of BO within the hybrid method while reducing the computational effort.

By integrating BO with IPOPT, the hybrid approach effectively incorporates feasibility checks within the optimization process, ensuring that the constraints are considered throughout. This feature differentiates BO-IPOPT from unconstrained optimization methods such as BOwLS (BAYESIAN optimization with local search, Gao et al. (2020)), which focus exclusively on enhancing the objective function accuracy without addressing constraint handling. By further developing BOwLS for constrained optimization, BO-IPOPT establishes a structured framework that balances exploration and exploitation while maintaining feasibility requirements in constrained optimization tasks.

To better illustrate the interaction between BO and IPOPT in the proposed hybrid BO-IPOPT, we first provide a brief introduction of each method in its standalone form in Sections 4.2.1 and 4.2.2, before discussion their hybridization in Section 4.2.3.

4.2.1. BAYESIAN Optimization

The BO framework is a global optimization method designed for expensive-to-evaluate black-box functions. It builds surrogate models based on samples at candidate solutions $\boldsymbol{x}_K$, where K is the iteration index, typically using the GAUSSIAN process (GP) model (Rasmussen and Williams, 2005; Hebbal et al., 2023). This surrogate model enables the analytical computation of both the estimated cost function and its associated uncertainty, facilitating a trade-off between exploitation – sampling in regions where the

surrogate predicts optimal values – and exploration – sampling in areas with the highest uncertainty. The selection of new samples is guided by an acquisition function.

Several methods have been proposed to incorporate constraints into BO. One approach weights the acquisition function by the probability of satisfying each constraint (Gelbart et al., 2014; Gardner et al., 2014). Another method, applicable to decoupled constraints, leverages the predictive entropy search (Hernández-Lobato et al., 2015) based on expected information gain. Pelamatti et al. (2024) employ a multi-output GAUSSIAN process to optimally determine which constraint should be evaluated at each step, while Lam and Willcox (2017) introduce a "look-ahead" strategy that selects the next evaluation point to maximize the long-term feasible reduction of the objective function. In Ariafar et al. (2019), the alternating direction method of multipliers is applied, modifying the objective function with auxiliary variables, whereas Gramacy et al. (2016), within the framework of AL, reformulate the constrained optimization problem as a sequence of unconstrained subproblems that are solved iteratively. An alternative strategy is presented in Picheny (2014), where a step-wise uncertainty reduction technique is employed to balance exploration and local refinement near the boundaries. All these methods are designed to handle only inequality constraints.

A common approach to handle both equality and inequality constraints is to convert equality constraints $\boldsymbol{h}(\boldsymbol{x})$ into inequality constraints of the form $\boldsymbol{h}(\boldsymbol{x}) \geq \mathbf{0}$ and $-\boldsymbol{h}(\boldsymbol{x}) \geq \mathbf{0}$. However, this transformation increases the number of constraints and may lead to constraint qualification issues (Nocedal and Wright, 2006). A more advanced method that directly handles both equality and inequality constraints without transformation is ALBO (Picheny et al., 2016), which integrates the unconstrained BO with the AL framework. Additionally, Priem et al. (2020) extended the SEGO method, which builds a separate surrogate model for each inequality constraint, to consider equality constraints as well. To improve the modeling of constraints in SEGO, Priem et al. (2020) incorporates a fixed scalar upper trust bound, originally introduced in Lam et al. (2015), Lam and Willcox (2017), and Priem et al. (2019), to provide better exploration of the design space. Building on this idea, Lu and Paulson (2022) proposed a modified upper trust approach, where an exact penalty function is integrated into the lower confidence bound acquisition function to efficiently address expensive constrained black-box optimization problems.

Existing approaches for managing both equality and inequality constraints, such as ALBO and SEGO, typically rely on constructing multiple surrogate models for different constraints. This results in substantial computational overhead, particularly in high-dimensional settings with a large number of constraints. Additionally, this strategy increases the number of hyperparameters requiring tuning and raises the risk of overfitting, as each surrogate model is trained independently. To overcome these challenges, we introduce a single surrogate model that simultaneously approximates both the objective function and constraints. Although single-model approaches are often considered less precise, our hybrid framework utilizes BO for global exploration and IPOPT for local refinement, thereby removing the necessity for highly accurate predictions. This enables the surrogate model to focus on providing effective global guidance for optimization.

To incorporate equality and inequality constraints into a single objective function, we adopt two strategies. First, we augment the objective function with penalty terms for both constraint types (Long and Wu, 2014):

$$u(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\delta}^\top \boldsymbol{h}(\boldsymbol{x})^2 + \boldsymbol{\tau}^\top \max(\mathbf{0}, \boldsymbol{g}(\boldsymbol{x}))^2, \tag{4.2}$$

where $\boldsymbol{\delta}, \boldsymbol{\tau} \geq \mathbf{0}$ are penalty weight vectors that quantify constraint violations. This penalty-based formulation has been already explored in Kyriakidis et al. (2024). Second, we adopt the AL framework (Picheny et al., 2016), following the ALBO approach, in which slack variables are introduced to convert inequality into equality constraints. This allows for an iterative process where a sequence of unconstrained optimization problems is solved progressively, simplifying the overall optimization process. The AL is formulated as follows:

$$u(\boldsymbol{x}, \boldsymbol{s}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}_{\boldsymbol{g}}^\top (\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s}) + \boldsymbol{\lambda}_{\boldsymbol{h}}^\top \boldsymbol{h}(\boldsymbol{x}) + \frac{1}{2\rho} \left[\sum_{w=1}^{q} (g_w(\boldsymbol{x}) + s_w)^2 + \sum_{r=1}^{p} h_r(\boldsymbol{x})^2 \right], \tag{4.3}$$

where $\rho > 0$ represents a penalty parameter controlling constraint violations, while $\boldsymbol{\lambda_g} \in \mathbb{R}_+^q$ and $\boldsymbol{\lambda_h} \in \mathbb{R}_+^p$ are the LAGRANGE multiplier vectors corresponding to inequality and equality constraints, respectively. The slack variable vector $\boldsymbol{s}$ is defined as:

$$\boldsymbol{s} = \max\{\mathbf{0}, -\boldsymbol{\lambda}_{\boldsymbol{g}} \rho - \boldsymbol{g}(\boldsymbol{x})\}. \tag{4.4}$$

At each outer iteration K, the LAGRANGE multipliers, penalty parameter, and slack variables are iteratively updated, forming a dynamic process described in more detail in Appendix A.3.1. This approach was chosen because its slack variable formulation closely aligns with that of IPOPT, allowing for a smooth integration between the two frameworks. Additionally, the iterative update mechanism reduces the number of hyperparameters requiring manual tuning, simplifying the optimization process. However, applying this iterative approach to high-dimensional problems remains challenging, as the AL is not solved analytically and often encounters a large number of constraints, particularly in infeasible regions. Despite these challenges, the AL framework performed effectively in all examined test cases within the BO-IPOPT methodology, as further discussed in Section 4.4.

In GPR, we employ one of the most commonly used kernels, namely the GAUSSIAN RBF kernel (Bartoli et al., 2019) due to its proven performance, simplicity, and effectiveness. It is defined as:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(\frac{-d(\boldsymbol{x}, \boldsymbol{x}')^2}{2l^2} \right), \tag{4.5}$$

where l represents the kernel's length scale and $d(\cdot, \cdot)$ denotes the EUCLIDEAN distance between points in the domain Ω. To ensure numerical stability and prevent overfitting, the kernel matrices undergo regularization using TIKHONOV regularization, incorporating a ridge parameter α. As initial sample points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}\}$ and their associated objective values $\boldsymbol{u} = \big[u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_{N_0})\big]$ are collected, the

GPR updates the underlying GP using standard conditioning rules (Rasmussen and Williams, 2005) so that the predictions $\bar{X} = \{\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_{N_\mathrm{c}}\}$ become $\hat{\boldsymbol{u}}(\bar{X}) \sim \mathrm{GP}(\boldsymbol{\mu}(\bar{X}), \Sigma(\bar{X}))$ with mean $\boldsymbol{\mu}(\bar{X})$ and predictive covariance matrix $\Sigma(\bar{X})$ given by:

$$\begin{aligned} \boldsymbol{\mu}(\bar{X}) &= K_*^\top \big(K + \alpha I\big)^{-1} \boldsymbol{u}^\top, \\ \Sigma(\bar{X}) &= K_{**} - K_*^\top \big(K + \alpha I\big)^{-1} K_*. \end{aligned} \tag{4.6}$$

Here, $K = k(X, X)$, $K_* = k(X, \bar{X})$, and $K_{**} = k(\bar{X}, \bar{X})$ are kernel matrices, I describes the identity matrix, and α is the regularization parameter. The computational complexity is dominated by the inversion of the correlation matrix $K + \alpha I$, making BO more challenging as more samples are considered in the GP model. The main hyperparameters governing the GP model are the length scale l and the regularization parameter α. The first determines the smoothness of the function, while the second regularizes the sensitivity of the regression toward noise in the case of stochastic objective functions.

For the acquisition function, there are several commonly used functions in BO, including the expected improvement (EI), the Watson and Barnes 2nd acquisition criterion (WB2) and its smoothed variant (WB2S), the probability of improvement (PI), and the lower confidence bound (LCB), as discussed in Watson and Barnes (1995), Brochu et al. (2010), Bartoli et al. (2019), and Greenhill et al. (2020). The choice of acquisition function typically depends on the specific requirements of the optimization problem. In this work, we focus on EI due to its widespread adoption in the field and its well-established effectiveness in balancing exploration and exploitation. For a new sample $\bar{\boldsymbol{x}}$, EI is defined as follows:

$$\mathrm{EI}(\bar{\boldsymbol{x}}|\hat{u}) = \begin{cases} (\hat{u}(\boldsymbol{x}^+) - \mu(\bar{\boldsymbol{x}}) - \xi)\Phi(Z) + \sigma(\bar{\boldsymbol{x}})\phi(Z), & \text{if } \sigma(\bar{\boldsymbol{x}}) > 0, \\ 0, & \text{if } \sigma(\bar{\boldsymbol{x}}) = 0, \end{cases} \tag{4.7}$$

with

$$Z = \begin{cases} \frac{\hat{u}(\boldsymbol{x}^+) - \mu(\bar{\boldsymbol{x}}) - \xi}{\sigma(\bar{\boldsymbol{x}})}, & \text{if } \sigma(\bar{\boldsymbol{x}}) > 0, \\ 0, & \text{if } \sigma(\bar{\boldsymbol{x}}) = 0. \end{cases} \tag{4.8}$$

In this formulation, $\mu(\bar{\boldsymbol{x}})$ and $\sigma(\bar{\boldsymbol{x}})$ represent the mean and standard deviation of the GPR predictions, respectively. The functions Φ and ϕ correspond to the cumulative and probability density functions of the standard normal distribution. The first term of EI promotes exploitation by favoring points with high predicted performance, while the second term encourages exploration by considering areas with high uncertainty. The parameter ξ acts as a threshold that determines the minimum expected improvement necessary to justify exploration, serving as a key hyperparameter in BO. A higher ξ value leads to increased exploration across the search space.

In traditional BO methods, the selection of new candidate solutions is typically driven by maximizing the EI function (Brochu et al., 2010; Malu et al., 2021; Lu and Paulson, 2022). The standard BO algorithm

begins with an initial set of randomly chosen candidate solutions and iterates between updating the GPR model and optimizing the EI function to determine the next candidate, as outlined in Algorithm 1.

Algorithm 1 Classical BO Algorithm

Input: Length scale l; regularization term α; exploration ξ; number of initialization points N_0; number of candidates N_c considered in EI; penalty weight vectors $\boldsymbol{\delta}$, $\boldsymbol{\tau}$; Lagrange multiplier vectors $\boldsymbol{\lambda_g}$, $\boldsymbol{\lambda_h}$; number of outer iterations K

Generate an initial set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}\}$ in Ω
Evaluate $y_{i_0} = u(\boldsymbol{x}_{i_0})$ or $y_{i_0} = u(\boldsymbol{x}_{i_0}, \boldsymbol{s}_{i_0})$ for $i_0 = 1{:}N_0$
Let $D_0 = \{(\boldsymbol{x}_{i_0}, y_{i_0})\}_{i_0=1}^{N_0}$
Construct a GP model $\hat{u}_0$ from D_0
$z = N_0, j = 0$
while $j < K$ **do**
    Generate a new group of points $\{\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_{N_\mathrm{c}}\}$ in Ω
    $\boldsymbol{x}_{z+1} \in \arg\max_{\boldsymbol{x} \in D_j} \mathrm{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$ for $i = 1{:}N_\mathrm{c}$
    $y_{z+1} = u(\boldsymbol{x}_{z+1})$
    $D_{j+1} = D_j \cup \{(\boldsymbol{x}_{z+1}, y_{z+1})\}$
    Update GP model $\hat{u}_{j+1}$ from D_{j+1}
    $z := z + 1, j := j + 1$
end while
return $y_\mathrm{min} = \min\{y_j\}_{j=1}^{K}$

4.2.2. Interior Point OPTimizer

The IPOPT solver from COIN-OR (computational infrastructure for operations research) is one of the most widely used open-source tools for solving high-dimensional nonlinear, nonconvex, constrained optimization problems. Originally developed by Wächter (2002) and Wächter and Biegler (2006), IPOPT employs a primal-dual interior-point method combined with a filtered line-search approach to identify a local solution. The algorithm requires all functions in (4.1) to be at least once and ideally twice continuously differentiable.

IPOPT tackles the optimization problem in (4.1) by approximately solving a sequence of barrier problems, iteratively converging to the optimal solution while remaining within the feasible set. To handle inequality constraints, it introduces slack variables, reformulating them as equalities: $\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s} = \boldsymbol{0}$ with $\boldsymbol{s} \geq \boldsymbol{0}$. Defining $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{s})$ and $\boldsymbol{c}(\tilde{\boldsymbol{x}}) = (\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{s})$, the problem in (4.1) is rewritten as:

$$\min_{\tilde{\boldsymbol{x}} \in \tilde{\Omega}} \Big\{ f(\tilde{\boldsymbol{x}}) \;\text{ s. t. }\; \boldsymbol{c}(\tilde{\boldsymbol{x}}) = \boldsymbol{0}, \tilde{\boldsymbol{x}}_\mathrm{L} \leq \tilde{\boldsymbol{x}} \leq \tilde{\boldsymbol{x}}_\mathrm{U} \Big\}. \tag{4.9}$$

This formulation is adjusted by incorporating a natural logarithmic barrier term into the objective function to handle the boundaries of the variable vector $\tilde{\boldsymbol{x}}$. In other words, IPOPT seeks to determine a Karush-Kuhn-Tucker optimality point for (4.9) by solving a series of problems with a progressively decreasing barrier parameter $\mu \in \mathbb{R}^+$, given by:

$$\min_{\tilde{\boldsymbol{x}}(\mu)\in\tilde{\Omega}} \Big\{ f(\tilde{\boldsymbol{x}}) - \mu \Big[\sum_{i=1}^{n+q} \ln(\tilde{x}_i - \tilde{x}_{\mathrm{L}}) + \sum_{i=1}^{n+q} \ln(\tilde{x}_{\mathrm{U}} - \tilde{x}_i) \Big] \ \text{ s. t. } \ \boldsymbol{c}(\tilde{\boldsymbol{x}}) = \mathbf{0} \Big\}. \tag{4.10}$$

The nonlinear primal-dual problems in (4.10) are iteratively solved for different values of the barrier parameter. As $\mu \to 0$, the computed solution $\tilde{\boldsymbol{x}}^\star(\mu)$ approaches the optimal solution $\tilde{\boldsymbol{x}}^\star$, which also solves (4.1). Each subproblem uses the previously computed solution as the initial point, and a NEWTON-type algorithm with line search is applied, involving the solution of indefinite sparse symmetric linear systems. The overall performance in terms of runtime, accuracy, and robustness strongly depends on the properties of the selected sparse linear solver. The open-source version of IPOPT utilizes MA27 for this purpose. In addition, IPOPT implements a specialized HESSIAN regularization technique to prevent maximizers and saddle points. For further algorithmic details, we refer the reader to Wächter (2002) and Wächter and Biegler (2006).

In summary, IPOPT is a highly efficient solver for finding local minima of (4.1). However, as with any local optimization method, its performance is strongly influenced by the initial starting point.

4.2.3. BO-IPOPT

The proposed BO-IPOPT method, first introduced in Kyriakidis et al. (2024) and further enhanced in Kyriakidis et al. (2025b), is illustrated in Algorithm 2, which consists of two main components: an initialization step and a loop over the number of outer iterations K. Below, we provide a detailed explanation of the procedure.

In the initialization step (lines 1-4), a set of initial points is generated using an appropriate strategy and then evaluated with the augmented objective function. These evaluations are then used to construct the initial GP model. Unlike the traditional BO, which separately models the objective function and constraints, BO in the hybrid BO-IPOPT framework enables global exploration, while IPOPT ensures local refinement and constraint satisfaction at each iteration. It is important to note that l and a are incorporated into the GP model, while the exploration parameter ξ is included in the EI function. Additionally, the input vectors $\boldsymbol{\delta}$, $\boldsymbol{\tau}$, $\boldsymbol{\lambda_g}$, and $\boldsymbol{\lambda_h}$ are required for evaluating the objective function in line 2 of Algorithm 2, as defined by (4.2) or (4.3). For the GP model, it is also recommended to normalize the input and output data to lie within the interval $[0, 1]$, which improves the numerical stability and prediction accuracy.

The algorithm alternates the BO and IPOPT steps until the maximum number of outer iterations K is reached (lines 6-13). At each iteration, candidate points with the highest EI values are selected by evaluating the acquisition function – either over a discrete set of candidates or the entire continuous space – similar to the EGO method. In either case, multiple candidate points can be chosen for each outer iteration K and serve as starting points for IPOPT, which then optimizes the real objective function and constraints. It should be noted that the number of best candidates selected at each outer iteration should be chosen considering that the GPR scales cubically with the number of samples involved due to the inversion of the covariance matrix via CHOLESKY factorization. Nevertheless, in all the investigated benchmark problems,

Algorithm 2 Hybrid Method BO-IPOPT

Input: Length scale l; regularization term α; exploration ξ; number of initialization points N_0; number of candidates N_c considered in EI; number of best candidates N_bc; penalty weight vectors $\boldsymbol{\delta}$, $\boldsymbol{\tau}$; Lagrange multiplier vectors $\boldsymbol{\lambda_g}$, $\boldsymbol{\lambda_h}$; number of outer iterations K

1: Generate an initial set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}\}$ in Ω
2: Evaluate $y_{i_0} = u(\boldsymbol{x}_{i_0})$ or $y_{i_0} = u(\boldsymbol{x}_{i_0}, \boldsymbol{s}_{i_0})$ for $i_0 = 1{:}N_0$
3: Let $D_0 = \{(\boldsymbol{x}_{i_0}, y_{i_0})\}_{i_0=1}^{N_0}$
4: Construct a GP model $\hat{u}_0$ from D_0
5: $z = N_0, j = 0$
6: **while** $j < K$ **do**
7: Generate a new group of points $\{\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_{N_\mathrm{c}}\}$ in Ω
8: $\{\boldsymbol{x}_{z+1}, \ldots, \boldsymbol{x}_{z+N_\mathrm{bc}}\} \in \underset{\boldsymbol{x} \in D_j}{\arg\max}\, \mathrm{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$ for $i = 1{:}N_\mathrm{c}$
9: Solve $[y_k^\star, \boldsymbol{x}_k^\star] = \mathrm{IPOPT}(\boldsymbol{x}_k)$ for $k = z+1{:}z+N_\mathrm{bc}$
10: $D_{j+1} = D_j \cup \{(\boldsymbol{x}_{z+1}^\star, y_{z+1}^\star), \ldots, (\boldsymbol{x}_{z+N_\mathrm{bc}}^\star, y_{z+N_\mathrm{bc}}^\star)\}$
11: Update GP model $\hat{u}_{j+1}$ from D_{j+1}
12: $z := z + N_\mathrm{bc}, j := j + 1$
13: **end while**
14: **return** $y_\mathrm{min} = \min\{y_j^\star\}_{j=1}^{KN_\mathrm{bc}}$

the computational cost of evaluating the objective function and constraints significantly outweighs the cost of the GPR. As a result, this overhead is not considered particularly relevant.

A notable challenge in the current implementation is the potential selection of points from the same region, which may cause the algorithm to converge to the same local optimum. However, this was not observed in the test and use cases investigated in this work, as demonstrated in Section A.3.12. This issue could nonetheless be mitigated using the multipoint expected improvement (q-EI) (Wang et al., 2020b), which optimizes EI across a batch of points with the batch size q, while considering correlations between them. Future work will explore the integration of q-EI into the BO-IPOPT framework. Nevertheless, a key distinction in BO-IPOPT is that while BO updates the GPR with EI-selected candidates, IPOPT refines these candidates, meaning that the GPR is ultimately updated using locally optimal solutions, which may differ significantly from those identified by EI.

At the end of the optimization process (line 14), the algorithm returns the minimum objective function value from the set of points D_K, excluding the initial dataset D_0. This ensures that the final solution y_min represents at least a local optimum of (4.1) and, ideally, the global minimum, while satisfying all constraints.

The BO-IPOPT algorithm exclusively uses feasible solutions (line 9). If IPOPT produces an infeasible local solution, the next-best candidate is selected, and IPOPT is restarted. This strategy has successfully led BO-IPOPT to feasible solutions in all cases in this work, and situations where the algorithm failed have not been encountered. In future work, for problems with a small feasible region, additional techniques such as repair functions or constraint relaxation (Samanipour and Jelovica, 2020; Sankaran, 1993) could be considered. Moreover, the hybrid approach recommends selecting multiple best candidates per iteration

(line 8), unlike the classical BO. This increases the probability of finding the global optimum faster. At the same time, the computational costs of the optimizer naturally increase as more IPOPT searches are performed sequentially.

Given that BO-IPOPT requires multiple local optimizations at each iteration for $N_{\text{bc}} > 1$, its computational cost may become significant for high-dimensional problems. To mitigate this, we propose parallelizing the evaluation of the selected points, which significantly reduces the computational time by leveraging independent evaluations. PYTHON's MULTIPROCESSING module (Foundation, 2023) – specifically the POOL function – can be used to assign tasks to multiple worker processes dynamically. This method reduces the overall CPU time to approximately the runtime of a single IPOPT evaluation, plus some communication and synchronization overhead. This parallelization is particularly beneficial for high-dimensional problems, but it offers diminishing returns for smaller problems, where communication costs may outweigh efficiency gains.

It is also important to highlight that the current BO-IPOPT formulation is limited to continuous variables. Future work will explore its extension to mixed-integer problems, where the BO part can leverage adaptive dimension reduction (Saves et al., 2023) to address integer variables, while IPOPT can incorporate relaxation techniques, branch-and-bound methods, outer approximation, or extended cutting planes (Belotti et al., 2013) to handle integer variables.

The BO-IPOPT method offers two key advantages. First, IPOPT enhances the efficiency of BO by moving candidate solutions toward optimal locations. If these solutions are local optima, the EI evaluation ensures global exploration by searching in unexplored regions of the solution space, and the regression is refined when necessary. If a global minimum is found, EI prioritizes the sampling around it in future iterations. Second, IPOPT benefits from the good starting points provided by BO. Note that any local solver could replace IPOPT; however, it was chosen for its efficiency, robustness, and open-source availability.

An important aspect of the BO component in our hybrid method is how it handles the constraints of the underlying optimization problem. The best candidates are evaluated in the classical BO approach based on the selected augmented objective function. In our formulation, a local solution $\boldsymbol{x}^\star$ for (4.2) and (4.3) ensures that $u(\boldsymbol{x}_k^\star) \equiv f(\boldsymbol{x}_k^\star)$ and $u(\boldsymbol{x}_k^\star, \boldsymbol{s}(\boldsymbol{x}_k^\star)) \equiv f(\boldsymbol{x}_k^\star)$, respectively. Since local solutions satisfy all constraints (rounding errors are assumed negligible here), both formulations lead to the same result (see also Appendix A.3.1 for the AL framework). However, in the initialization phase, the constraints are most likely not fulfilled, meaning that the initial GP model derived from the initialization points is strongly influenced by how the constraints are considered at this stage. In addition, the chosen constraint-handling approach affects the GP model at each outer loop, as the initial dataset D_0, which contains the initialization points, remains a subset of D_j and is continuously considered in the acquisition function EI (see line 8).

Since BO in this work guides IPOPT rather than being used as a standalone optimizer as usual, its techniques for sampling point selection, constraint handling, and hyperparameter settings need adjustment. These enhancements aim to improve the global exploration and computational efficiency of BO, as well as its integration with the local search capabilities of IPOPT.

4.3. Benchmark Problems

To evaluate the performance of the hybrid BO-IPOPT, we apply this method to four nonlinear, nonconvex, constrained optimization problems chosen to reflect variations in problem size, nonconvexity, and nonlinearity. These problems include two benchmark test cases and two conceptual use cases, namely the well-known dynamic economic dispatch (DED) problem and the optimal operation of a RSG system.

The performance of optimization algorithms is typically evaluated using benchmark problems, which can be categorized based on the number of variables n. Here, we define small-scale problems as those with $n \leq 100$, medium-scale as $100 < n \leq 5{,}000$, and large-scale as $n > 5{,}000$. This work focuses on small- and medium-scale problems, where the functions in (4.1) are explicitly known, allowing for efficient gradient evaluation. Focusing on this range is also natural because most standard benchmark problems fall within it (Liang et al., 2006; Santra et al., 2020; Eriksson and Poloczek, 2021), providing analytically defined objective functions and constraints. However, we specifically consider constrained problems that combine nonlinearity, nonconvexity, and high dimensionality, making them particularly challenging for local search methods. Below, the four nonlinear, nonconvex, constrained optimization problems (two test and two conceptual use cases) considered in this work are briefly introduced; for more details, the reader is referred to the original papers.

4.3.1. Simple Test Problem

The first artificial test problem, originally proposed by Sakawa and Yauchi (2000), is formulated as follows:

$$\begin{aligned}\min f(\boldsymbol{x}) = {} & x_1^3 + (x_2-5)^2 + 3(x_3-9)^2 - 12x_3 + 2x_4^3 + 4x_5^2 + (x_6-5)^2 - 6x_7^2 \\ & + 3(x_7-2)x_8^2 - x_9x_{10} + 4x_9^3 + 5x_1x_3 - 3x_1x_7 + 2x_8x_7,\end{aligned} \tag{4.11}$$

subject to

$$-3(x_1-2)^2 - 4(x_2-3)^2 - 2x_3^2 + 7x_4 - 2x_5x_6x_8 + 120 \geq 0, \tag{4.12}$$

$$-5x_1^2 - 8x_2 - (x_3-6)^2 + 2x_4 + 40 \geq 0, \tag{4.13}$$

$$-x_1^2 - 2(x_2-2)^2 + 2x_1x_2 - 14x_5 - 6x_5x_6 \geq 0, \tag{4.14}$$

$$-0.5(x_1-8)^2 - 2(x_2-4)^2 - 3x_5^2 + x_5x_8 + 30 \geq 0, \tag{4.15}$$

$$3x_1 - 6x_2 - 12(x_9-8)^2 + 7x_{10} \geq 0, \tag{4.16}$$

$$4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 105, \tag{4.17}$$

$$10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0, \tag{4.18}$$

$$-8x_1 + 2x_2 + 5x_9 - 2x_{10} \leq 12, \quad \boldsymbol{x} \in [-5, 10]^{10}. \tag{4.19}$$

The optimization problem (4.11)-(4.19) represents a small-scale, nonlinear, and nonconvex problem with 10 dimensions, constrained solely by inequalities. The problem has multiple local minima and one global minimum $\boldsymbol{x}^\star$, with an objective value of approximately $f(\boldsymbol{x}^\star) \approx -216.65649$ efficiently computed by the BARON solver in this thesis. Additionally, several inequalities are active at $\boldsymbol{x}^\star$. The nonconvexities in this problem arise mainly from cubic and bilinear terms in the objective function and from bilinear terms in the constraints.

4.3.2. Constrained ACKLEY Function

The second test problem examined in this study is the well-known ACKLEY function, commonly used to assess the performance of optimization algorithms. This function is nonlinear, nonconvex, and highly multimodal, with numerous local minima and a single global minimum. Originally designed as an unconstrained problem, it has been adapted into a constrained optimization problem by incorporating two inequality constraints, as described in Eriksson and Poloczek (2021). The resulting n-dimensional constrained optimization problem is expressed as follows:

$$\min f(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, \tag{4.20}$$

subject to

$$\sum_{i=1}^{n} x_i \leq 0, \quad \|\boldsymbol{x}\|_2 - 5 \leq 0, \quad \boldsymbol{x} \in [-5, 10]^n. \tag{4.21}$$

For the experimental study, we consider a small-scale version of the problem by setting $n = 100$. The optimization problem has a global minimum of $f(\boldsymbol{x}^\star) = 0$ at $\boldsymbol{x}^\star = \mathbf{0}$. The problem is nonconvex due to the highly multimodal objective function.

4.3.3. Dynamic Economic Dispatch

The DED problem is one of the most significant and widely studied nonlinear, nonconvex, constrained optimization problems in practical applications. In this work, it serves as the third use case. The main objective of DED is to determine the optimal scheduling of committed power generators over a specified time horizon while satisfying load demand and adhering to various technical constraints. In other words, the goal is to minimize the total cost of electricity generation. The optimization problem, considering valve-point effects (Attaviriyanupap et al., 2002; Balamurugan and Subramanian, 2007; Ravikumar Pandi and Panigrahi, 2011; Yadav and Deep, 2014; Santra et al., 2020), is formulated as follows:

$$\min f(\boldsymbol{P}) = \sum_{t=1}^{T} \sum_{i=1}^{N_{\mathrm{G}}} a_i P_{it}^2 + b_i P_{it} + c_i + \left| e_i \sin\left(f_i \left(P_{it}^{\min} - P_{it}\right)\right)\right|, \tag{4.22}$$

subject to

$$\sum_{i=1}^{N_{\mathrm{G}}} P_{it} = P_{\mathrm{D}t}, \ \ \forall t \in \mathcal{T}, \tag{4.23}$$

$$P_i^{\min} \leq P_{it} \leq P_i^{\max}, \ \ \forall i \in \mathcal{I}, \ \ \forall t \in \mathcal{T}, \tag{4.24}$$

$$P_{it} - P_{i(t-1)} \leq \mathrm{UR}_i, \ \ \forall i \in \mathcal{I}, \ \ P_{i(t-1)} - P_{it} \leq \mathrm{DR}_i, \ \ \forall i \in \mathcal{I}, \tag{4.25}$$

with $\mathcal{I} = \{1, \ldots, N_{\mathrm{G}}\}$ and $\mathcal{T} = \{1, \ldots, T\}$. The objective function in (4.22) consists of a nonsmooth formulation, combining a quadratic polynomial and an absolute sinusoidal term. The nonconvexity of the problem arises from the valve-point effect in the objective function. Equalities and inequalities, all of which are linear, are specified by power balances (4.23), generator capacities (4.24) and unit ramp rate limits (4.25). For the typical 24-hour operation of a 5-unit, 10-unit, and 30-unit system (i.e., $N_{\mathrm{G}} = 5, 10,$ and 30), the corresponding problem sizes are 120, 240, and 720, respectively. As a result, all three DED cases are classified here as medium-scaled problems. The scenario input data for the corresponding cost function coefficients a_i, b_i, c_i, e_i, f_i, maximum $P_i^{\max}$ and minimum $P_i^{\min}$ power outputs, ramp up UR_i and down DR_i rate, and the load demand $P_{\mathrm{D}t}$ are provided in Appendix A.3.4.

4.3.4. Renewable Steam Generation

The fourth use case focuses on the operational optimization of an industrial energy supply system for RSG, as recently proposed in Walden et al. (2023). The objective of the optimization problem is to minimize operating costs while accounting for fluctuations in wind energy availability and electricity prices. The discretized nonlinear, nonconvex, constrained optimization problem is written as follows:

$$\min \ f(\boldsymbol{P}_{\mathrm{grid}}) = \sum_{k=1}^{n} P_{\mathrm{grid}}^k g_{\mathrm{grid}}^k \Delta t, \tag{4.26}$$

subject to

$$P_{\mathrm{grid}}^k + P_{\mathrm{WT}}^k = 3F_{\mathrm{HTHP}}\big(T_{\mathrm{I}}^k, \dot{m}_{\mathrm{I}}^k, T_{\mathrm{III}}^k, R^k\big), \tag{4.27}$$

$$T_{\mathrm{II}}^k = F_{\mathrm{HTHX}}\big(T_{\mathrm{I}}^k, \dot{m}_{\mathrm{I}}^k, T_{\mathrm{III}}^k, R^k\big), \ \ T_{\mathrm{IV}}^k = F_{\mathrm{LTHX}}\big(T_{\mathrm{I}}^k, \dot{m}_{\mathrm{I}}^k, T_{\mathrm{III}}^k, R^k\big), \tag{4.28}$$

$$T_2^k = T_{\mathrm{II}}^k \beta_1^k + T_1^k\big(1 - \beta_1^k\big), \ \ T_{\mathrm{I}}^k = T_3^k \beta_2^k + T_4^k\big(1 - \beta_2^k\big), \tag{4.29}$$

$$T_2^k = 201.92 + \tfrac{1{,}819.32}{3\dot{m}_{\mathrm{II}}^k}, \ \ T_3^k = 196.3 - \tfrac{188.4}{3\dot{m}_{\mathrm{I}}^k}, \tag{4.30}$$

$$\dot{Q}_{\mathrm{s,ch}}^k = 3\dot{m}_{\mathrm{II}}^k c_{p,\mathrm{f}}\big(T_{\mathrm{II}}^k - T_1^k\big)\big(1 - \beta_1^k\big), \ \dot{Q}_{\mathrm{s,dch}}^k = 3\dot{m}_{\mathrm{I}}^k c_{p,\mathrm{f}}\big(T_4^k - T_3^k\big)\big(1 - \beta_2^k\big), \tag{4.31}$$

$$\dot{Q}^k_{\text{s,ch}}\dot{Q}^k_{\text{s,dch}} \leq \gamma, \tag{4.32}$$

$$T^k_1 = T^k_{\text{II}} - \epsilon_{\text{ch}}\big(T^k_{\text{II}} - T^{k-1}_{\text{s}}\big),\ \ T^k_4 = T^k_3 - \epsilon_{\text{dch}}\big(T^k_3 - T^{k-1}_{\text{s}}\big), \tag{4.33}$$

$$T^k_{\text{s}} = T^{k-1}_{\text{s}} + \tfrac{\dot{Q}^k_{\text{s,ch}} - \dot{Q}^k_{\text{s,dch}}}{m_{\text{s}} c_{p,\text{s}}}\Delta t,\ \ T^0_{\text{s}} = \widetilde{T}_0,\ \ T^n_{\text{s}} = T^0_{\text{s}}, \tag{4.34}$$

$$T^k_{\text{I}} \in [177, 250],\ \ \dot{m}^k_{\text{I}} \in [5, 16],\ \ T^k_{\text{III}} \in [60, 100],\ \ R^k \in [0.8, 1.53], \tag{4.35}$$

for $k = 1, \ldots, n$ and with time step size Δt. The objective function in (4.26) is linear and depends on the grid power consumed by the high-temperature heat pump. All the constraints in (4.27)-(4.32) are nonlinear due to the nonlinear surrogate models for the system components, bypass mechanisms, and heat flows during charging and discharging, including interactions between charging and discharging, making the problem nonconvex. Linear constraints in (4.33)-(4.34) describe the effectiveness of the thermal energy storage and the boundary conditions for the storage temperature. For a typical one-day system operation with hourly time steps, the problem results in a total dimension of 408, classifying it as a medium-scale problem. The scenario input data, including wind turbine energy production P^k_{WT} and electricity prices g^k_{grid}, is provided in Appendix A.3.4.

4.4. Results and Analysis

The analysis of the BO-IPOPT method in this study is divided into two main parts. First, several aspects of the BO component are explored to improve both the global exploration and computational efficiency of this hybrid method, as well as its integration with the local search in IPOPT. This includes the effective selection of new candidates for the GP model, constraint handling within the BO framework, the optimization of hyperparameters, and a comparison between the performance of the GP model and a simple linear model. In the second part, the performance of BO-IPOPT compared to state-of-the-art methods is evaluated based on accuracy, computational costs, scalability (i.e., the ability to maintain computational efficiency and find near-optimal solutions as the problem dimensionality increases), and robustness (i.e., consistent performance across repeated runs), using the benchmark problems described earlier.

In this study, MS-IPOPT (see Algorithm 4 in Appendix A.3.2) is chosen for comparison due to the wide use of MS algorithms, as highlighted in Section 4.1. The goal is to determine whether the BO framework can offer better initial starting points for IPOPT in our hybrid approach compared to the random-based starting points used in MS-IPOPT, increasing the probability of achieving lower objective function values. Additionally, the competing hybrid method, BOwLS (see Algorithm 5 in Appendix A.3.3), is included for comparison as it shares similarities with BO-IPOPT but presents some technical distinctions, which are further explained in Appendix A.3.3. Finally, SCBO, SEGOKPLS, and SEGOKPLS+K are incorporated into the study as they represent effective BO methods for tackling constrained optimization problems in higher-dimensional spaces.

All optimization algorithms are implemented in Python 3.8 (Van Rossum and Drake, 2009), incorporating GAMS (GAMS Development Corporation, 2025) or PYOMO (Hart et al., 2011) for solving the optimization problems deterministically with the IPOPT solver (using its default settings). The interface between Python and GAMS is established through the GAMS API (GAMS, 2025). Additionally, GPR is implemented using the SKLEARN library (Pedregosa et al., 2011). For generating random points – in both initialization and at each outer iteration to evaluate the acquisition function – the LATINHYPERCUBE package (using the default settings) from SCIPY is considered.

4.4.1. Evaluating EI: Discrete Set of Candidates vs. Continuous Space

The selection of new candidate points for the GP model at each outer iteration is a crucial aspect of the BO component in our hybrid approach. Specifically, the next sampling point can be chosen in two ways: by evaluating the EI acquisition function over a predefined discrete set of candidate points or by optimizing EI over the entire continuous search space using numerical optimization techniques, which is a common practice in the literature. To assess the impact of these two strategies, we compare their effects on both the accuracy and computational efficiency of BO-IPOPT. While recent studies have explored alternative formulations such as logarithmic expected improvement LogEI (Ament et al., 2023), which can enhance EI's performance in high-dimensional and numerically challenging settings, as well as compositional acquisition functions (Grosnit et al., 2021), which can refine optimization strategies, our focus remains on standard EI due to its well-established reliability. It is important to note that all numerical experiments in this section are conducted solely on the simple test problem defined in (4.11)-(4.19). The reason for this is that applying the continuous-space approach to the larger benchmark problems would result in extremely high computational costs, making a comprehensive analysis impractical.

For the experimental setup, the following configuration is adopted: Constraints within the BO framework are managed using (4.2), and the key hyperparameters are set as follows: $N_0 = 10$, $N_c = 500$, $N_{bc} = 1$, and $\xi = 0.01$[1]. Additional parameters include $\alpha = 0.1$, $l = 100$, $\boldsymbol{\tau} = \mathbf{100}$, and $\boldsymbol{\delta} = \mathbf{0}$, with the latter choice reflecting the absence of equality constraints in our test cases. The value $\boldsymbol{\tau} = \mathbf{100}$ was chosen, given that the optimizer exhibited good performance with this setting in Kyriakidis et al. (2024), while the choice $N_{bc} = 1$ follows the classical sequential BO scheme (see line 8 in Algorithm 1). The values selected for α and l are intentionally larger than the default values $\alpha = 10^{-10}$ and $l = 1$[2], ensuring that the GP model in BO captures broader global trends in the search space rather than focusing on small-scale variations. This approach allows BO to efficiently guide the search toward promising regions, while IPOPT subsequently refines the solution to enforce feasibility concerning all constraints at each outer iteration K. Although the benchmark problems analyzed in this study are noise-free, including α plays a crucial role in maintaining numerical stability. Specifically, α ensures that the covariance matrix remains well-conditioned, preventing singularities during matrix inversion and mitigating potential numerical

[1]This value corresponds to the default setting in the SCIKIT-OPTIMIZE library (The scikit-optimize contributors, 2020).
[2]The SKLEARN library (Pedregosa et al., 2011) uses these as default values.

instabilities. Moreover, this parameter prevents overfitting and underfitting by ensuring that the GP model remains sufficiently flexible. In high-dimensional settings or when the data availability is limited, α further contributes to robustness by preventing the covariance matrix from becoming ill-conditioned. Unlike conventional BO strategies, which typically employ covariance kernels with adaptive length scales l optimized during training, our approach utilizes isotropic covariance kernels with a fixed length scale. This design choice prioritizes global exploration over local refinement, streamlining the BO process by eliminating the computational overhead associated with length-scale optimization (see Appendix A.3.5). As a result, BO efficiently identifies promising regions, while IPOPT handles the final optimization steps. Additionally, the values chosen for N_0 and N_c balance computational efficiency and accuracy, preventing excessive CPU time consumption due to repeated inversion of the correlation matrix and evaluation/optimization of the acquisition function at each outer iteration. This careful parameter selection ensures that the hybrid optimization framework remains scalable and computationally efficient.

To evaluate the EI function over the GPR using a continuous optimization approach, we solve the problem $\boldsymbol{x}_{z+1} = \arg\max_{\boldsymbol{x}\in D_j} \mathrm{EI}(\bar{\boldsymbol{x}}_i|\hat{u}_z)$ with two widely used solvers from SciPy: the multi-start L-BFGS-B (MS-L-BFGS-B) and the DE algorithm. The L-BFGS-B method is initialized from 20 different starting points with default settings to mitigate the risk of converging to local optima, while the DE algorithm is configured with a population size of 100, a maximum number of 200 iterations, a relative convergence tolerance of 0.01, a recombination parameter of 0.7, and a mutation factor ranging from 0.5 to 1. The last three parameters correspond to the default settings in the DE implementation, whereas the population size, number of iterations, and number of starting points were selected to achieve a good trade-off between solution quality and computational cost. The analysis is structured in two parts: First, we investigate how different EI optimization strategies influence BO itself, and then we assess their impact on the overall BO-IPOPT framework. Given the stochastic nature of both BO and BO-IPOPT, all numerical experiments are performed 100 times, with each run consisting of 300 outer iterations ($K = 300$). This repetition ensures a robust statistical evaluation of the methods.

The optimization results for the simple test case using the original BO approach are displayed in Fig. 4.1. The left-hand plot shows the average minimum objective function value as a function of the number of outer iterations K, while the right-hand plot presents the corresponding CPU time. As expected, evaluating the EI over the entire continuous search space using DE (optimization with DE in Fig. 4.1) leads to slightly faster convergence for $K \geq 29$ compared to evaluating EI over a discrete set of candidates (discrete candidates in Fig. 4.1). However, for the MS-L-BFGS-B solver (optimization with MS-L-BFGS-B in Fig. 4.1), the convergence behavior is similar to that of the discrete candidate evaluation approach. This is because MS-L-BFGS-B often gets trapped in local minima, which negatively impacts the overall BO performance. While DE demonstrates faster convergence in this example, this trend can vary depending on the problem characteristics and dimensionality, which suggests that a broader generalization should be made cautiously. Although DE increases the likelihood of accurately finding the global optimum of the acquisition function compared to the discrete candidate method, it comes at

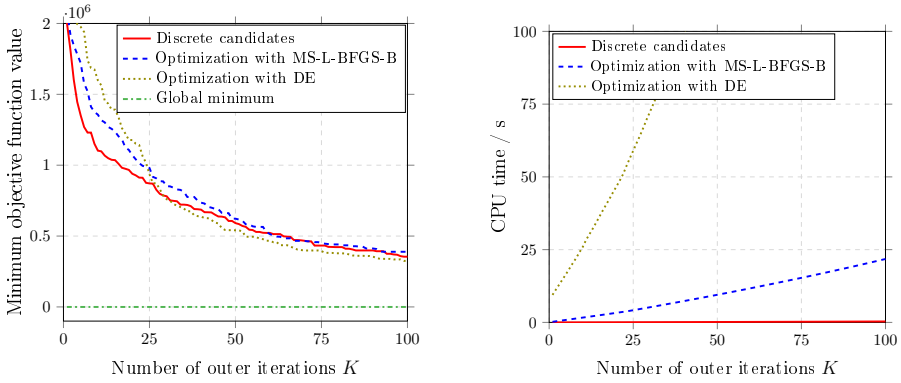

Figure 4.1: Optimization results for the simple test problem (4.11)-(4.19) using the original BO: comparison of the averaged minimum objective function value (**left**) and averaged CPU time (**right**) over 100 trials by evaluating the EI over a discrete set of candidates and through a continuous optimization procedure (via MS-L-BFGS-B and DE) as a function of the number of outer iterations K. The global minimum $f(\boldsymbol{x}^\star) \approx -216.65649$ determined by the BARON solver is also visualized on the left. Note that the values on the y-axis of the left figure are substantially larger than the global minimum because constraint violations cause the BO method to find infeasible solutions, preventing convergence to the feasible global optimum. This is why only results up to $K = 100$ are shown, as the high objective function values remain up to $K = 300$ where tested.

the cost of significantly higher computational expense due to the large number of function evaluations required. Similarly, MS-L-BFGS-B also demands considerable computational resources. Additionally, the BO solutions shown in Fig. 4.1 (left) remain far from the true global minimum, as proved by their relatively high minimum objective function values. This is primarily due to the presence of penalty terms in the objective function (4.2), which are not fully satisfied in these results – a common challenge faced by BO methods dealing with constraints. The extremely slow convergence rate observed here can be effectively mitigated by employing the hybrid BO-IPOPT approach, which will be addressed next.

In the BO-IPOPT framework, two approaches are examined to evaluate the EI: one over a discrete candidate set and another over the entire continuous space, specifically using MS-L-BFGS-B (see Fig. 4.2). The DE method is excluded due to its impractical computational demands, as already shown on the right of Fig. 4.1. Fig. 4.2 highlights the advantages of integrating BO with a local solver, demonstrating that BO-IPOPT achieves faster convergence to the global minimum while maintaining constraint satisfaction without requiring any adjustments to hyperparameters compared to the previous numerical experiments. This marks a significant improvement over the standard BO results shown in Fig. 4.1, which also explains why the range of the objective function values in Fig. 4.1 is larger than in Fig. 4.2. However, evaluating EI over a continuous space in BO-IPOPT significantly increases the computational costs, with a noticeable boost in convergence speed (fewer iterations K needed) compared to selecting EI over a discrete set of candidates. To offer a clearer comparison, the solver performance is visualized on the right side of Fig. 4.2, showing the minimum objective function value over the CPU time. The results show similar efficiency between the two approaches. A key observation is that the CPU time scales almost linearly when evaluating EI over a discrete set of points, whereas the continuous-space optimization leads to a superlinear increase in computational effort as the number of outer iterations grows. Increasing the

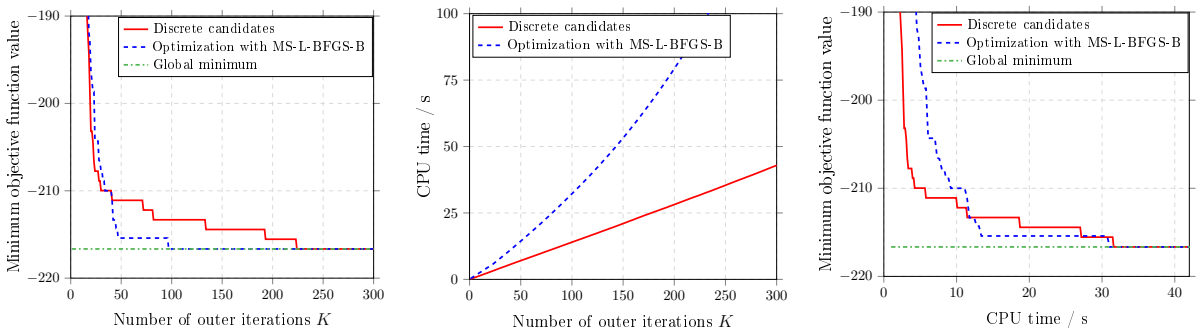

Figure 4.2: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials by evaluating the EI over a discrete set of candidates and through a continuous optimization procedure (via MS-L-BFGS-B) as a function of the number of outer iterations K. The global minimum $f(\boldsymbol{x}^\star) \approx -216.65649$ determined by the BARON solver is also visualized.

problem dimension (and thus also the EI dimension) will have a greater negative effect on the performance of BO-IPOPT when evaluating the EI through a continuous optimization procedure than over a discrete set of candidates. It is also important to emphasize that in this study, we focus on optimization scenarios where the computational burden of optimizing the acquisition function over a continuous space is nonnegligible relative to the cost of evaluating the objective function and constraints.

All in all, the evaluation of EI over a discrete set of points within the BO-IPOPT framework indicates a more balanced compromise between convergence speed toward the global optimum and computational efficiency, particularly for medium- and large-scale problems. Consequently, in all subsequent experiments, we adopt this strategy to identify the most suitable candidates at each outer iteration of the hybrid approach.

4.4.2. Constraint Handling in BO

Another important aspect of BO-IPOPT that requires further investigation is how equality and inequality constraints (included in the respective optimization problem) are effectively handled within the BO framework. In this work, two commonly used approaches (see Section 4.2.1) are examined: The first involves incorporating the constraints as penalty terms in the objective function (4.2), while the second employs the slack AL method in (4.3), which iteratively solves a sequence of simpler unconstrained subproblems (see Appendix A.3.1). To prevent a substantial increase in CPU time (see Appendix A.3.9), a single GP model is built rather than independent surrogate models. While single-model approaches are often regarded as less precise, our hybrid framework leverages BO for broad exploration and IPOPT for fine-tuned local optimization. This synergy reduces the reliance on highly accurate predictions, as BO efficiently navigates the global search space while IPOPT ensures precise convergence. As discussed in Section 4.2.3, both constraint-handling techniques directly influence the initial GPR, which is based on the initialization points, and subsequently impact the selection of new points at each iteration K using the EI. As expected, both techniques have a similar impact on the computational costs.

In the following analysis, we evaluate both BO-IPOPT techniques in terms of accuracy across three test cases: the simple test problem (4.11)-(4.19), the constrained Ackley function (4.20)-(4.21), and the DED problem (4.22)-(4.25) with five units (input data provided in Appendix A.3.4). It is important to highlight that the objective function (4.22) in the DED problem is nonsmooth, meaning it is not differentiable at all points. As a result, conventional gradient-based NLP solvers like IPOPT can not be applied directly. To address this challenge and reformulate the problem into a differentiable form suitable for gradient-based optimization, the cost function must be reformulated. Following the approach proposed in Pan et al. (2018), we introduce auxiliary variables within (4.22) and incorporate inequality constraints. These inequalities are then converted into equality constraints through the use of additional slack variables. While this reformulation ensures differentiability, it also increases the problem size, resulting in a 5-unit DED problem with 480 variables.

In contrast, the BO framework in the hybrid approach can handle the nondifferentiable formulation directly, eliminating the need for any reformulation. As a result, the original problem with 120 variables is used in the BO part. However, employing two different formulations for BO and the local search in BO-IPOPT presents a practical challenge: The initial starting points generated by BO for the local solver lack the necessary slack variables, leading to incomplete information. To address this issue in greater depth, we provide a detailed discussion in Appendix A.3.10.

Due to the 480 variables and nonsmooth, nonconvex structure introduced by valve-point effects, deterministic global optimization solvers become impractical for the 5-unit DED problem, as the computational effort grows exponentially with problem size, even when run on computing clusters (Neumaier et al., 2005). This challenge has motivated the development and application of hybrid optimization algorithms (combining stochastic and/or local deterministic methods) for DED problems with valve-point effects, rather than relying on deterministic global solvers (He et al., 2011; Santra et al., 2020). Accordingly, to approximate the global minimum, we employ MS-IPOPT, BO-IPOPT, and BOwLS with 10,000 different starting points to comprehensively explore the parameter space. The best-obtained solution, $f(\boldsymbol{x}^\star) \approx 42{,}524.2785$, serves as our estimated global minimum.

To account for the stochastic nature of the optimizers, multiple experimental runs are conducted: 100 repetitions for the two test problems and 50 for the DED problem, with $K = 300$ outer iterations. The hyperparameters remain consistent with those specified in Section 4.4.1. For the DED problem, which was not considered in that section, we set $\boldsymbol{\delta} = \boldsymbol{\tau} = \mathbf{100}$ for BO-IPOPT with the penalty-based method (4.2). The number of trials for the DED problem is reduced due to its higher dimensionality, which significantly increases the computational effort; however, it remains sufficient to assess the optimizer's performance reliably.

The optimization results for the three problems are presented in Fig. 4.3. As shown, both methods exhibit similar convergence rates. However, the AL method offers a significant advantage: The hyperparameters are chosen based on predefined guidelines (see Appendix A.3.1), making this approach parameter-free. In contrast, the penalty-based method in (4.2) relies on user-defined penalty weights, which are often unknown

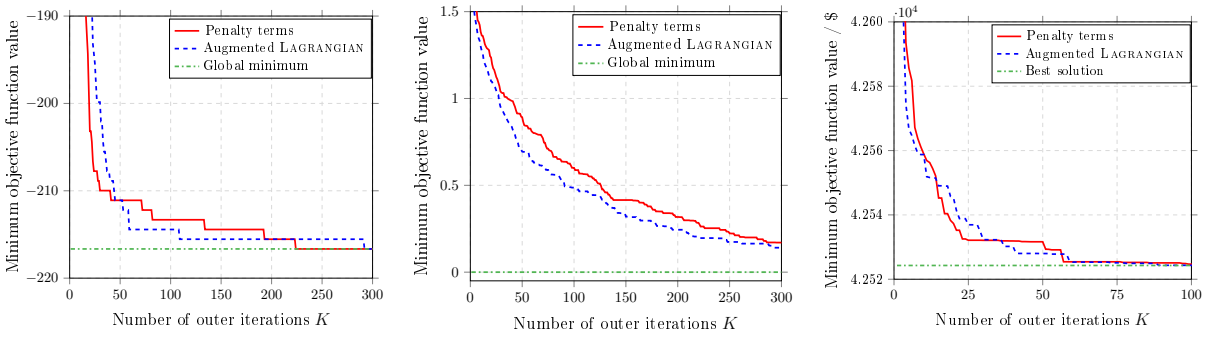

Figure 4.3: Optimization results for the simple test problem (4.11)-(4.19) (**left**), the constrained Ackley function (4.20)-(4.21) (**middle**), and the DED problem with 5 units (4.22)-(4.25) (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 trials for the two test cases and 50 trials for the DED problem considering the two constraint handling techniques, i.e., (4.2) vs. (4.3), as a function of the number of outer iterations K. We display the results for the DED problem only up to $K = 100$ because the best solution is quickly reached on average. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

and problem-dependent. Specifically, selecting appropriate values is not trivial, as the hyperparameters are vectors with different entries depending on the types of constraints and the degree to which each constraint is enforced during optimization. In this thesis, different entries have not been systematically investigated, as doing so would be required to find the optimal values, which would lead to high computational costs. While constraint handling primarily influences the initial GPR, selecting suitable values for the initial surrogate model can enhance the optimization process, as these choices also impact the points selected at each iteration K.

It is essential to emphasize that the AL method has been integrated with BO and IPOPT in such a way that it avoids the challenges associated with high-dimensional problems and large numbers of constraints, as discussed in Section 4.2.1. In BO-IPOPT, the GP model does not need to provide an extremely detailed approximation of the objective function and constraints across the entire search space. Instead, it acts as a global guide, identifying regions with promising solutions by balancing exploration and exploitation. Once a promising region is identified, IPOPT takes over, using gradient-based optimization to efficiently navigate the local landscape and optimize the objective function while respecting the defined constraints. In contrast, Picheny et al. (2016) utilize multiple surrogate models – typically one for the objective function and separate models for each constraint. However, this method lacks a local solver like IPOPT, which means the global search depends entirely on the surrogate models to approximate and satisfy constraints. This can lead to inefficiencies, especially in high-dimensional spaces or when dealing with a large number of constraints. Overall, we consider the AL approach for handling equality and inequality constraints in BO-IPOPT, and we consider this parameter-free technique for all subsequent experiments.

4.4.3. Parameter Study and Setting

As previously discussed, our proposed BO-IPOPT method includes several hyperparameters that influence both the global guidance provided by the BO component and the computational efficiency of the

hybrid method. Therefore, we aim to identify suitable values for these parameters to provide a balance between fast convergence and low computational cost, ensuring effective guidance from the BO and mitigating scalability challenges in problems of higher dimensions. Although these hyperparameters are naturally problem-dependent, we conduct an experimental analysis to assess their impact on the hybrid method. This helps us establish reasonable parameter values that remain fixed throughout the optimization process, rather than being adapted dynamically as it is typically done in traditional BO methodologies.

The hyperparameters considered in the hybrid method include the length scale l, the regularization term α, the exploration term ξ, the number of initialization points N_0, the number of candidates N_c considered in EI, and the number of best candidates N_bc at each outer iteration. It is important to note that the variation of l, α, and ξ does not directly affect the computational cost of BO-IPOPT (without simultaneously changing $N_0, N_\mathrm{c}, N_\mathrm{bc}$). As a result, we exclude the comparison of CPU time for these three parameters.

The parameter study is conducted based on three benchmark problems: the simple test problem, the constrained Ackley function, and the 5-unit DED problem. Although the number of benchmark problems is limited and does not allow for a fully optimal parameter selection, it is still essential to reduce the number of free parameters in BO-IPOPT. However, a more detailed quantitative evaluation will be the focus of future research. For the numerical experiments in this section, we use the same hyperparameter settings as in Section 4.4.1, treating them as the baseline. Each parameter is analyzed individually while keeping all others fixed at their baseline values. To account for the method's randomness, each experiment is repeated 100 times for the test problems and 50 times for the DED problem, as in Section 4.4.2.

It is worth noticing that variations in parameter values have little impact on the BO-IPOPT's performance in the simple test problem (see Figs. A.13-A.16 in Appendix A.3.8) due to its low complexity. Therefore, we primarily focus on the other two optimization problems, where hyperparameter selection significantly influences the optimizer's performance (see Figs. 4.4-4.9). In high-dimensional problems, the choice of hyperparameters becomes even more critical, as the complexity of the search space increases and surrogate models struggle to accurately capture intricate relationships across many dimensions. Additionally, some hyperparameters have a direct effect on the computational cost of the optimization process, which becomes increasingly significant as the dimension of the problem grows.

First, we analyze the influence of the length scale l on BO-IPOPT's performance, with the results illustrated in Fig. 4.4. The choice of l significantly influences the convergence of the optimizer. Smaller values of l result in slower convergence, requiring more iterations (K) to reach an optimal solution. This occurs because a smaller length scale allows the model to capture rapid variations in the function, making it more sensitive to local fluctuations. As a consequence, the optimizer tends to focus on fine details, increasing the risk of getting trapped in local minima and slowing down the convergence. In contrast, larger values for l lead to smoother function approximations, promoting broader exploration and faster convergence. However, too large values may over-smooth the function, potentially reducing the optimizer's ability to accurately capture the problem's structure. Our results suggest that $l = 100$ provides the best

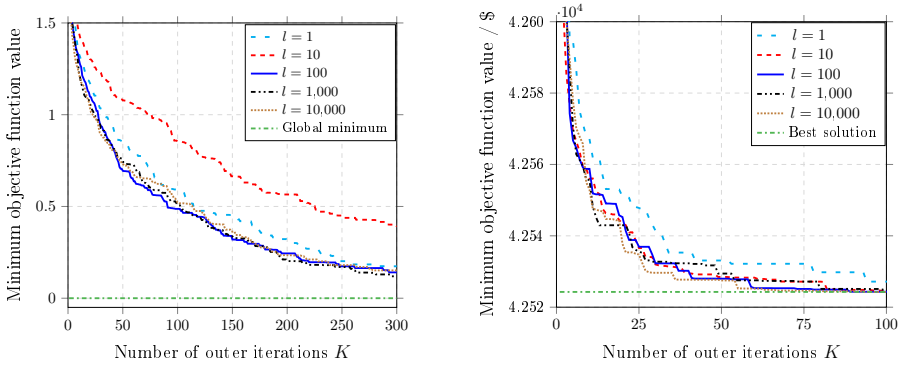

Figure 4.4: Optimization results for the constrained ACKLEY function (4.20)-(4.21) (**left**) and the DED problem (4.22)-(4.25) with 5 units (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 and 50 trials for different length scales l as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

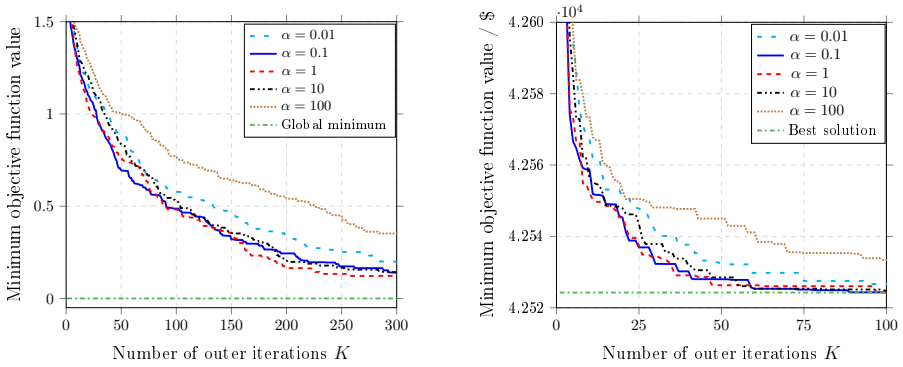

Figure 4.5: Optimization results for the constrained ACKLEY function (4.20)-(4.21) (**left**) and the DED problem (4.22)-(4.25) with 5 units (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 and 50 trials for different regularization terms α as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

balance for BO-IPOPT, since greater values do not yield noticeable improvements in convergence. This choice enables the GP model in the BO component to explore the entire search space more effectively, while IPOPT refines the search within promising regions identified by BO.

Fig. 4.5 presents the performance of BO-IPOPT for different values of the regularization term α. The results indicate that large α values cause the GP model to underfit the data, oversimplifying the surrogate model and leading to suboptimal performance. As a result, the hybrid method shows a slower convergence rate. On the other hand, very small α values increase the risk of overfitting, reducing the model's ability to generalize to unseen data and also slowing down the convergence. Therefore, an intermediate value is preferable, and based on our findings, we recommend setting $\alpha = 0.1$ as a balanced choice for BO-IPOPT.

The third parameter analyzed is the exploration term ξ, with the corresponding results displayed in Fig. 4.6. Variations in ξ have little impact on the convergence rate of BO-IPOPT for the considered problems. This is primarily because the local solver weakens the effect of ξ in the BO component, reducing the need for precise tuning of this parameter. However, the absence of exploration (i.e., $\xi = 0$) results in

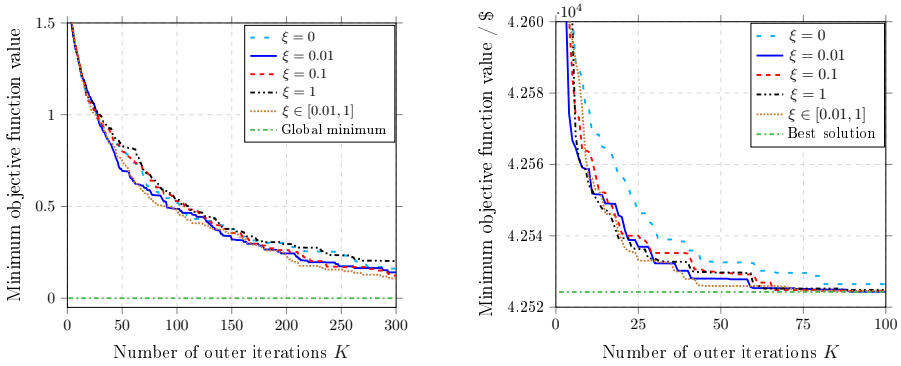

Figure 4.6: Optimization results for the constrained ACKLEY function (4.20)-(4.21) (**left**) and the DED problem (4.22)-(4.25) with 5 units (**right**) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 and 50 trials for different exploration terms ξ as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

slightly slower convergence in the DED problem. While different constant values of ξ lead to a good and similar performance for BO-IPOPT, we recommend using a monotonically decreasing sequence, starting at $\xi = 1$ and gradually reducing to $\xi = 0.01$. This approach promotes exploration in the initial iterations and shifts the focus toward exploitation as the optimization progresses. Moreover, the monotonically decreasing strategy provides performance comparable to constant choices of ξ, while offering the practical advantage of dynamically adapting the balance between exploration and exploitation without requiring problem-specific tuning of ξ.

Next, we examine the remaining parameters N_0, N_c, and N_{bc}, which impact both the accuracy and computational cost of BO-IPOPT, as illustrated in Figs. 4.7-4.9. Compared to the previous figures, these present also the solver performance (CPU time vs. minimum objective function value) for a more comprehensive comparison.

As shown in Fig. 4.7 (middle), the number of initialization points N_0 has a significant influence on the computational effort of BO-IPOPT, particularly for very large values of $N_0 \geq 500$. However, as indicated on the left of Fig. 4.7, the convergence rate is not significantly affected by changes in N_0. Larger values of N_0 lead to a substantial increase in CPU time for BO-IPOPT because the initial GPR starts with a larger set of sample points. Consequently, inverting the correlation matrix at each outer iteration becomes computationally intensive as more samples are added. A value of $N_0 = 10$ offers a good balance (see the right side of Fig. 4.7) between faster convergence and low computational costs, as it allows the initial GPR to explore the entire space while keeping the optimizer's CPU time low compared to higher values of N_0.

The results of BO-IPOPT with varying numbers of candidates N_c are presented in Fig. 4.8. Increasing N_c leads to a higher number of acquisition function evaluations at each outer iteration, which slightly increases the computational costs of the hybrid method (as shown in the middle figure). A larger N_c also increases the probability of identifying a suitable candidate, which can improve the surrogate model and, consequently, the convergence rate. However, too large values (e.g., $N_c = 1{,}000$) can increase the CPU time without offering significant improvements in the solution accuracy compared to smaller values (e.g., $N_c = 500$).

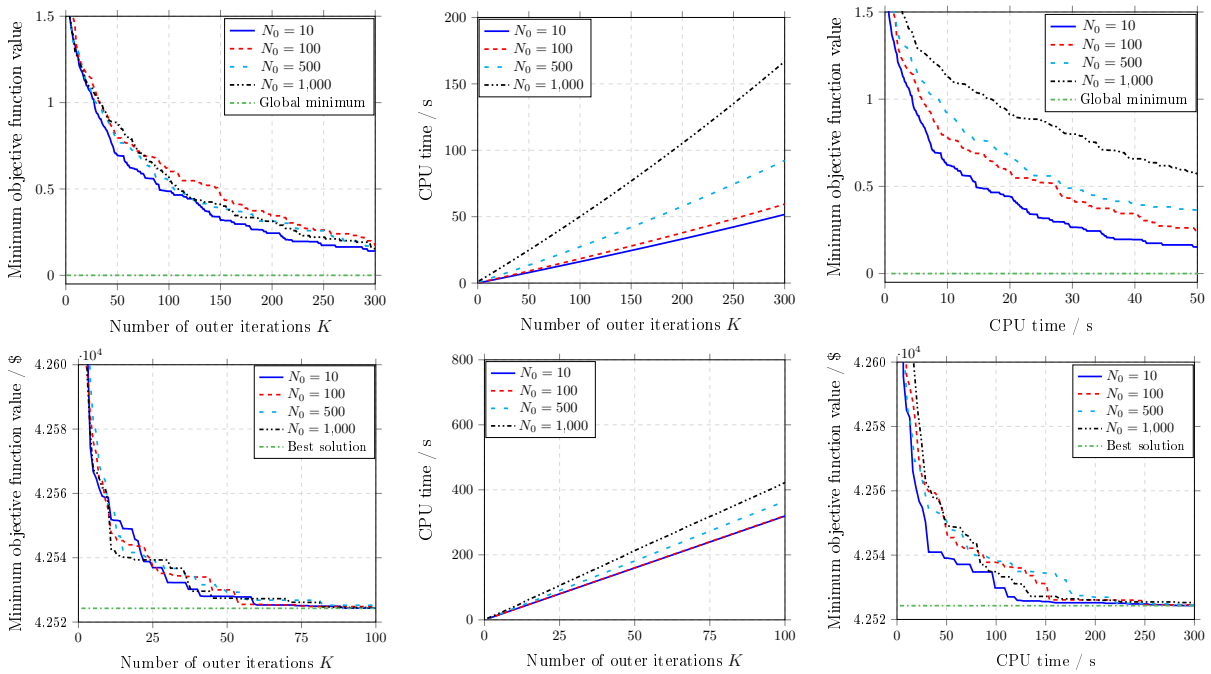

Figure 4.7: Optimization results for the the constrained Ackley function (4.20)-(4.21) (**top**) and the DED problem (4.22)-(4.25) with 5 units (**bottom**) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 and 50 trials for different numbers of initialization points N_0 as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

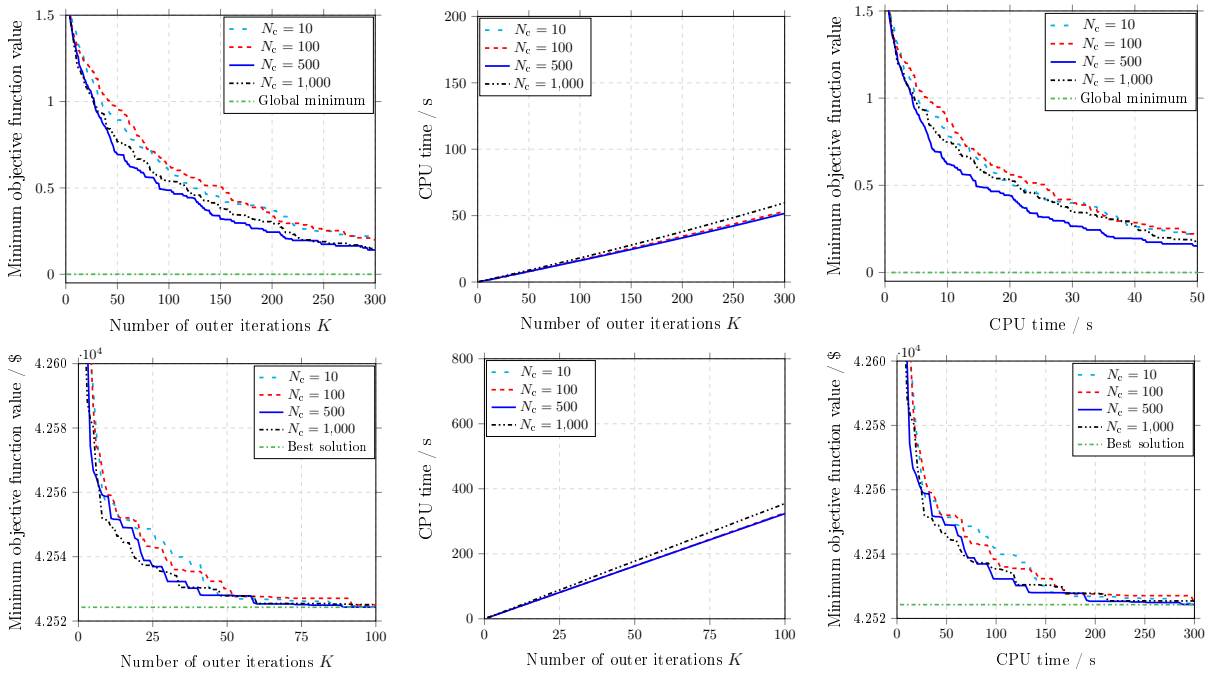

Figure 4.8: Optimization results for the the constrained Ackley function (4.20)-(4.21) (**top**) and the DED problem (4.22)-(4.25) with 5 units (**bottom**) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 and 50 trials for different numbers of candidates N_c as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

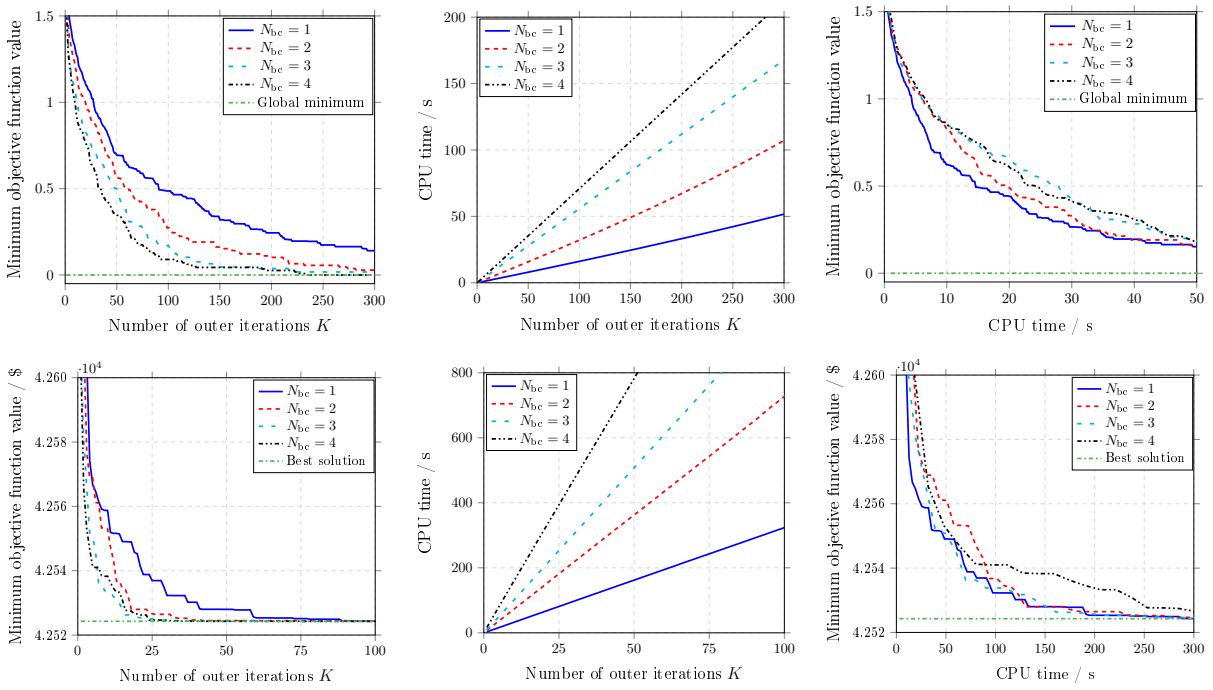

Figure 4.9: Optimization results for the constrained ACKLEY function (4.20)-(4.21) (**top**) and the DED problem (4.22)-(4.25) with 5 units (**bottom**) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 and 50 trials for different numbers of best candidates N_{bc} as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

Therefore, setting $N_{\text{c}} = 500$ in BO-IPOPT is considered a reasonable choice (see the right side of Fig. 4.8), as it provides a balance between convergence and CPU time.

The final parameter, N_{bc}, investigated in BO-IPOPT, has the most significant effect on both the convergence rate and CPU time of the optimizer, as shown in Fig. 4.9. Each additional best candidate (used at each outer iteration) increases the probability of finding the global minimum more quickly. However, this also results in higher computational costs due to the additional IPOPT searches. Overall, the solver's performance for both problems remains unaffected (see the right side of Fig. 4.9). Contrary to expectations, a larger N_{bc} does not provide a clear advantage, since the ratio of convergence rate and CPU time is roughly linear[3]. However, we recommend using $N_{\text{bc}} > 1$ since local searches are highly suitable for parallel computing due to their independent nature. With additional best candidates, BO-IPOPT generally has a higher probability of finding the global optimum in less CPU time (due to parallel searches) as more candidates are added to the GPR at each outer iteration. Nevertheless, the number of best candidates selected at each outer iteration should be chosen carefully, as the GPR scales cubically with the number of samples involved due to the inversion of the covariance matrix. In all investigated cases, however, this is not particularly relevant, as explained in Sections 4.4.6 and 4.4.7. For the studies that follow in this

[3]Increasing N_{bc} while keeping K fixed leads to a similar performance as increasing K while fixing N_{bc}, e.g., for the ACKLEY problem, $N_{\text{bc}} = 3$, $K = 100$ and $N_{\text{bc}} = 1$, $K = 300$ (which corresponds to a total number of 300 local searches) achieve nearly the same minimum objective function value in almost the same CPU time.

chapter, we have chosen $N_{\text{bc}} = 2$, as it increases the likelihood of finding the global optimum compared to $N_{\text{bc}} = 1$. However, since BO-IPOPT has not been parallelized in this chapter to ensure a fair comparison with the BO approaches in Section 4.4.6, values higher than two lead to excessively high CPU times, particularly for larger problems such as those presented in Section 4.3.3 and 4.3.4.

4.4.4. Surrogate Modeling in BO-IPOPT: GP vs. Linear Model

Although the GP model is known for its flexibility and uncertainty estimates, in BO-IPOPT its role is simplified to providing global guidance rather than capturing the full nonlinear system behavior. To better understand this role, we compare the GP model with a simple linear surrogate model to see if its added sophistication results in improved performance or if, within the hybrid method, the GP model essentially behaves like the simpler linear model for the benchmarks studied.

A critical challenge that arises when employing a linear model in this context is the lack of uncertainty quantification, which is a key component in the standard BO acquisition strategy (e.g., EI). Without predictive variance, the question becomes how to effectively select the most promising candidates at each outer iteration. Based on Regis and Shoemaker (2007), a weighted criterion that combines the quality of the objective function values and the distance from the already evaluated points in D is employed to guide exploration and exploitation. This selection criterion can be expressed as:

$$\alpha(\bar{\boldsymbol{x}}_i) = w_u v_u(\bar{\boldsymbol{x}}_i) + w_d v_d(\bar{\boldsymbol{x}}_i) \text{ for } i = 1, \ldots, N_{\text{c}}, \tag{4.36}$$

with $w_u + w_d = 1$. The parameter w_u describes the weight for the quality criterion v_u, while w_d denotes the weight for the distance criterion v_d. The two included criteria are defined as follows:

$$v_u(\bar{\boldsymbol{x}}_i) = \frac{\hat{u}(\bar{\boldsymbol{x}}_i) - \hat{u}_{\text{min}}}{\hat{u}_{\text{max}} - \hat{u}_{\text{min}}}, \quad v_d(\bar{\boldsymbol{x}}_i) = \frac{d_{\text{max}} - d(\bar{\boldsymbol{x}}_i)}{d_{\text{max}} - d_{\text{min}}} \quad \text{for } i = 1, \ldots, N_{\text{c}}, \tag{4.37}$$

where the minimum and maximum values are determined by the following equations:

$$\hat{u}_{\text{min}} = \min_i \hat{u}(\bar{\boldsymbol{x}}_i),\ \hat{u}_{\text{max}} = \max_i \hat{u}(\bar{\boldsymbol{x}}_i),\ d_{\text{min}} = \min_i d(\bar{\boldsymbol{x}}_i),\ d_{\text{max}} = \max_i d(\bar{\boldsymbol{x}}_i) \text{ for } i = 1, \ldots, N_{\text{c}}. \tag{4.38}$$

The minimum distance of each sample to already evaluated points in D is calculated by:

$$d(\bar{\boldsymbol{x}}_i) = \min_j \|\bar{x}_i - x_j\| \text{ for } i = 1, \ldots, N_{\text{c}},\ j = 1, \ldots, N_0 + KN_{\text{bc}}. \tag{4.39}$$

Through the normalization procedure in (4.37), both the quality and distance criterion take values between zero and one. The goal of the weighted selection criterion is to prioritize high-quality candidates while avoiding those that are located too close to previously evaluated points. This aims to improve efficiency, as points located near previously evaluated samples generally offer limited informational gain.

The performance of BO-IPOPT considering a GP and a linear model is evaluated based on the four benchmark problems introduced in Section 4.3. In this section, the IPOPT solver is used either through Pyomo or through Gams, depending on the specific problem being addressed. More specifically, IPOPT is used via Pyomo for the two benchmark test cases, whereas Gams is employed for the DED and RSG problems. The choice of Gams for the DED and RSG problems is motivated by the need to test additional local solvers available within the Gams environment in Section 4.4.7.

Before discussing the performance of the methods based on the benchmarks, we are providing at this point more information about the setup and the optimal solution of the conceptual use cases. For both the DED and RSG problem, we consider a typical one-day operation with hourly time step size, where the corresponding input data can be found in Appendix A.3.4. The resulting DED optimization problem (4.22)-(4.25) for 10 and 30 units contains 240 and 720 decision variables with the best known solutions being 1,015,438.967 USD and 3,051,105.813 USD, respectively (Santra et al., 2020). As mentioned in Section 4.4.2, the cost function of the DED problem (4.22) should be reformulated so that gradient-based NLP solvers such as IPOPT can be applied. By introducing additional variables, the 5-, 10-, and 30-unit DED problem now contain 480, 960, and 2,880 variables, respectively, while the BO part in the hybrid approach considers the original problems with 120, 240, and 720 decision variables, respectively, to avoid unreasonable increase in CPU time.

For the RSG system described in (4.26)-(4.35), the 24-hour optimization problem involves 408 decision variables, placing it beyond the practical limits of commercial global optimization solvers such as Baron, due to the exponential growth of computational effort with problem size (Neumaier et al., 2005). To address this, MS-IPOPT, BO-IPOPT, and BOwLS have been applied with 10,000 randomly generated starting points, with the best solution being at 1,046.53 €. This result is therefore used as the reference solution for the best possible minimum.

In this section, BO-IPOPT with the GP model includes following parameters (see Section 4.4.3): $l = 100$, $\alpha = 0.1$, a monotone sequence $\xi \in [0.01, 1]$, an initial sampling size of $N_0 = 10$, a constraint handling parameter of $N_\mathrm{c} = 500$, and a batch size of $N_\mathrm{bc} = 2$. For the linear model, BO-IPOPT employs the same parameter settings as used with the GP model – defined according to the hyperparameter study in Section A.3.6 – additionally using $w_u = 0.9$ and $w_d = 0.1$. To ensure robust comparisons and account for the stochastic nature of BO-IPOPT, multiple independent runs were performed: 100 trials for the simple test problem and the constrained Ackley function, 50 trials for the RSG and the DED problem with 5 and 10 units, and 20 trials for the 30-unit DED problem. As already mentioned, the different numbers of runs are due to the increasing computational effort with higher-dimensional problems; however, they are sufficient to assess the optimizer's performance reliably.

The results of BO-IPOPT using the GP model are compared with those obtained using the linear surrogate model in Figs. 4.10 and 4.11. Both approaches show a similar convergence rate toward the global minimum or best-known solution across all benchmark problems. Moreover, the CPU time for both variants are nearly identical, and are therefore omitted here. This outcome is particularly noteworthy given

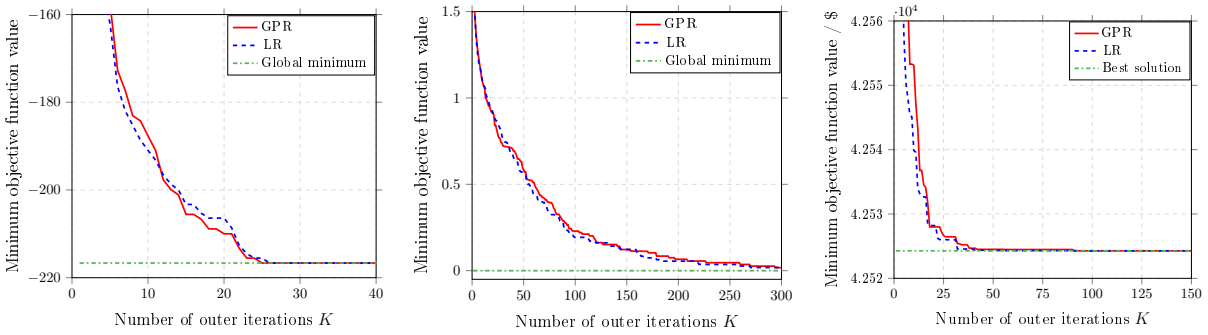

Figure 4.10: Optimization results for the simple test problem (4.11)-(4.19) (**left**), the constrained Ackley function (4.20)-(4.21) (**middle**), and the DED problem (4.22)-(4.25) with 5 units (**right**): comparison of the averaged minimum objective function value over 100 and 50 trials using BO-IPOPT with a linear and GP model as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the \$ sign for brevity.

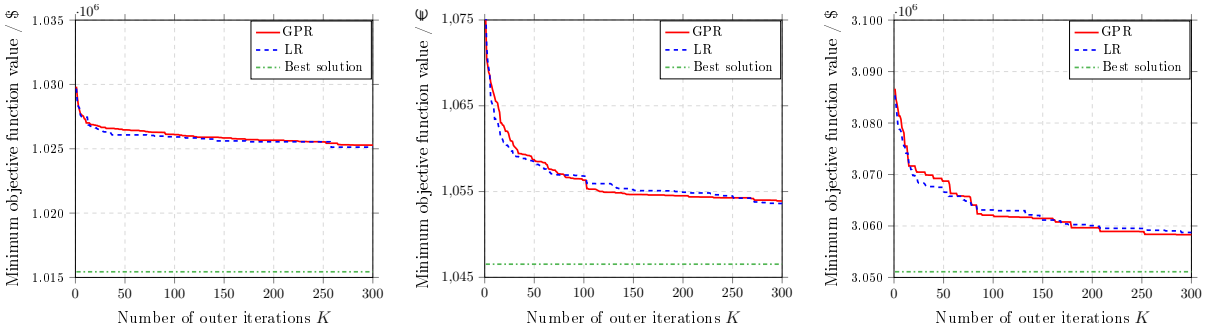

Figure 4.11: Optimization results for the DED problem (4.22)-(4.25) with 10 units (**left**), for the RSG problem (4.26)-(4.35) (**middle**), and the DED problem (4.22)-(4.25) with 30 units (**right**): comparison of the averaged minimum objective function value over 50 and 20 trials using BO-IPOPT with a linear and GP model as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the \$ sign for brevity.

that the GP model is typically favored in BO due to its flexibility and its ability to quantify uncertainty. However, in the context of BO-IPOPT, the surrogate model mainly provides global guidance, while IPOPT takes over for local, high-resolution optimization. As such, the GP model is not tasked with representing all the detailed nonlinear behavior of the system, but rather with approximating the global structure of the objective function along with the constraints in a smooth and informative way. Under these conditions, even a simple linear model – combined with a suitable candidate selection strategy – can direct the search effectively, leading to comparable optimization trajectories.

Nevertheless, in the following sections, the GP model continues to be considered in the hybrid method. The primary motivation is its ability to integrate naturally into the BO, particularly through acquisition functions like EI, which leverage predictive uncertainty. Additionally, while the linear model proved sufficient in the considered benchmarks, the GP model offers greater modeling flexibility, which may become beneficial in more complex applications beyond the scope of this study.

4.4.5. Trust Region Method in BO-IPOPT

Another aspect investigated in this work is the incorporation of a trust region method inspired by Eriksson et al. (2019) into the BO part of the proposed hybrid optimizer. Trust region methods play a crucial role in constrained optimization problems, particularly when dealing with high-dimensional spaces or noisy data. By dynamically adjusting the search space, they balance the trade-off between exploration and exploitation, ensuring that the optimization process remains efficient and focused while avoiding excessive sampling in irrelevant areas.

In our approach, the trust region is applied at each iteration K, where a new sample of points is generated (see line 7 in Algorithm 2), and the bounds of the sample region are adaptively adjusted. The trust region is initialized as a hypercube with a side length of L set to $L = L_{\text{init}}$ and a center $\boldsymbol{x}_{\text{center}}$ that is determined based on the initial candidate points. If there exist feasible points (satisfying all constraints), the center is set to the feasible point with the best objective value; otherwise, it is set to the point with the smallest constraint violation. This ensures that the initial trust region is centered on a promising area of the search space. On the one hand, a success occurs when a newly evaluated point improves upon the current best solution, leading to an update of the trust region center $\boldsymbol{x}_{\text{center}}$ to this new best point. On the other hand, a failure is recorded when no sampled point outperforms the current best solution, in which case the center remains unchanged. The bounds of the trust region are adjusted as follows:

$$\mathbf{tr}_{\text{lb}} = \boldsymbol{x}_{\text{center}} - \frac{L}{2}, \quad \mathbf{tr}_{\text{ub}} = \boldsymbol{x}_{\text{center}} + \frac{L}{2}. \tag{4.40}$$

The trust region undergoes resizing according to the following rules: If the success counter n_s reaches the threshold τ_s, the side length is expanded to $L = \min(2L, L_{\text{max}})$ and the counter is reset. If the failure counter n_f reaches τ_f, the side length is reduced to $L = L/2$ and the counter is similarly reset. When the side length falls below the minimum threshold L_{min}, the current trust region is terminated, and a new one is initialized according to the same criteria as the first trust region.

Following the settings proposed by Eriksson et al. (2019), the hyperparameters used in our implementation include $\tau_s = \max(3, n/10)$ and $\tau_f = n/N_{\text{bc}}$, where n is the problem dimension and N_{bc} is the number of best candidates selected at each outer iteration. The trust region's side length is constrained within $L_{\text{min}} = 2^{-7}$ (ensuring that the trust region never collapses completely before restarting), $L_{\text{max}} = 1.6$, with an initialization value of $L_{\text{init}} = 0.8$. These values showed good performance across benchmark problems investigated by Eriksson et al. (2019). The remaining hyperparameters in BO-IPOPT are employed based on Section 4.4.3 as follows: $l = 100$, $\alpha = 0.1$, monotone sequence $\xi \in [0.01, 1]$, $N_0 = 10$, $N_{\text{c}} = 500$, and $N_{\text{bc}} = 2$.

In this section, we compare the performance of the hybrid approach both with and without the incorporation of the trust region method. The comparison is carried out across the four benchmark problems introduced in Section 4.3. As expected, including the trust region method does not significantly influence the CPU time of the hybrid optimizer, i.e., the computational costs are almost identical to the

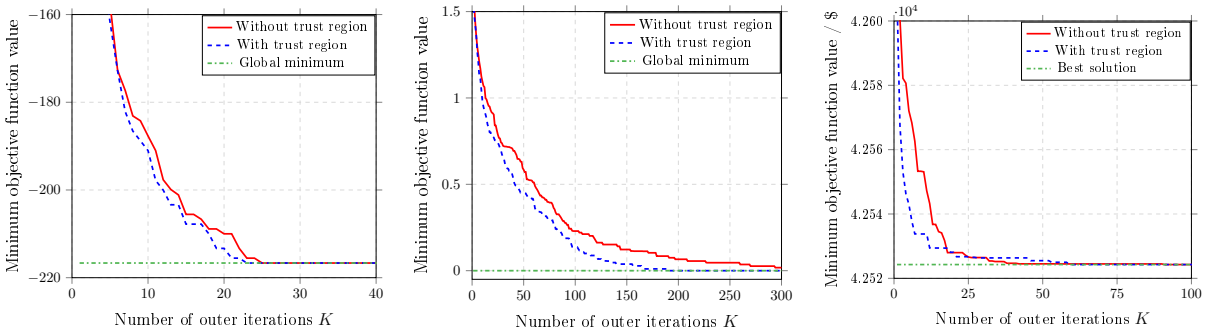

Figure 4.12: Optimization results for the simple test problem (4.11)-(4.19) (**left**), the constrained Ackley function (4.20)-(4.21) (**middle**), and the DED problem (4.22)-(4.25) with 5 units (**right**): comparison of the averaged minimum objective function value over 100 and 50 trials using BO-IPOPT without and with the trust region method as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the \$ sign for brevity.

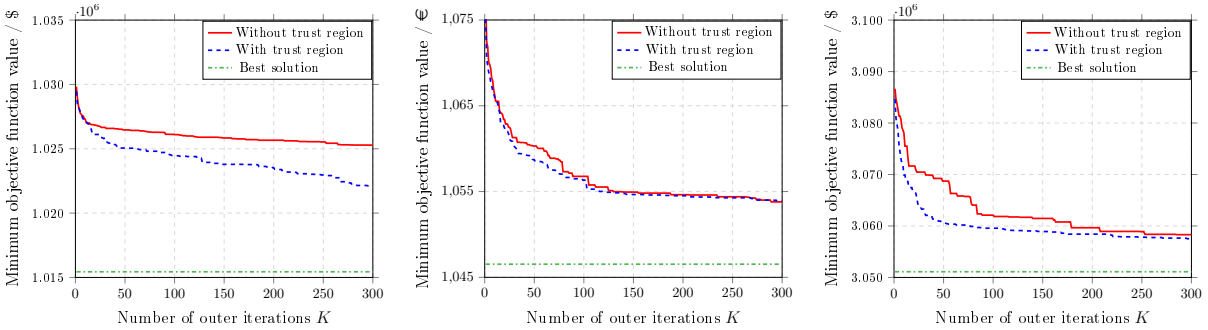

Figure 4.13: Optimization results for the DED problem (4.22)-(4.25) with 10 units (**left**), for the RSG problem (4.26)-(4.35) (**middle**), and the DED problem (4.22)-(4.25) with 30 units (**right**): comparison of the averaged minimum objective function value over 50 and 20 trials using BO-IPOPT without and with the trust region method as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the \$ sign for brevity.

case without the trust region. For this reason, the graphs showing CPU time over the number of outer iterations are not displayed in this section. To account for the stochastic nature of BO-IPOPT, multiple independent runs were carried out, as in Section 4.4.4: 100 trials for the simple test problem and the constrained Ackley function, 50 trials for the RSG problem and the DED problems with 5 and 10 units, and 20 trials for the DED problem involving 30 units.

Figs. 4.12 and 4.13 display the minimum objective function value as a function of the number of outer iterations. As seen in the figures, incorporating the trust region method in BO-IPOPT enhances the convergence speed of the hybrid method. The improvement provided by the trust region approach can be observed from the first iterations and continues throughout the optimization process. This effect is particularly noticeable in the constrained Ackley function, as well as in the DED problem with 10 and 30 units because of the challenging nature and larger search space of the problems. In the constrained Ackley function, the objective function value is improved by up to 100 % compared to the case without the trust region. Similarly, in the DED problems, including the trust region approach leads to daily cost

savings of up to 3,166 USD and 8,524 USD for the 10-unit and 30-unit case, respectively. This highlights the ability of the trust region method to guide the search efficiently, particularly in complex problems. The trust region method helps BO maintain a global perspective while refining the search within a structured and dynamically adjusted space. This allows IPOPT to operate in a well-guided yet flexible search region, reducing highly suboptimal areas. Given these benefits, this approach is adopted in the following sections.

4.4.6. Comparison of SCBO, SEGOKPLS, SEGOKPLS+K, and BO-IPOPT

This section compares our fine-tuned hybrid method with three state-of-the-art BO methods – SCBO, SEGOKPLS, and SEGOKPLS+K – which are designed for high-dimensional optimization and were introduced earlier in Section 4.1. The comparison focuses on the simple test problem and the constrained Ackley function, which provide a clear illustration of the advantages and limitations of each method.

For BO-IPOPT, the following parameter settings based on Section 4.4.3 are used: $l = 100$, $\alpha = 0.1$, monotone sequence $\xi \in [0.01, 1]$, $N_0 = 10$, $N_\mathrm{c} = 500$, and $N_\mathrm{bc} = 2$. Constraints in BO-IPOPT are handled using the AL framework introduced in (4.3), and the EI criterion is evaluated over a discrete set of candidate points. Additionally, the trust region settings in BO-IPOPT are based on the methodology described in Section 4.4.5. To ensure a fair comparison, the same values for N_0, N_c, and N_bc are applied across all optimization methods. SCBO is implemented using BoTorch (Balandat et al., 2020), employing a Matérn kernel and an extended Thompson sampling approach for constrained optimization, which selects new points for the GPR at each outer iteration K. Moreover, SCBO utilizes a trust region method such as the one described by Eriksson et al. (2019), with trust region parameters set as proposed by Eriksson et al. (2019) and applied to BO-IPOPT to ensure consistency across methods. The KPLS and KPLSK components in SEGOKPLS and SEGOKPLS+K are implemented using the Smt library (Saves et al., 2024). The number of principal components is set to two for the simple test problem and nine for the constrained Ackley function, since a higher value is not feasible due to the ten initial points in the surrogate models (for the objective function and constraints), which is consistent with the initialization size used in BO-IPOPT and SCBO. The constrained Ackley function requires more principal components due to its higher dimension. The acquisition function, which is based on the EI criterion weighted by the probability of constraint satisfaction, is optimized using the MS-L-BFGS-B with 10 restarts. Parallel computing is not utilized for this maximization to maintain alignment with the components in BO-IPOPT that are not parallelized, such as both the BO part and the sequential IPOPT local searches, as well as with the SCBO method. This ensures fair and consistent resource usage across all methods. All other hyperparameters for SCBO, SEGOKPLS, and SEGOKPLS+K are kept at their default settings. To address the stochastic behavior of the optimizers, multiple independent runs were performed: 100 trials for the simple test problem and the constrained Ackley function.

As illustrated in Figs. 4.14 and 4.15, BO-IPOPT demonstrates a superior convergence rate toward the global minimum compared to alternative BO methods while simultaneously maintaining lower computational costs. This results in significantly improved performance, achieving lower minimum

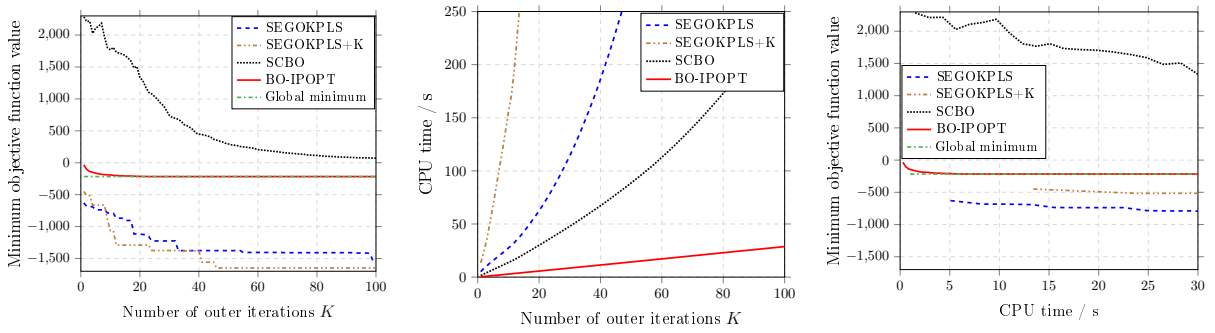

Figure 4.14: Optimization results for the simple test problem (4.11)-(4.19): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials using SCBO, SEGOKPLS, SEGOKPLS+K, and BO-IPOPT as a function of the number of outer iterations K.

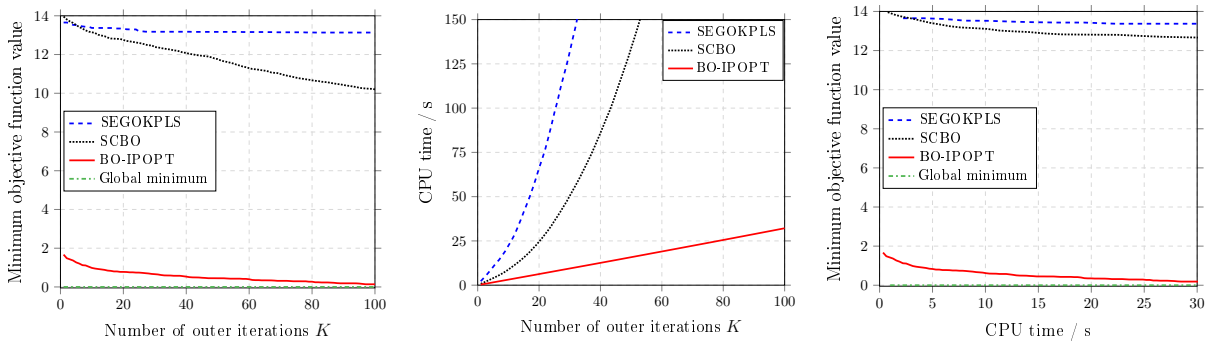

Figure 4.15: Optimization results for the constrained Ackley function (4.20)-(4.21): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials using SCBO, SEGOKPLS, and BO-IPOPT as a function of the number of outer iterations K.

objective function values within the same CPU time while satisfying the constraints. More precisely, BO-IPOPT shows an average improvement of approximately 120 % compared to SCBO in the minimum objective function value at $K = 23$ in the simple test problem, where the hybrid method reaches the global minimum. In contrast, SEGOKPLS and SEGOKPLS+K reach much lower objective function values compared to BO-IPOPT and the global optimum, but these solutions do not satisfy the problem constraints (see Fig. 4.16 on the left), as further discussed later in this section. In the constrained Ackley function, the average performance of BO-IPOPT is about 99 % better than SEGOKPLS and SCBO at $K = 100$. Notably, SEGOKPLS+K has not been tested on the constrained Ackley function. This is because, after several outer iterations, constructing its surrogate model demands a large number of points, which exceeds the limitations of our setup. Addressing this issue would require a substantial increase in either N_0 or N_{bc}, which would lead to an inconsistent comparison with the other optimizers. This improved performance highlights the advantage of integrating the local search capabilities of IPOPT with the global exploration of BO.

The higher computational cost of SCBO, SEGOKPLS, and SEGOKPLS+K can be attributed to several factors: They employ separate surrogate models for constraints and the objective function, continuously

update hyperparameters throughout optimization, and solve an additional optimization problem over a continuous space to determine new sample points at each outer iteration K. In contrast, BO-IPOPT utilizes a single surrogate model that incorporates both the objective function and constraints by keeping the hyperparameters fixed while evaluating the EI over a discrete set of candidates. This approach leads to a linear increase in CPU time over the number of outer iterations. These observations imply that, for larger benchmarks such as the DED and RSG problems, the computational time of the state-of-the-art BO methods would increase even more significantly with problem dimension and the number of constraints – further highlighting the computational efficiency and scalability benefits of the hybrid BO-IPOPT method.

The number of function evaluations is not directly compared across methods in this section, as done in Section 4.4.7. BO-IPOPT follows the same BO search framework as the other approaches, performing two function evaluations per outer iteration to ensure fair comparison. However, the hybrid method also includes additional IPOPT function evaluations, which are typically faster than those performed by BO. Consequently, despite executing more function evaluations per iteration, BO-IPOPT still requires less CPU time than competing BO methods. To achieve the total function evaluations performed by BO-IPOPT at each iteration, SCBO, SEGOKPLS, and SEGOKPLS+K would require considerably greater computational resources. For this reason, a direct comparison of total function evaluations between BO-IPOPT and the other BO approaches would be misleading and is not included in this study.

It is worth noticing that BO-IPOPT is the only approach that fulfills all constraints from the very first outer iteration K (see Fig. 4.16). The other methods struggle with the constraint satisfaction, particularly in the constrained Ackley function, where none of the BO approaches besides BO-IPOPT successfully fulfill them. For the simple test problem, SCBO satisfies all constraints at $K = 25$ (see Fig. 4.16). SEGOKPLS and SEGOKPLS+K, however, produce objective function values lower than the global optimum (see the left plot in Fig. 4.14), which arises because these solutions violate some constraints, as clearly seen in Fig. 4.16 on the left. Similarly, for the constrained Ackley function (see Fig. 4.15), SEGOKPLS fails to fully satisfy all constraints, though in this case, its minimum objective function value remains above the global optimum.

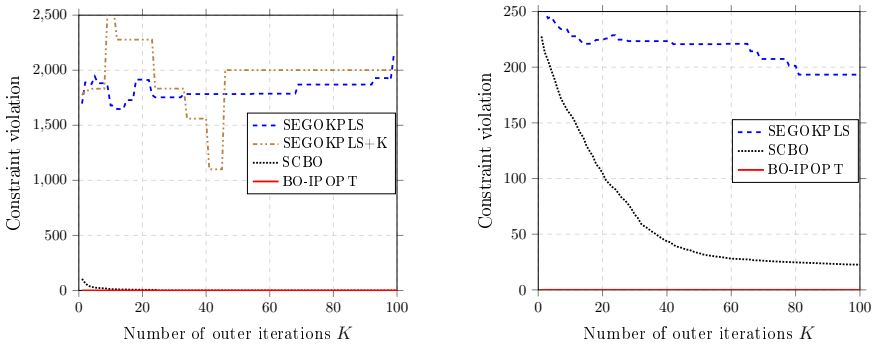

Figure 4.16: Optimization results for the simple test problem (4.11)-(4.19) (**left**) and the constrained Ackley function (4.20)-(4.21) (**right**): comparison of the constraint violation over 100 trials using SCBO, SEGOKPLS, SEGOKPLS+K, and BO-IPOPT as a function of the number of outer iterations K.

4.4.7. Comparison of MS-IPOPT, BOwLS, and BO-IPOPT

In this section, we evaluate the performance of our refined BO-IPOPT method against two competing approaches: the conventional random-based MS-IPOPT (see Algorithm 4) and BOwLS (see Algorithm 5). The comparison is conducted across four benchmark problems introduced in Section 4.3, assessing accuracy, computational efficiency (CPU time), scalability (i.e., the ability to maintain computational efficiency and find near-optimal solutions as the problem dimensionality increases), and robustness (i.e., consistent performance across repeated runs).

To ensure a fair comparison between the hybrid optimization approaches, we also employ IPOPT as the local solver for BOwLS. This adjustment is necessary because the conjugate gradient method from the SciPy package, originally used in Gao et al. (2020), has not been designed for constrained optimization problems. In the BO component of the hybrid methods, BO-IPOPT incorporates constraints using the AL framework (see (4.3)), whereas BOwLS employs a penalty term formulation (see (4.2)). For consistency across different problem types, we define $\boldsymbol{\delta} = \mathbf{0}, \boldsymbol{\tau} = \mathbf{100}$ for the two benchmark tests and $\boldsymbol{\delta} = \boldsymbol{\tau} = \mathbf{100}$ for the conceptual case studies. Further details on how BOwLS integrates (4.2) are provided in Appendix A.3.3. To keep consistency in the evaluation, both hybrid methods use the same parameter settings based on the findings from Section 4.4.3. These include $l = 100$, $\alpha = 0.1$, a monotone sequence $\xi \in [0.01, 1]$, an initial sampling size of $N_0 = 10$, a constraint handling parameter of $N_\mathrm{c} = 500$, and a batch size of $N_\mathrm{bc} = 2$. Additionally, the same acquisition function is applied to both approaches. Furthermore, BO-IPOPT employs the trust region method with the parameter settings defined in Section 4.4.5. MS-IPOPT is also configured with $N_\mathrm{bc} = 2$, i.e., two randomly generated points serve as starting locations for the local search at each outer iteration. As already mentioned in Section 4.4.4, IPOPT is used via Gams in the RSG and DED problems. For a fair comparison of the computational performance of the three optimization methods in these problems, all initial starting points in MS-IPOPT are generated in Python before being passed to Gams for solution processing.

The optimization results of all three methods across the benchmark problems are presented in Figs. 4.17-4.21, while the results corresponding to the simple test problem are presented separately in Fig. A.17 in Appendix A.3.8. The comparison includes the averaged minimum objective function value, the averaged CPU time, and the overall solver performance (i.e., averaged minimum objective function value over the averaged CPU time). Additionally, confidence intervals are displayed using the standard deviation, to evaluate the robustness of the algorithms across repeated runs. Furthermore, Figs. A.21-A.22 in Appendix A.3.11 show the evolution of the minimum objective function value with respect to the total number of function evaluations for each benchmark problem. To account for the stochastic nature of the optimizers, multiple independent runs were conducted, as in Section 4.4.4: 100 trials for both the simple test problem and the constrained Ackley function, 50 trials for the RSG problem and the DED problems with 5 and 10 units, and 20 trials for the 30-unit DED problem. It is important to highlight that while previous work (Santra et al., 2020) has reported lower best minima for the DED problems than those

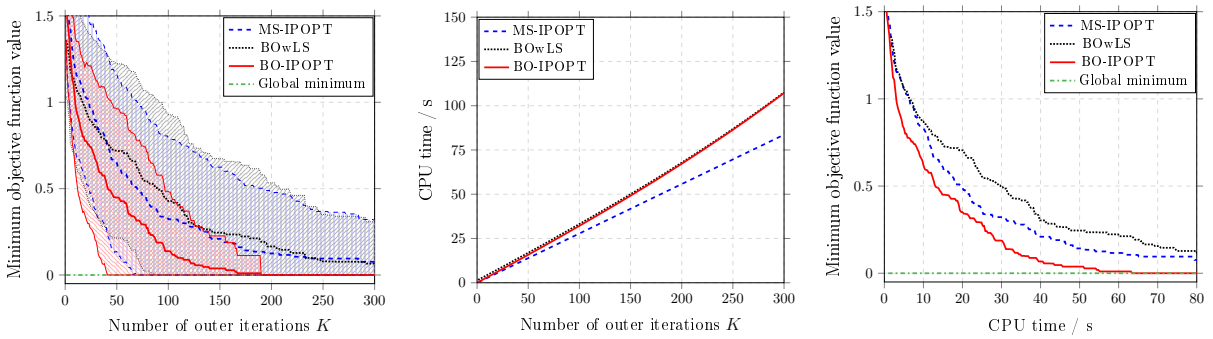

Figure 4.17: Optimization results for the constrained Ackley function (4.20)-(4.21): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer.

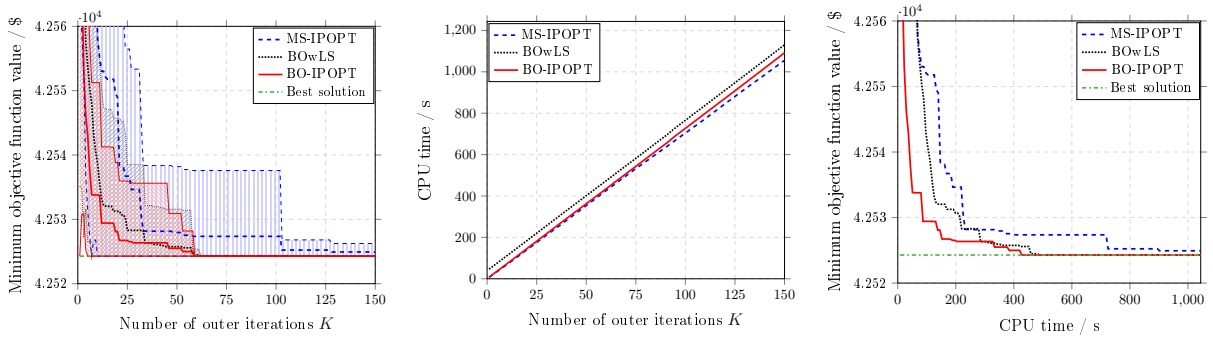

Figure 4.18: Optimization results for the DED problem (4.22)-(4.25) with 5 units: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 50 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer. Currency values are given in USD, represented by the $ sign for brevity.

achieved with the optimizers used here, a direct comparison would be misleading. The reason is that the hybrid optimization approach, employed in Santra et al. (2020) and consisting of two stochastic methods, was explicitly developed for the DED problem, whereas BO-IPOPT is designed as an optimization method applicable to a broad range of nonlinear, nonconvex, constrained problems rather than being tailored specifically for DED optimization. Consequently, comparing BO-IPOPT with problem-specific methods would not provide a fair assessment of its efficiency.

Our proposed BO-IPOPT method demonstrates superior performance in both convergence speed and robustness compared to the other two optimization approaches (see Figs. 4.17-4.21). As expected, MS-IPOPT and BO-IPOPT initially (at $K = 1$) reach higher objective function values than BOwLS, as they rely on a random initialization, whereas BOwLS leverages local optima for its starting points. Despite this initial difference, BO-IPOPT converges faster toward the global minimum, requiring fewer iterations

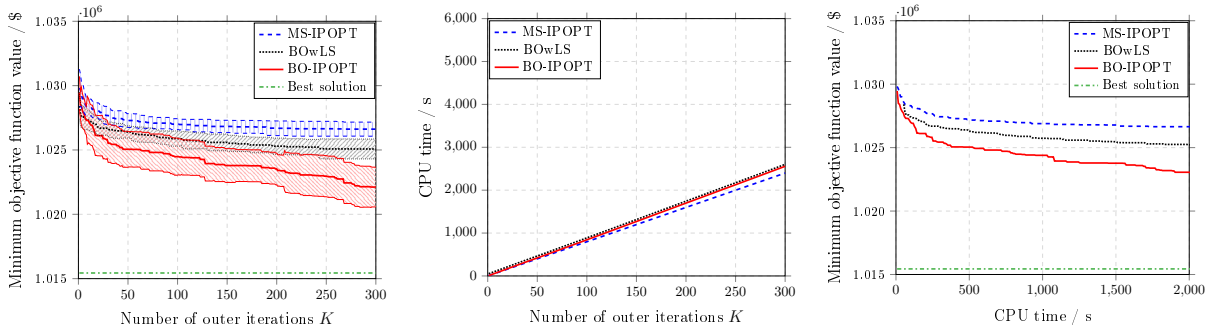

Figure 4.19: Optimization results for the DED problem (4.22)-(4.25) with 10 units: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 50 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer. Currency values are given in USD, represented by the $ sign for brevity.

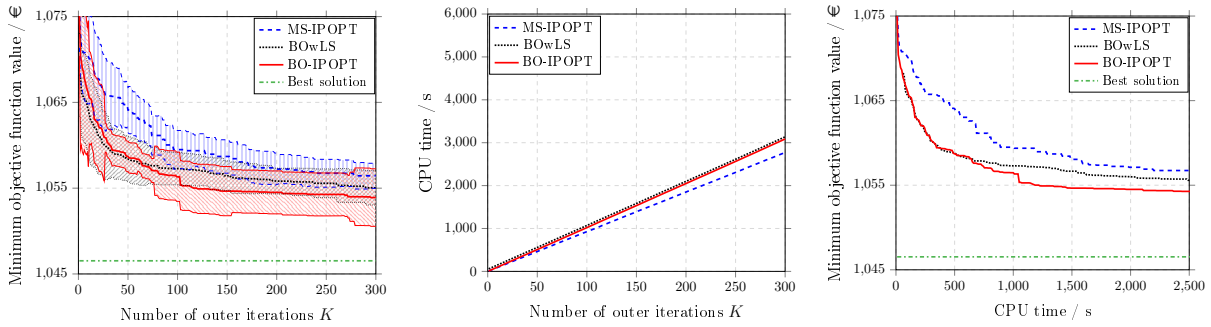

Figure 4.20: Optimization results for the RSG problem (4.26)-(4.35): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 50 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer.

K and function evaluations on average. This improved performance is particularly evident in the Ackley function and the DED problem with 10 and 30 units, which combine high dimensions, nonlinearity, and nonconvexity. While BO-IPOPT demonstrates superior performance across all investigated problems except for the simple problem, the advantage is especially pronounced in these cases due to their challenging nature and larger search spaces. In the Ackley function, BO-IPOPT achieves up to a 96 % and 93 % lower objective function values at the same CPU time compared to BOwLS and MS-IPOPT, respectively. Similarly, in the DED problems, BO-IPOPT demonstrates potential daily cost savings of up to 4,500 USD and 3,000 USD at the same CPU time compared to MS-IPOPT and BOwLS, respectively, in the case with 10 units, and up to 23,000 USD and 13,000 USD compared to MS-IPOPT and BOwLS, respectively, in the case with 30 units. This behavior can be attributed to two key factors: First, MS-IPOPT exhibits slower convergence, because it is heavily influenced by the randomly selected starting points. Second, BO-IPOPT

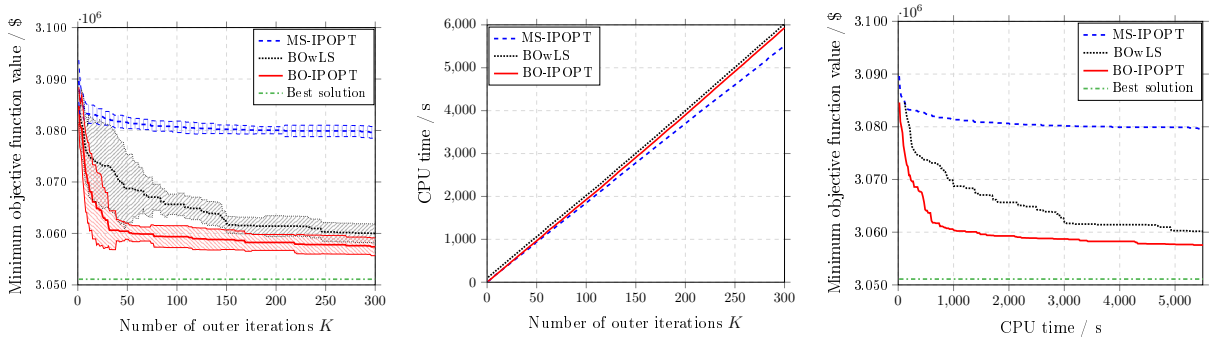

Figure 4.21: Optimization results for the DED problem (4.22)-(4.25) with 30 units: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 20 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer. Currency values are given in USD, represented by the $ sign for brevity.

benefits from a more effective surrogate model, as its sampling strategy ensures better exploration of the search space. In contrast, BOwLS focuses its sampling around the initial local optima, limiting the exploration and potentially slowing the convergence. This limitation of BOwLS is further examined in Appendix A.3.7.

Regarding robustness, BO-IPOPT outperforms both competing methods in worst-case (upper-bound) and best-case (lower-bound) performance scenarios. Notably, as the problem dimensionality increases, BO-IPOPT's worst-case performance often aligns with the average performance of the other two methods, or even approaches their best-case outcomes. It is also worth emphasizing that in the investigated cases, the selected points at each outer iteration in BO-IPOPT did not converge to the same local optimum, as demonstrated in Section A.3.12. The success of the hybrid approach, and consequently of the BO component, implies the effectiveness of the GPR. Given the large dimensions of the problems considered in this work, the results are particularly promising, as they indicate that the BO component of the hybrid method successfully overcomes the typical curse of dimensionality (i.e., the exponential increase of samples required to construct and maintain an accurate surrogate model as the dimensions increase), which frequently restricts BO in problems of high dimensions.

In terms of computational efficiency, MS-IPOPT shows better performance compared to both hybrid approaches, primarily because it does not require additional computations for evaluating the EI criterion or training the GP model. In contrast, this extra computational effort in the hybrid methods grows slightly as K increases, since the inversion of the correlation matrix in GPR becomes more computationally intensive as additional samples are incorporated at each iteration. Despite this, the CPU time for both hybrid approaches and MS-IPOPT follows an almost linear trend. As expected, BOwLS has higher computational costs than BO-IPOPT. This is due to its initialization strategy, which involves constructing the initial GPR by performing multiple local searches. As a result, the CPU time curve for BOwLS is consistently

Table 4.1: CPU time of MS-IPOPT, BOwLS, and BO-IPOPT for the 5-, 10-, and 30-unit DED problem at $K = 300$.

Optimizer	5 units	10 units	30 units
MS-IPOPT	2,110 s	2,399 s	5,512 s
BOwLS	2,260 s	2,602 s	6,024 s
BO-IPOPT	2,184 s	2,557 s	5,929 s

shifted upward by a constant factor relative to BO-IPOPT. Furthermore, this discrepancy in computational expense between the two hybrid approaches becomes increasingly pronounced as the dimensionality of the problem grows. More precisely, in the simple test problem, the overhead is approximately 1 second, while for the DED problem with 30 units, it increases to around 100 seconds.

The benchmark problems also provide valuable insights into the scalability of the optimization methods. Across all approaches, the CPU time exhibits a linear growth trend, with the rate of increase becoming more pronounced as the dimensionality of the optimization problem expands. To further analyze this, the DED problem is considered as a representative case, because it allows the evaluation of the optimizers on the same problem under different dimensions – specifically, by varying the number of units (5, 10, and 30). As shown in Table 4.1, the computational time for MS-IPOPT, BOwLS, and BO-IPOPT rises approximately linearly as the number of units in the DED problem increases, with the hybrid methods requiring more CPU time than MS-IPOPT. This observation highlights an important aspect: Although training the GP model in hybrid approaches requires additional computational effort as more samples are incorporated, its impact on the overall CPU time remains relatively minor compared to the computational cost of the local search. As the problem dimensionality increases, the computational burden of the local optimization becomes the dominant factor, accounting for roughly 82–98 % of the total CPU time depending on the problem and number of outer iterations K. This significantly outweighs the additional overhead introduced by the training of the surrogate model.

A question that arises at this point is whether and to what extent the choice of the local solver used within BO-IPOPT affects the performance of the hybrid method. To explore this, comparative tests have been conducted involving IPOPT and two widely used commercial solvers: XPRESS (Fair Isaac Corporation, 2025) and CONOPT4 (Drud, 1994). The tests were applied to two benchmark problems – the 10-unit DED problem and the RSG problem – whose results are visualized in Figs. A.25 and A.26 in Appendix A.3.13. As expected, the choice of the local solver significantly impacts both convergence speed and computational efficiency. While the commercial solvers can reduce the CPU time, they provide inconsistent outcomes in terms of minimizing the objective function value. Notably, BO-XPRESS demonstrates exceptionally rapid convergence to the best-known solution for the RSG problem. However, in the DED problem, it performs unexpectedly worse than BO-IPOPT, highlighting that commercial solvers do not always guarantee optimal performance. This observation underlines the importance of carefully selecting local solvers, as their effectiveness may vary depending on the problem at hand. A

more in-depth analysis and comparison of local solvers could be a valuable direction for future research, potentially leading to further enhancements in the hybrid optimization approach.

Overall, BO-IPOPT demonstrates superior performance compared to the other two methods (i.e., MS-IPOPT and BOwLS), as it effectively balances accuracy, robustness, and computational efficiency across the optimization problems examined in this study. Its advantages become even more pronounced as the problem complexity increases, particularly in cases where deterministic global solvers – while theoretically able to guarantee a global optimum – can not provide comparable performance within the same CPU time due to the exponential growth in computational effort with problem size. However, it is worth noticing that none of the optimizers converge (on average) to the best-known solution for the 10- and 30-unit DED problem or the RSG problem within the $K = 300$ outer iterations. This underscores the complexity of these optimization problems while also highlighting the potential for further enhancements to BO-IPOPT in future research.

4.5. Summary

This chapter introduced the novel hybrid optimization method, BO-IPOPT, which is designed to effectively tackle nonlinear, nonconvex, constrained optimization problems, especially of higher dimensions ($\gtrsim 100$ decision variables). The hybrid method combines BO with IPOPT to leverage the global search of the former and the local search of the latter. Since BO does not operate independently as usually but works alongside IPOPT in this work, several aspects have been investigated to enhance its global exploration, improve its computational efficiency, and optimize its integration with the local search capabilities of the open-source IPOPT solver.

First, we demonstrated that evaluating the EI acquisition function on a discrete candidate set – rather than relying on continuous optimization, as it is commonly done – offers a practical and efficient technique for sample selection in BO-IPOPT. This approach strikes a balance between computational cost and convergence speed, while preserving the global search capabilities of BO and better addressing the scalability challenges of high-dimensional problems. Looking ahead, we plan to enhance our framework by incorporating novel acquisition strategies, such as LogEI (Ament et al., 2023) and compositional acquisition functions (Grosnit et al., 2021), and evaluating their performance across a broad range of benchmark problems. Another promising direction involves examining the correlation between candidate points to enhance batch selection using the q-EI criterion (Wang et al., 2020b). While our current study did not encounter issues related to convergence toward the same local optima, investigating q-EI may still offer benefits by improving the diversity and overall quality of the selected candidates.

Second, we propose constructing a single surrogate model for constraint handling in BO within the AL framework, integrated with IPOPT. In contrast to approaches that rely on independent surrogate models for constraints and the objective function, our method reduces computational effort while providing effective global guidance. This approach efficiently handles both equality and inequality constraints

without requiring user-defined hyperparameters, making it a reliable solution for complex optimization problems with numerous constraints.

Third, we conducted a first comprehensive analysis of the hyperparameters included in BO-IPOPT to improve its global exploration while reducing the need for manual tuning. In contrast to conventional BO approaches that adjust these parameters dynamically during optimization – often leading to increased computational expenses – our method avoids this additional burden, ensuring a more efficient optimization process.

Fourth, we demonstrated that the GP model used within the BO-IPOPT framework, where it primarily serves as a globally smooth guiding function, achieves a performance comparable to that of a simple linear surrogate model. This highlights the different role of the GP model in BO-IPOPT compared to the classical BO, where it is typically employed to capture detailed local variations of the objective function and the constraints.

Fifth, a trust region mechanism was incorporated into the BO-IPOPT framework, where at each iteration K, a new set of candidate points is sampled within dynamically adjusted bounds (see line 7 in Algorithm 2). The integration of this adaptive trust region significantly improved the performance of the hybrid method. Specifically, it allowed BO to retain its global search capabilities while guiding IPOPT within a more focused and adaptable region.

To demonstrate the effectiveness of the proposed hybrid optimization method, we applied it to two test problems and two conceptual use cases, each differing in problem size, nonconvexity, and nonlinearity. We then compared our optimizer against advanced BO techniques, including SCBO, SEGOKPLS, and SEGOKPLS+K, as well as against the classical random-based MS-IPOPT and the hybrid BOwLS method. The results showed that BO-IPOPT significantly outperforms these state-of-the-art approaches in terms of convergence rate and robustness, achieving up to 120 % lower objective function values within the same computational time. The advantages of BO-IPOPT become even more significant in more complex problems, effectively mitigating the curse of dimensionality often encountered in traditional BO approaches.

Regarding the computational costs, BO-IPOPT is computationally more efficient than BOwLS, primarily because it eliminates the need for local searches when initializing the GP model. This does not only reduce the computational overhead but also prevents the surrogate model from being overly restricted to the initial local optima. Compared to MS-IPOPT, BO-IPOPT requires higher computational costs as more samples are incorporated, mainly due to the GP model being rebuilt at each outer iteration. However, across all benchmark problems, the increase in CPU time exhibits a linear trend relative to the number of outer iterations. Additionally, experimental evaluations using the DED problem with 5, 10, and 30 units confirm that BO-IPOPT maintains an approximately linear growth in CPU time as the problem dimensionality increases. In the hybrid method, the computational burden of the local optimization becomes the dominant factor, accounting for roughly 82–98 % of the total CPU time and significantly outweighing the additional overhead introduced by the surrogate model. This suggests that our hybrid

framework has the potential to be applied to even larger optimization problems beyond those considered in this study.

Last but not least, our hybrid approach provides the flexibility to switch between local solvers, which is particularly beneficial as various algorithms – both commercial and noncommercial – perform well in different types of problems. While this study specifically addresses problems with continuous variables, the framework could be adapted for mixed-integer optimization in future research.

5. Energy Management of Conceptual Industrial Utility System

In this chapter, the energy utility system for RSG introduced in Chapter 2 is examined to evaluate the performance of the hybrid optimization approach "BO-IPOPT" within an energy management framework, where the time available for operational optimization is limited. First, the methodology is introduced to optimize the system's operation under realistic conditions, aiming to minimize operating costs while considering uncertainties in wind data.

The chapter then analyzes the energy management strategy by comparing the performance of BO-IPOPT with alternative state-of-the-art optimization algorithms. Last, the influence of three key parameters – optimization horizon, uncertainty in wind forecasts, and CPU time (runtime) – on the system performance are investigated. By analyzing their effects, we aim to provide an insight into the practical implementation of the BO-IPOPT method for more efficient and robust energy management.

5.1. Methodology

The main goal of this chapter is to operate the industrial system for renewable steam generation described in Chapter 2 in a way that minimizes the operating costs, while accounting for uncertainties in wind data. This was first investigated for the underlying system by Kyriakidis and Bähr (2025), where a different BRAYTON-based HTHP was used. In the present work, a revised BRAYTON-based HTHP is implemented, featuring a two-stage compressor instead of a single-stage one. Additionally, updated compressor and turbine performance maps are applied, and a hybrid modeling approach is considered to better capture the component behavior (see Section 2.3). To support the operation of such systems, the tool "CoDeOpT" is employed. Originally introduced in Kansara and Roldán Serrano (2024) for design and configuration optimization, it was further extended in Kyriakidis et al. (2025a) to enable energy management functionalities based on a RHA.

As illustrated in Fig. 5.1, CoDeOpT features a modular architecture designed for industrial applications, consisting of two main modules: an offline design optimization module (Kansara and Roldán Serrano, 2024) and a real-time energy management module (Kyriakidis et al., 2025a). The design optimization module operates offline and determines the optimal system configuration, including the selection and sizing of system components.

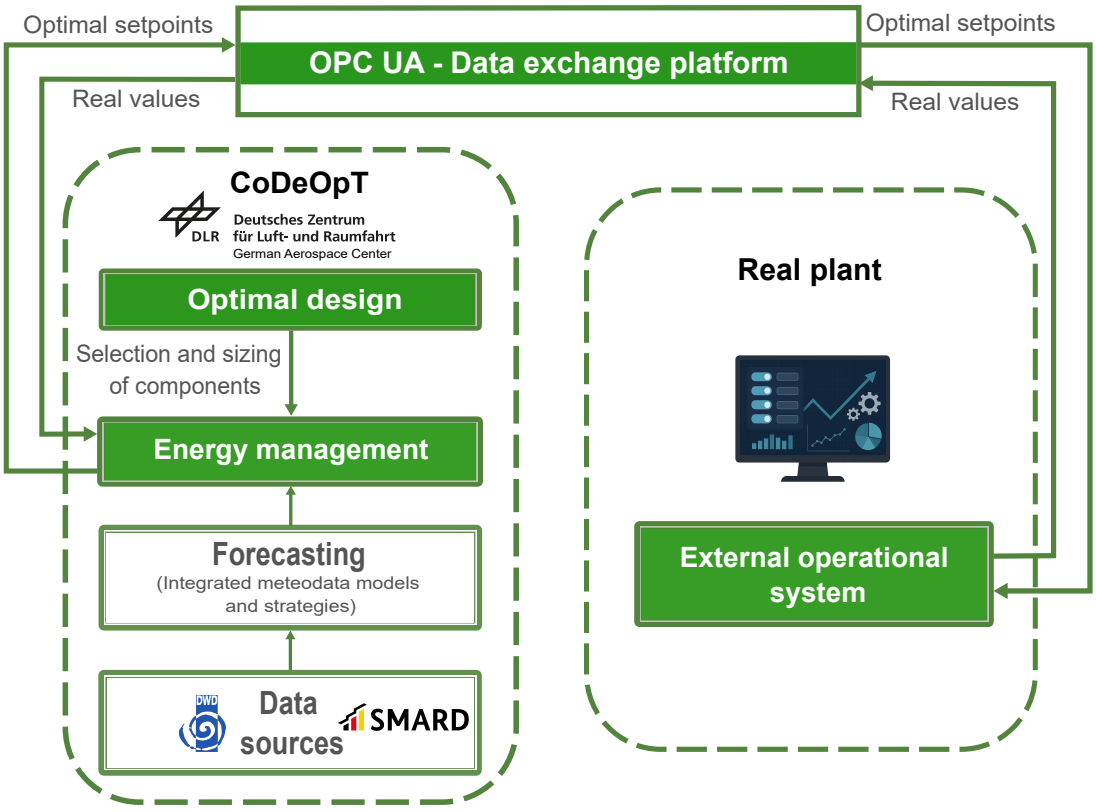

Figure 5.1: Architecture of CoDeOpT, illustrating its communication with the real plant through the OPC UA interface for exchanging real measurements and optimal setpoints.

The energy management module, on the other hand, is responsible for the operational optimization of the system in real time, integrating forecasting, detailed component and system modeling, optimization, and communication with the real plant. More precisely, the energy management module utilizes historical weather data from the GWS platform, which is then processed by the internal forecasting module to generate wind speed predictions. For electricity prices, both historical and day-ahead market data are directly obtained from the SMARD platform (SMARD, 2025). During each RHA iteration, CoDeOpT computes optimal setpoints, which are sent to the real plant via the OPC UA interface (Free OPC UA Contributors, 2023), and the plant's control system then operates the components based on these setpoints. At the same time, actual plant data are received via the same interface, allowing the system to adjust its operation dynamically in response to process deviations or unpredicted conditions. It should be noted, however, that real-time measurement feedback is not considered in this chapter. Unlike the case study investigated in Chapter 6, the present system has not been considered within the CoDeOpT framework for the design optimization module. Consequently, no design optimization has been performed for this system. Instead, all design parameters are based on data from Walden et al. (2023), since design optimization is beyond the scope of the present work.

In this study, wind speed data for Hoyerswerda (located in the eastern part of Germany) are selected for the months of January and June, including one representative week with high wind speeds and another

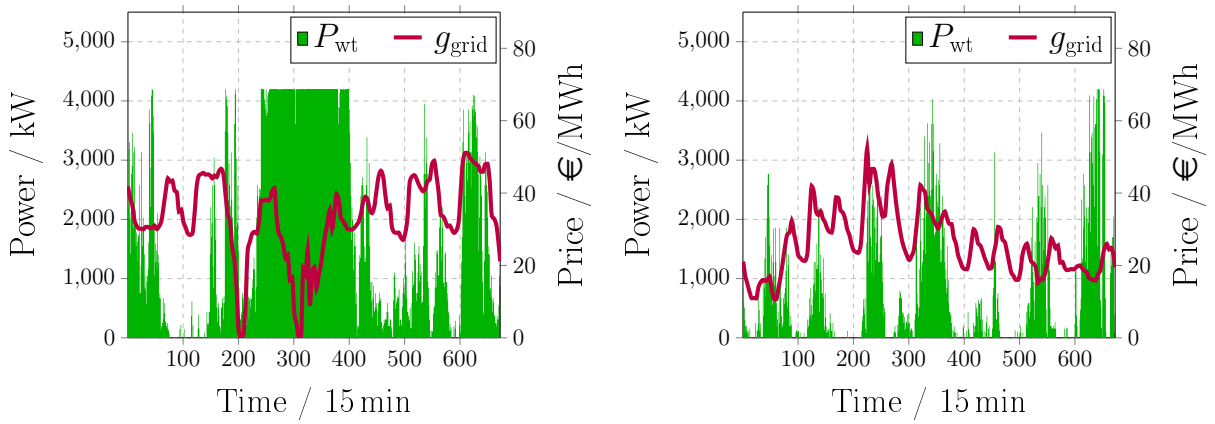

Figure 5.2: Visualization of the input data for the one-week scenario: wind power and electricity price for Hoyerswerda (Germany) in winter (**left**) and summer 2020 (**right**). Each tick on the x-axis represents a 15-minute interval.

with low wind speeds (see Fig. 5.2). To align the datasets to a common 15-minute resolution, the original wind and electricity data – initially sampled at 10-minute and hourly intervals, respectively – are resampled using monotonic cubic interpolation.

Additionally, the roughness length z_0 in (2.1) from Section 2.2 must be defined to provide a reasonable value for wind profile calculations. For this purpose, ERA5 data (Hersbach et al., 2023) are utilized. ERA5 is a global atmospheric reanalysis dataset provided by the European Centre for Medium-Range Weather Forecasts, offering hourly estimates of various meteorological variables on a high-spatial-resolution grid of approximately 31×31 km. Considering average wind speeds at heights of 10 and 100 meters from 2010 to 2020 and using (2.1) from Section 2.2, the roughness length z_0 is estimated to be approximately 0.43 meters. To determine the wind speed at 10 and 100 meters from ERA5 data, bicubic interpolation has been applied (see Appendix A.1.2). It should be noted that ERA5 data are not used directly within CoDeOpT for providing weather data, as these data are not available in real-time or sufficiently up-to-date. However, they have been used here because they provide wind speed measurements at multiple heights, which is not available from the GWS platform.

5.2. Results

The performance of the energy management is highly sensitive to various configuration parameters. These include the chosen optimization algorithm, the fixed CPU time allocated to the optimizer per RHA iteration, the accuracy of the wind forecasting models, and the selected length of the optimization horizon at each RHA iteration. In this section, we present a comparative evaluation of several stochastic and deterministic optimization techniques: GA, PSO, IPOPT with single-start (with and without warm-starting (WS)), the MS variant MS-IPOPT, and the hybrid BO-IPOPT approach. For this conceptual system, only nonlinear programming solvers are considered, while linear programming solvers applied

to linearized versions of the system are not taken into account. This is because linearized models may systematically underestimate or overestimate the true system performance, making direct objective comparisons misleading. A truly fair comparison could only be conducted if data from the real plant were available, which is not the case for this system. Moreover, linearizing the optimization problem for the conceptual system is not straightforward: the presence of numerous bilinear and trilinear terms, along with 3rd-degree polynomial HTHP surrogate models, makes the system highly nonlinear and complicates effective linearization. This technique often requires introducing additional binary and continuous variables and constraints, which can substantially increase the problem size and reduce computational efficiency.

The stochastic GA is inspired by the principle of natural selection, where populations of candidate solutions evolve over time through genetic operators such as selection, crossover, and mutation (Holland, 1975). PSO, in contrast, draws from the collective behavior observed in bird flocks or fish schools, where particles iteratively adjust their positions in the search space based on their own historical performance and that of their neighbors (Panigrahi et al., 2011). In this study, GA and PSO consider default values based on Panigrahi et al. (2011) and Solgi (2020). To handle constraints, both GA and PSO incorporate them as penalty terms in the objective function (see (4.2)). While both stochastic solvers are among the most popular choices in nonlinear optimization, especially in the context of RHA-based energy management (see Chapter 1), this work also explores more advanced strategies to assess their effectiveness under realistic operational constraints.

IPOPT, the deterministic local solver, is employed via the Pyomo modeling environment using its default settings. It is commonly used in scientific and engineering applications due to its efficiency and robustness in handling large-scale nonlinear optimization problems. However, as a local method, it can be trapped in local minima that are potentially far from the global optimum. To enhance its practical performance in solving sequences of related problems – as it is common in RHA frameworks – we also consider IPOPT with a WS strategy. WS techniques leverage information from the previously obtained solution to accelerate the convergence, especially when solving a sequence of optimization problems (Olivares et al., 2015). However, it is important to note that the quality of the WS information plays a crucial role; inappropriate initialization can lead to poor local minima rather than speed gains.

To better handle the nonconvex nature of the optimization problem, we evaluate MS-IPOPT, a MS strategy in which multiple initial points are randomly sampled and each is locally optimized using IPOPT. In our implementation, four random starting points are generated per outer iteration, matching the number of available CPU cores for parallel execution. To improve upon the purely random initialization of MS-IPOPT, we also consider the novel BO-IPOPT. Similar to MS-IPOPT, BO-IPOPT executes four IPOPT evaluations in parallel at each outer iteration. While Chapter 4 has shown that BO-IPOPT outperforms MS-IPOPT in terms of accuracy and robustness, our goal here is to compare their performance within the strict time constraints of real-time energy management scenarios. BOwLS is not included in this comparison because it is also a hybrid method, whereas BO-IPOPT represents a more advanced and efficient variant, as demonstrated in Section 4.4.7. All parallel executions are implemented using Python's

MULTIPROCESSING module. To ensure a fair comparison with the two stochastic solvers, the stochastic part of BO-IPOPT is not parallelized, matching the GA and PSO configurations.

All algorithms were implemented in PYTHON 3.12, with experiments running on a system equipped with an Intel(R) Core(TM) i7-8665U CPU. Each optimizer, except for IPOPT and WS-IPOPT, is constrained to a fixed CPU time per RHA iteration to identify the optimal trajectories for the system's setpoints. In contrast, IPOPT and WS-IPOPT are initialized only once to perform a local search and therefore, they typically require less time – about two seconds on average at each RHA iteration – compared to the other solvers.

Given the inherent randomness in the implemented algorithms, each experiment was repeated ten times to average out the stochastic variability. The baseline scenario assumes a 24-step prediction horizon, corresponding to a 6-hour prediction window with 15-minute intervals, resulting in an optimization problem with 480 decision variables. This horizon offers a balance between including a sufficient number of future steps for effective operational planning and the reliability of wind forecasting models (≤ 6 h ahead as mentioned in Chapter 3). The sensitivity of the optimization performance to both CPU time and optimization horizon length is investigated later in this section. Since the global optimum of the underlying problem is unknown and the use of a global deterministic solver (e.g., BARON) becomes computationally impractical for problems of this scale (over 100 decision variables), the best solution found among MS-IPOPT, BO-IPOPT, GA, and PSO with a 5-minute CPU time per RHA iteration is used as a practical benchmark for evaluating all algorithms.

The first step in the analysis focuses on comparing the performance of the different optimizers integrated into the RHA framework. To ensure a fair and consistent comparison, all numerical experiments are conducted without introducing uncertainties into the input data. Additionally, the CPU time for solving the optimization problem at each RHA iteration is fixed at 15 seconds. This time limit was chosen based on the moderate dimensionality of the problem. It also ensures that the solutions can be computed quickly enough – within the approximately 60-second computation window defined in this work – to allow the transfer of control signals and system responses during real-time operation.

The results for both the winter and summer test weeks are illustrated in Fig. 5.3. These box plots depict the distribution of the accumulated objective function values obtained across repeated experiments for each optimization method. Among all tested solvers, BO-IPOPT achieves the lowest operating costs (up to 35 % lower than the stochastic solvers and up to 4 % lower than the IPOPT-based methods) and shows the smallest spread in the results, reflecting a more consistent and robust performance. Specifically, during the winter week, BO-IPOPT achieves on average (measured by the median) a relative error of only 0.6 % with respect to the best solution, compared to approximately 54 % for GA, 55 % for PSO, 5 % for IPOPT, 4 % for WS-IPOPT, and 2 % for MS-IPOPT. For the summer week, BO-IPOPT maintains a superior performance with an average error of about 0.2 %, while GA, PSO, IPOPT, WS-IPOPT, and MS-IPOPT reach relative errors of roughly 8 %, 9 %, 2 %, 2 %, and 1 %, respectively.

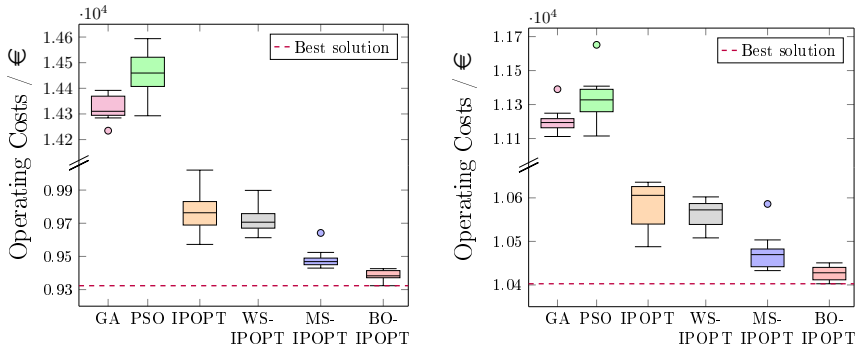

Figure 5.3: Comparison of operating costs over 10 trials for different optimizers in winter (**left**) and summer (**right**) with an optimizer CPU time of 15 seconds per RHA iteration, a 24-step optimization horizon, and assuming that the input data is known. IPOPT and WS-IPOPT require only about two seconds on average at each RHA iteration. The circles in the box plots represent outliers.

It is worth noticing that BO-IPOPT is the only optimization method that consistently reaches the benchmark solution across both scenarios. As expected, the MS approach outperforms single-start strategies such as IPOPT and WS-IPOPT, thanks to its enhanced exploration capabilities. However, MS-IPOPT still falls short of BO-IPOPT, primarily due to its reliance on randomly generated initial points. Among the single-start methods, WS-IPOPT produces a noticeably narrower range of outcomes than IPOPT, as it benefits from better-informed initial guesses that reduce sensitivity to initialization and improve consistency across runs. The stochastic optimizers GA and PSO, on the other hand, face following challenges: their population-based search mechanisms and reliance on random variation require more iterations to identify feasible solutions with low objective function values. Under the strict time constraints per RHA iteration, they are often unable to sufficiently explore the search space or resolve constraint violations – resulting in infeasible solutions, with constraint violations on the order of 10^6, and consequently higher operating costs compared to the other solvers. BO-IPOPT's superior performance can be attributed to its hybrid structure: BO provides good initial guesses via global exploration, which are then refined by IPOPT to ensure fast local convergence and constraint satisfaction.

It is important to note that higher wind speeds in winter introduce greater variability, which complicates the optimization task due to the greater flexibility in system operation and the resulting increase in possible local minima for the optimizer. This added difficulty amplifies the differences in performance between the solvers, making the winter scenario more challenging than the summer one.

Although IPOPT and WS-IPOPT are designed as single-start local optimizers, they do not always converge successfully on the first run in practice. As a result, restarts are necessary to reach a feasible solution. This behavior was also noticeable in the investigated energy utility system for RSG: IPOPT exhibited a noticeable frequency of restarts, particularly in the winter scenario, where the increased variability made the optimization problem more challenging to solve. In the winter scenario, on average about 24 % of the RHA iterations required at least one restart, and in the summer week, this share decreased to approximately 10 %. Of these, 17 % (winter) and 8 % (summer) on average failed more than once.

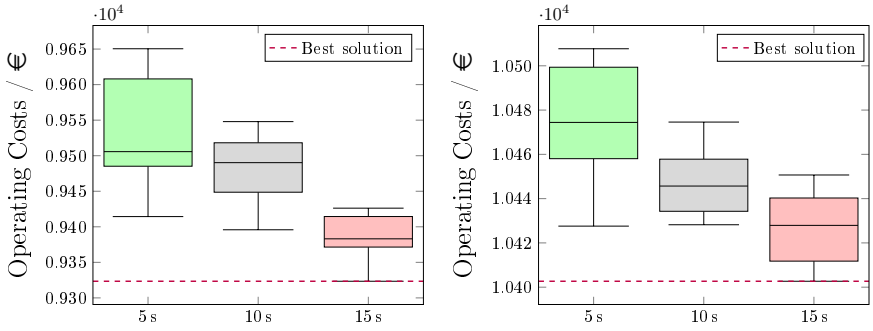

Figure 5.4: Comparison of operating costs over 10 trials for different CPU times of BO-IPOPT in winter (**left**) and summer (**right**) with a 24-step optimization horizon and assuming that the input data is known.

WS-IPOPT demonstrated improved robustness, with only 10 % (winter) and 6 % (summer) of the RHA iterations needing a restart. However, even this approach experience failures, with 12 % and 5 % of the unsuccessful runs, respectively, failing multiple times. These results confirm that, while WS improves reliability, the convergence remains a challenge. This further supports the advantage of BO-IPOPT, which not only improves the solution quality but also enhances the convergence to a feasible solution at each RHA iteration. Given the consistent superiority of BO-IPOPT in both cases, it is selected for the subsequent analyses.

It is often necessary to further reduce the optimizer's CPU time in order to ensure that the optimized operational strategy is available in time to effectively respond to rapidly changing system states and requirements. Therefore, we next evaluate the impact of shorter CPU time limits, still assuming no uncertainties in the input data and maintaining a 24-step optimization horizon. As shown in Fig. 5.4, reducing the allowed CPU time per RHA iteration for BO-IPOPT to 10 or 5 seconds results, as expected, in a slight increase in the accumulated operating costs. However, this increase remains relatively small – on average up to 1.3 % for the winter week and up to 0.4 % for the summer week when considering CPU times down to 5 seconds. Shorter CPU times also lead to wider confidence intervals, indicating greater variability and uncertainty in the optimization outcomes. These findings indicate that while longer CPU times per RHA iteration provide the most reliable and consistent results, even very short times can still be applied if stricter time requirements are necessary. For the subsequent investigations, a CPU time of 15 seconds per RHA iteration is considered, as it offers a good balance between computational efficiency and solution robustness, reflected in narrower confidence intervals, while also remaining within the time limit (approximately 60 seconds per RHA iteration for the optimal setpoint determination) defined in this thesis.

As previously discussed, incorporating uncertainties in the input data is essential for a more realistic assessment. To address this, wind speed forecasting models are now integrated into the RHA. In this context, MLR, HGBR, and KNN using the Rec forecasting strategy are evaluated, as they demonstrated strong performance in multi-step forecasting scenarios, while maintaining low computational demands (under 4 seconds) in Chapter 3. Additionally, the CNN-LSTM model is applied due to its good forecasting

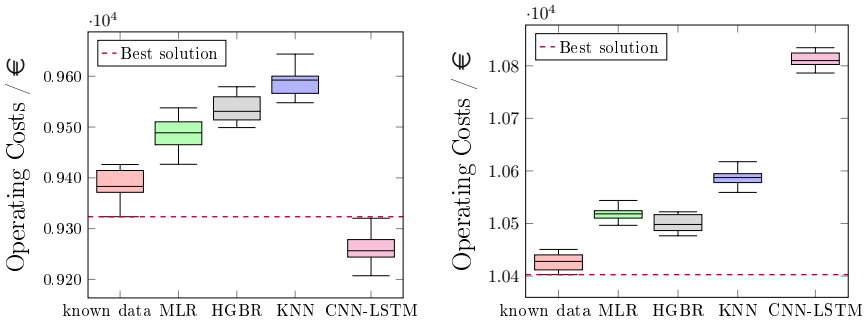

Figure 5.5: Comparison of operating costs over 10 trials for different forecasting models for the wind data using BO-IPOPT with a CPU time of 15 seconds and a 24-step optimization horizon in winter (**left**) and summer (**right**).

accuracy. However, because of its high computational cost, it is trained only once at the beginning of the energy management process and not retrained at every RHA iteration like the other models.

The results in Fig. 5.5, obtained with an optimizer CPU time of 15 seconds per iteration (excluding forecasting model training time), show that incorporating wind forecast uncertainties with MLR, HGBR, and KNN results in slightly higher accumulated operating costs for both winter and summer weeks compared to the ideal case with known wind data. This is mainly due to the tendency of these models to underestimate the wind availability, leading to increased grid electricity consumption and, consequently, higher operating expenses. On average, the cost increase reaches up to 2.1 % in winter and 1.5 % in summer. In contrast, the CNN-LSTM model shows a 1.5 % cost reduction in the winter week due to overestimation of wind availability, but shows a 3.7 % cost increase during the summer. The suboptimal performance of this model is attributed to the lack of retraining at each RHA, which limits the model's adaptability to evolving data patterns and reduces the forecasting accuracy over time. It is worth noticing that periodic retraining of the CNN-LSTM model offline and in parallel with the energy management could help maintain high forecasting accuracy. However, this was not considered within the scope of the present work.

Overall, the simple MLR model performs very well, with operating costs deviating by up to 1 % from the operating costs computed assuming known data, while also requiring minimal training effort. For this reason, MLR is selected for the final analysis presented below. It is also worth noticing that the confidence intervals across all forecasting methods are comparable.

The final parameter investigated in this study is the length of the optimization horizon, which plays a crucial role in determining the operation strategy implemented through the RHA-based energy management approach. The horizon length affects the performance in two key ways: On the one hand, a longer horizon allows the optimization to better incorporate future system states, such as variations in renewable energy generation and electricity prices, thereby enabling more effective storage utilization. On the other hand, extending the horizon increases the dimensionality and consequently the computational effort required to solve the optimization problem, and also amplifies the impact of forecast uncertainty. These trade-offs are particularly relevant in practical applications, where energy management must balance short-term

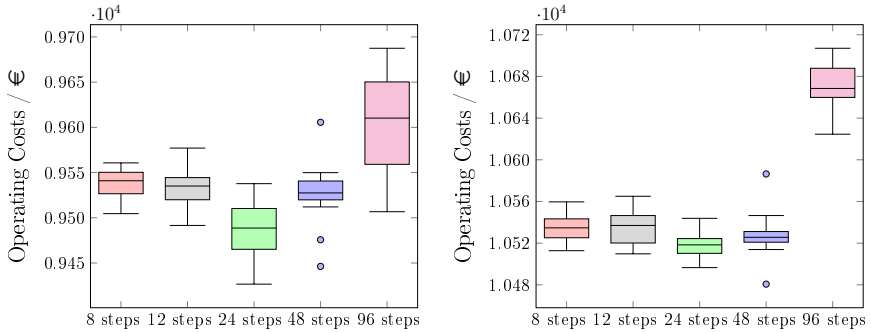

Figure 5.6: Comparison of operating costs over 10 trials for different optimization horizons using BO-IPOPT with a CPU time of 15 seconds and the MLR method for the wind data forecast in winter (**left**) and summer (**right**).

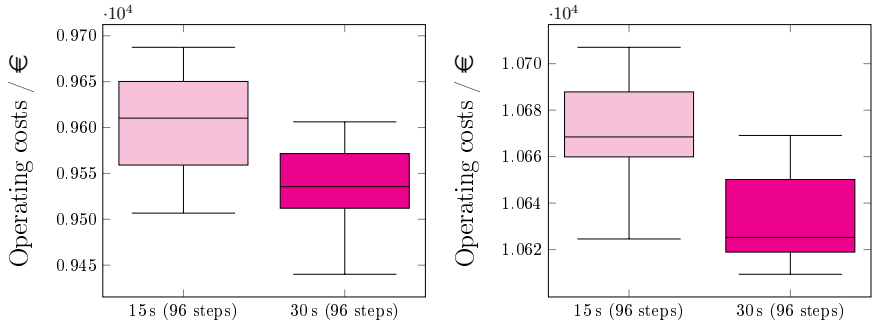

Figure 5.7: Comparison of operating costs over 10 trials for different CPU times using BO-IPOPT with a 96-step optimization horizon and the MLR method for the wind data forecast in winter (**left**) and summer (**right**).

responsiveness with long-term planning goals. To investigate these effects, numerical experiments were conducted using horizon lengths of 12, 24, 48, and 96 steps – corresponding to 3, 6, 12, and 24 hours – while maintaining a fixed time resolution of 15 minutes. Moreover, a CPU time limit of 15 seconds per RHA iteration was set for the BO-IPOPT. As shown in Fig. 5.6, extending the horizon from 8 to 24 steps improves the operating cost results, with median values decreasing and variability remaining low. Nevertheless, the reduction in cost when increasing the horizon from 8 to 24 steps is relatively modest, resulting in only 0.55 % in winter and 0.18 % in summer. Increasing the horizon to 48 steps introduces slightly greater variability in the results, as the optimization problem grows in size and becomes more challenging to solve within the time limits. Despite this, the operating costs mean increase is relatively small – only 0.41 % in winter and 0.07 % in summer. When the horizon is extended to 96 steps, however, both the mean operating costs and the variability of the results increase. This is attributed to the higher dimensionality of the optimization problem, which becomes more challenging to solve accurately within the 15-second time limit. Allowing a longer optimization time – e.g., 30 seconds – improves the solution quality, as reflected by reduced operating costs shown in Fig. 5.7.

As shown in Fig. 5.8, where a 96-step optimization horizon with and without uncertainties is tested, the results indicate that the energy management is not significantly affected by extending the forecasting horizon. This is because the RHA primarily relies on the first step of each forecast, making decisions

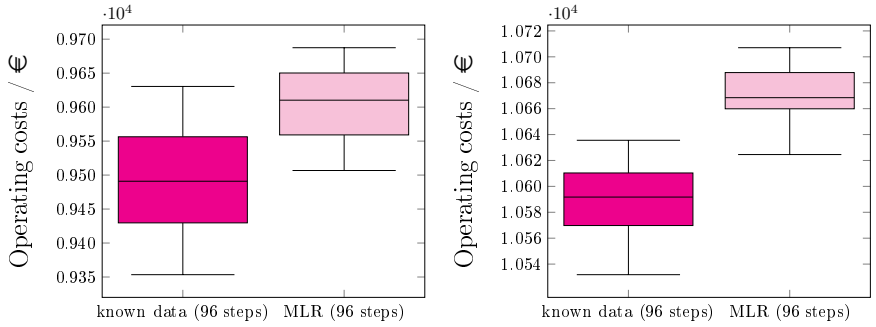

Figure 5.8: Comparison of operating costs over 10 trials using BO-IPOPT with a CPU time of 15 seconds, a 96-step optimization horizon, and the MLR method for the wind data forecast in winter (**left**) and summer (**right**). The results compare the case without forecast uncertainties to the case with uncertainties.

relatively insensitive to more distant predictions. The increase in cost when accounting for uncertainties is modest – approximately 1.3 % in winter and 0.7 % in summer – and aligns with the results obtained for the 24-step horizon. The difference is slightly larger for the 96-step case due to the longer horizon, but the effect remains minor.

While these findings are specific to the energy utility system analyzed here, the results highlight the robustness of BO-IPOPT under tight time constraints and its reliable performance across horizon lengths ranging from 8 to 48 steps. The horizon of 24 steps offered the best trade-off between predictive planning and computational feasibility, although horizons of 8, 12, and 48 steps yielded results that were close in performance and not significantly worse. Moreover, the results in this section underscore the effectiveness of a simple MLR model in handling uncertainties in wind data with high accuracy and minimal computational overhead. These insights may generalize to other systems of comparable complexity. However, for higher-dimensional systems, longer optimization times may be necessary due to increased problem sizes, or alternatively, shorter horizons may be required to maintain real-time feasibility – an aspect that will be explored in next chapter in a real-world energy utility system.

6. Energy Management of Real-World Industrial Utility System

In this chapter, a real-world industrial energy utility system is investigated to assess the performance of the hybrid optimization method "BO-IPOPT" within an energy management framework. The method is applied in a practical setting to evaluate its effectiveness under real operational conditions.

The chapter begins by introducing the system configuration, including the modeling of key components and the formulation of the corresponding optimization problem. Then, forecasting models are presented and compared to account for uncertainties in system operation, particularly those related to renewable energy generation. This is followed by an energy management analysis that compares the performance of BO-IPOPT with other state-of-the-art optimization algorithms.

Finally, the impact of factors such as optimization horizon, uncertainties in input data, and CPU time on the system performance is systematically investigated, as previously done for the conceptual energy utility system in Chapter 5.

6.1. Use Case

This section presents the industrial energy system analyzed in this work, including a detailed overview of its configuration, the modeling approach for each component, and the formulation of the associated operational optimization problem.

6.1.1. System Description

The use case investigated in this study is a company located in Herzberg, in the German state of Brandenburg, operating in the food and cosmetics industry. As part of its strategy to decarbonize both heat and electricity consumption within the European project "SINNOGENES" (SINNOGENES, 2025), the company has recently implemented a comprehensive retrofit of its energy infrastructure. This retrofit integrates renewable energy sources and additional system components, enabling a more sustainable energy supply. The energy management of the resulting system – illustrated in Fig. 6.1 – is analyzed in this work with the goal of minimizing operating costs and CO_2 emissions through optimal operation.

The system represents the integration of renewable technologies and TESs, designed to decarbonize the facility's energy supply. The system's configuration is displayed in Fig. 6.1. At its core is a STC, which charges two stratified TES units (TES 1 and TES 2). These tanks are hydraulically connected to the

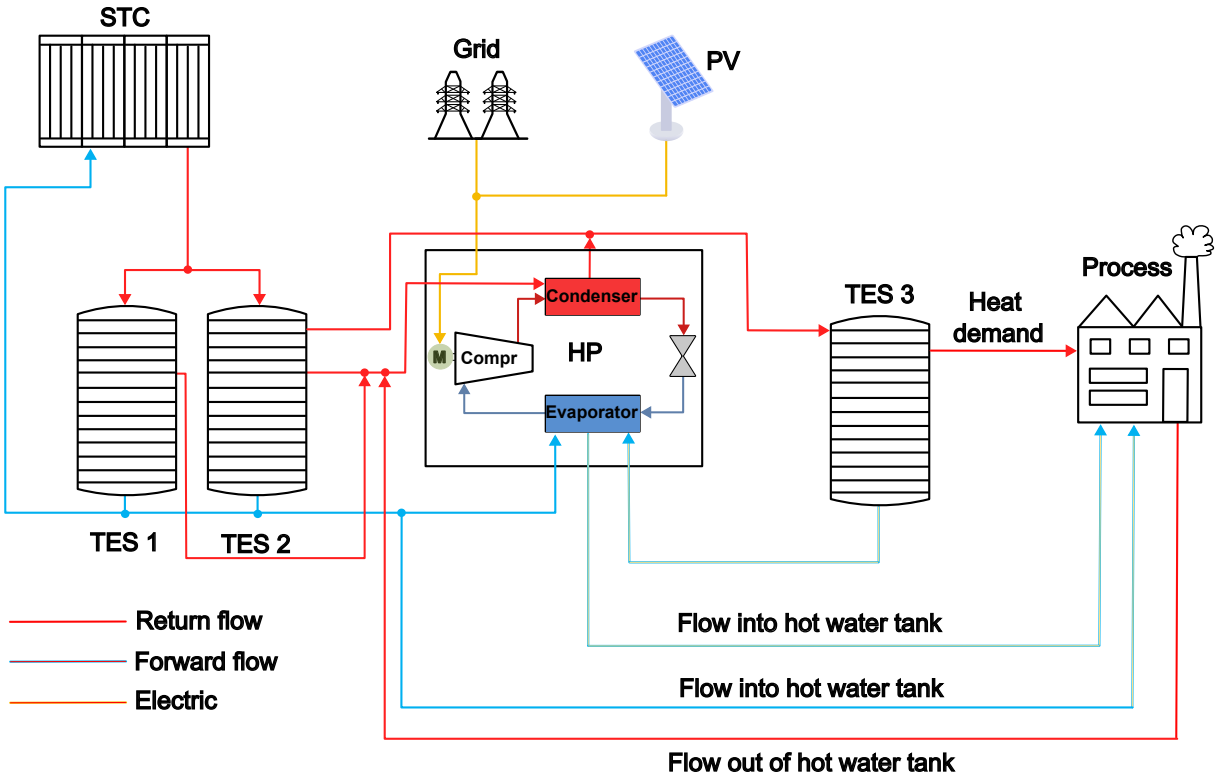

Figure 6.1: Visualization of the real-world industrial energy utility system.

STC: water from the lower layers of the TES is circulated to the STC inlet, and after heating, returned to the upper layers of the TESs – preserving stratification and improving storage efficiency.

The stored thermal energy is used in several ways. A portion supplies both the source and sink sides of a HP, which operates using electricity from an on-site PV system and, if required, the electrical grid. Additionally, the HP's source side is supported by a separate hot water tank located within the process, which can also be partially charged using thermal energy from the initial two TES units. If the PV production exceeds the on-site demand, the excess electricity can either drive the HP to charge a third stratified TES unit (TES 3) or, if not needed, be sold to the market. Moreover, the flow leaving the evaporator on the secondary side – with temperatures up to about 42 °C – is directed to the hot water tank in the process for storage and later use.

The HP upgrades the low-temperature thermal energy – characterized by source side temperatures up to 45 °C – to higher temperatures (up to 90 °C) suitable for industrial use. The thermal output is stored in TES 3, which serves as the primary thermal buffer for the industrial process. It delivers hot water at 88–90 °C and a flow rate of 3,000–3,500 l/h. This corresponds to a maximum thermal power output of approximately 260 kW. This third TES can also receive excess heat from the first two TES units when their top-layer temperatures are sufficiently high. Conversely, during low solar availability or peak demand, the lower layers of the third TES can return heat to the HP's source side, enhancing the system flexibility. All TES units operate within a temperature range of approximately 25–90 °C, allowing for effective thermal

stratification and dynamic charging/discharging strategies. In addition, the hot water tank in the process operates with temperature levels ranging from 25 °C to 60 °C. Water is used as the heat transfer medium and storage fluid throughout the system due to its compatibility with all components and favorable thermal properties.

6.1.2. Component Modeling

To assess the performance and enable the optimization of the renewable energy utility system presented in the previous subsection, detailed modeling of individual components is required. This involves developing mathematical representations for the STC, PV, HP, and TES units. These models serve as the foundation for the optimization problem and are designed to reflect the physical behavior of each component.

Solar Thermal Collector

The STC model estimates the thermal energy output based on the incident solar irradiance, collector area, optical and thermal properties, and relevant temperature differentes. The thermal output of the collector is defined as follows by Kansara and Roldán Serrano (2024):

$$\begin{aligned} \dot{m}_{\text{STC}}\, c_{p,\text{water}}(T_{\text{STC,in}}, T_{\text{STC,out}})\, (T_{\text{STC,out}} - T_{\text{STC,in}}) = \\ (1 - \gamma_{\text{STC}})\, A_{\text{STC}} \left[K_{\text{STC}}\, I\, \eta_{\text{STC,opt}} - \beta_{\text{STC},0} \sum_{i=1}^{4} \beta_{\text{STC},i}\, \Delta T^{i}_{\text{STC,amb}} \right], \end{aligned} \tag{6.1}$$

where $\dot{m}_{\text{STC}}$ is the mass flow rate through the collector, and $c_{p,\text{water}}(T_{\text{STC,in}}, T_{\text{STC,out}})$ denotes the temperature-dependent specific heat capacity of water. The outlet and inlet temperatures of the collector are represented by $T_{\text{STC,out}}$ and $T_{\text{STC,in}}$, respectively, while A_{STC} is the collector area. The variable I refers to the solar irradiance incident on the collector, and $\eta_{\text{STC,opt}}$ defines its optical efficiency. Thermal losses are captured by polynomial terms with coefficients $\beta_{\text{STC},i}$, corresponding to different orders of the temperature difference. The variable γ_{STC} accounts for periods during which the collector is either partially or fully deactivated and takes continuous values in the range $[0, 1]$.

The term K_{STC} denotes the incidence correction factor, which adjusts for the angle at which the solar irradiance strikes the collector surface. It is determined using the following equation from Kansara and Roldán Serrano (2024):

$$K_{\text{STC}} = 1 - \frac{1 - \alpha_{\text{STC}}}{0.55} \left(\frac{1}{\cos(\theta_{\text{STC}})} - 1 \right), \tag{6.2}$$

where θ_{STC} is the angle of incidence and α_{STC} an empirical coefficient. The temperature difference between the collector and ambient, $\Delta T_{\text{STC,amb}}$, is calculated as:

$$\Delta T_{\text{STC,amb}} = \frac{T_{\text{STC,out}} + T_{\text{STC,in}}}{2} - T_{\text{amb}}, \tag{6.3}$$

where T_{amb} is the ambient temperature. All parameters, summarized in Table 6.1, are derived from the collector's technical specifications and Solar Keymark Database (2025).

Table 6.1: STC parameters.

Parameter	Value	Unit
A_{STC}	135	m^2
$\eta_{\text{STC,opt}}$	0.715	1
α_{STC}	0.97	1
$\beta_{\text{STC,0}}$	1.1	1
$\beta_{\text{STC,1}}$	3.31	$\text{W/(m}^2\,\text{K)}$
$\beta_{\text{STC,2}}$	0.011	$\text{W/(m}^2\,\text{K}^2\text{)}$
$\beta_{\text{STC,3}}$	0	$\text{W/(m}^2\,\text{K}^3\text{)}$
$\beta_{\text{STC,4}}$	0	$\text{W/(m}^2\,\text{K}^4\text{)}$
θ_{STC}	32	$^\circ$

Photovoltaics

The PV system is modeled to calculate its power output based on the incident solar irradiance, the total module area, and the combined efficiencies of the PV modules and inverter. The PV power output can be expressed as:

$$P_{\text{PV}} = (1 - \gamma_{\text{PV}})\, A_{\text{PV}}\, I\, \eta_{\text{PV,mod}}\, \eta_{\text{PV,inv}}, \tag{6.4}$$

with

$$\eta_{\text{PV,mod}} = \left(\eta_{\text{PV,nom}} + a_{\text{PV,1}} \ln\left(\frac{I}{I_{\text{PV,nom}}}\right)\right)\left(1 + a_{\text{PV,2}}\left(T_{\text{amb}} + a_{\text{PV,3}} I - T_{\text{PV,nom}}\right)\right). \tag{6.5}$$

In this context, A_{PV} represents the PV module area, I denotes the solar irradiance, $\eta_{\text{PV,inv}}$ is the inverter efficiency, and $\eta_{\text{PV,nom}}$ refers to the nominal efficiency of the PV module. The empirical coefficients $a_{\text{PV,1}}$, $a_{\text{PV,2}}$, and $a_{\text{PV,3}}$ are introduced to account for the part-load performance of the module under varying irradiance and temperature conditions. Additionally, T_{amb} indicates the ambient temperature, $I_{\text{PV,nom}}$ is the nominal solar irradiance, and $T_{\text{PV,nom}}$ is the nominal temperature. The γ_{PV} variable reflects any operational restrictions or deactivation of the PV system and takes continuous values in the range $[0, 1]$. All parameters, which are summarized in Table 6.2, are based on the design values used in the experimental setup and are supported by data from King et al. (2007), Jacobson and Jadhav (2018), Canadian Solar (2022), and DLR Solar Research (2024). The installed PV system has a nominal peak power capacity of 55 kWp.

Table 6.2: PV parameters.

Variable	Value	Unit
A_{PV}	500	m^2
$P_{\text{PV,peak}}$	55	kWp
$\eta_{\text{PV,nom}}$	0.212	1
$\eta_{\text{PV,inv}}$	0.95	1
$a_{\text{PV,1}}$	0.06408	1
$a_{\text{PV,2}}$	-0.0035	1/K
$a_{\text{PV,3}}$	0.0275	$(\text{K m}^2)/\text{W}$
$T_{\text{PV,nom}}$	42	°C
$I_{\text{PV,nom}}$	1,000	W/m^2

Heat Pump

The HP is represented using surrogate models that approximate two key output quantities: the outlet temperature on the sink side $T_{\text{h,out}}$ and the electrical power consumption P_{el}. These models were derived from the manufacturer-provided performance data, available through the project "SINNOGENES" (SINNOGENES, 2025), using second-order polynomial regression.

The resulting surrogate models exhibit high accuracy over the entire operational range, including part-load conditions. The coefficient of determination (R^2) reaches 99.9 % for $T_{\text{h,out}}$ and 99.6 % for P_{el}, indicating excellent predictive capability. The main influencing variables based on the p-value and AIC methods (see Section 3.2) are the inlet temperatures on the source and sink sides ($T_{\text{c,in}}$ and $T_{\text{h,in}}$), along with the mass flow rate on the sink side ($\dot{m}_{\text{h,in}}$), all of which have a strong impact on the HP's thermal and electrical behavior. The sink-side outlet temperature is estimated by the following regression-based function:

$$T_{\text{h,out}} = f(T_{\text{c,in}}, T_{\text{h,in}}, \dot{m}_{\text{h,in}}, \dot{m}^2_{\text{h,in}}, T_{\text{c,in}}T_{\text{h,in}}, T_{\text{c,in}}\dot{m}_{\text{h,in}}, T_{\text{h,in}}\dot{m}_{\text{h,in}}). \tag{6.6}$$

The electrical power input is modeled as:

$$P_{\text{el}} = f(T_{\text{c,in}}, T_{\text{h,in}}, \dot{m}_{\text{h,in}}, T^2_{\text{c,in}}, T^2_{\text{h,in}}, \dot{m}^2_{\text{h,in}}, T_{\text{c,in}}T_{\text{h,in}}, T_{\text{c,in}}\dot{m}_{\text{h,in}}, T_{\text{h,in}}\dot{m}_{\text{h,in}}). \tag{6.7}$$

These surrogate models are integrated into the overall optimization framework, ensuring both physical fidelity and computational efficiency.

Thermal Energy Storage

Each TES unit in this study is represented using a physics-based multi-node 1-D model (Cadau et al., 2019). In this approach, the storage tank is divided into N vertical segments (nodes), with each node characterized by a uniform temperature T_i. This structure captures vertical temperature gradients while disregarding horizontal temperature differences. For each node, the model applies volume and energy

balance equations, accounting for both mass flow and heat transfer processes. The general forms of these governing equations are expressed as follows:

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \sum \dot{V}(t) = 0, \tag{6.8}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum h(t)\dot{m}(t) + \sum \dot{Q}(t), \tag{6.9}$$

where the summations run over all mass/volume flows and all heat-transfer contributions affecting the node. (6.8) enforces volume conservation within each node, while (6.9) describes energy conservation, accounting for both enthalpy flows and external heat transfer – including interactions between neighboring nodes and losses to the environment. The dynamic volume and energy balances for a generic node i are formulated as follows by Cadau et al. (2019):

$$\dot{V}_{\text{vert,out}} = \frac{\dot{m}_{\text{in}}}{\rho(T_{\text{in}})} + \dot{V}_{\text{vert,in}} - \frac{\dot{m}_{\text{out}}}{\rho(T_i)}, \tag{6.10}$$

$$\begin{aligned} m_i\, c_p(T_i)\, \frac{\mathrm{d}T_i}{\mathrm{d}t} = {} & \dot{m}_{\text{in}}\, c_p(T_{\text{in}})\, T_{\text{in}} - \dot{m}_{\text{out}}\, c_p(T_{\text{out}})\, T_{\text{out}} \\ & - \dot{V}_{\text{vert,out}}\, \rho(T_{\text{vert,out}})\, c_p(T_{\text{vert,out}})\, T_{\text{vert,out}} \\ & + \dot{V}_{\text{vert,in}}\, \rho(T_{\text{vert,in}})\, c_p(T_{\text{vert,in}})\, \dot{V}_{\text{vert,in}} \\ & + k(T_{i-1}, T_i)\, (T_{i-1} - T_i) - k(T_i, T_{i+1})\, (T_i - T_{i+1}) \\ & - U A\, (T_i - T_{\text{amb}}). \end{aligned} \tag{6.11}$$

In this context, m_i denotes the fluid mass in node i, while $c_p(T)$, $\rho(T)$, and $k(T)$ represent the temperature-dependent specific heat capacity, density, and thermal conductivity, respectively. The terms $\dot{m}_{\text{in}}$ and $\dot{m}_{\text{out}}$ describe external mass flows, i.e., fluid entering or leaving node i from connected components such as STS, HP, or other TES. In contrast, $\dot{V}_{\text{vert,in}}$ and $\dot{V}_{\text{vert,out}}$ represent internal vertical volume flows between neighboring nodes within the same TES unit, with the associated temperatures $T_{\text{vert,in}}$ and $T_{\text{vert,out}}$. The parameters U and A account for heat losses of each node to the environment, where U is the overall heat transfer coefficient of the insulation and A is the heat transfer surface area. For this study, U is set to 0.043 kW/(m^2 K), and A is specified as 12.53 m^2, based on the storage design. Each TES unit is divided into four vertical layers (nodes), consistent with the storage design, with the node index i starting at the top and increasing downward. This level of detail provides sufficient spatial resolution for modeling temperature gradients without introducing excessive computational cost. The terms involving k represent thermal interactions between neighboring nodes – so-called "pseudo-conduction" terms – which serve as an approximation for convective mixing. The assignment of signs and temperatures for vertical flows depends on the flow direction and is handled as follows:

$$T_{\text{vert,out}} = T_i \quad \text{if} \quad \dot{V}_{\text{vert,out}} > 0, \tag{6.12}$$

$$T_{\text{vert,in}} = T_{i-1} \quad \text{if} \quad \dot{V}_{\text{vert,in}} > 0, \tag{6.13}$$

$$T_{\text{vert,out}} = T_{i+1} \quad \text{if} \quad \dot{V}_{\text{vert,out}} < 0, \tag{6.14}$$

$$T_{\text{vert,in}} = T_i \quad \text{if} \quad \dot{V}_{\text{vert,in}} < 0. \tag{6.15}$$

This model can effectively capture key aspects of the TES performance, such as stratification, the dynamics of charging and discharging, and thermal losses.

6.1.3. Resulting Optimization Problem

This study addresses an optimization problem aimed at determining the optimal operation of the aforementioned energy utility system that balances both economic and environmental objectives. The problem is framed as a discrete-time, multi-period optimization over a finite planning horizon. The continuous time interval $[t_0, t_n]$ is divided into n equally spaced time steps, with each step having a duration $\Delta t = t_k - t_{k-1}$ for $k = 1, \ldots, n$. All decision variables are treated as piecewise constant over each interval and evaluated at the discrete time points t_k.

The optimization objective is to minimize a weighted sum of operating costs and CO_2 emissions. The operating costs result from electricity expenses due to grid power imports and are partially offset by revenues from excess PV electricity fed into the grid. CO_2 emissions are assumed proportional to the imported grid electricity and calculated using a fixed emission factor that accounts for the carbon intensity of certified green electricity, including associated infrastructure and supply emissions. These emissions are classified as Scope 2 (indirect emissions) according to World Resources Institute & World Business Council for Sustainable Development (2004). The complete objective function, consisting of both operating costs and CO_2 emissions, is expressed as:

$$\min \; f(\boldsymbol{P}_{\text{grid}}, \boldsymbol{P}_{\text{PV,sell}}) = w_{\text{cost}} \sum_{k=1}^{n} (P^k_{\text{grid}}\, g_{\text{pr,grid}} - P^k_{\text{PV,sell}}\, g_{\text{PV,sell}}) \Delta t + w_{\text{em}} \sum_{k=1}^{n} P^k_{\text{grid}}\, g_{\text{em,grid}}\, \Delta t. \tag{6.16}$$

Here, P^k_{grid} represents the imported grid power at time step k, $P^k_{\text{PV,sell}}$ denotes the PV power fed into the grid, and Δt is the duration of each time step. The grid electricity price $g_{\text{pr,grid}}$ is fixed at 0.17 €/kWh, the PV feed-in tariff $g_{\text{PV,sell}}$ at 0.16 €/kWh, and the emission factor $g_{\text{em,grid}}$ at 0.0557 kg/kWh, based on the project "SINNOGENES" (SINNOGENES, 2025). The scalar weights w_{cost} and w_{em} are each set to 0.5. However, this does not result in equal influence on the optimization, because the cost term has a greater numerical magnitude than the CO_2 term due to values of the grid electricity price, the PV feed-in tariff, and the emission factor. Nevertheless, the CO_2 contribution still plays a role in guiding the solution, in line with the decision made together with the company in Herzberg within the SINNOGENES project. In

future work, the effect of different weight values on the optimization results could be further investigated to explore the trade-offs between economic and environmental objectives. The optimization problem is subject to the following constraints:

- **Component models:** For each time step k, the STC, PV, HP, and TES components must operate according to their respective models introduced in Section 6.1.2. These models incorporate temperature-dependent water properties, which are vital for accurately simulating thermal processes (Wagner and Pruss, 2002). Specifically, the specific heat capacity is modeled as a third-order polynomial function of temperature, while the thermal conductivity and density follow second-order polynomial representations.

- **Flow and connectivity constraints:** At each time step k, mass and energy flows throughout the system are governed by equality constraints that enforce conservation laws at every interconnection node. These nodes represent points where several flow streams meet or split. The network topology, including optional bypass paths, is described in Section 6.1.1. The general forms of the conservation equations at a node l are given by:

$$\sum_{i \in \mathcal{I}_{l,\text{in}}} \dot{m}_{i,l}^k = \sum_{j \in \mathcal{I}_{l,\text{out}}} \dot{m}_{j,l}^k \quad \text{(mass conservation)}, \tag{6.17}$$

$$\sum_{i \in \mathcal{I}_{l,\text{in}}} \dot{m}_{i,l}^k h_{i,l}^k = \sum_{j \in \mathcal{I}_{l,\text{out}}} \dot{m}_{j,l}^k h_{j,l}^k + \dot{Q}_{l,\text{loss}}^k \quad \text{(energy conservation)}, \tag{6.18}$$

  where $i \in \mathcal{I}_{l,\text{in}}$ and $j \in \mathcal{I}_{l,\text{out}}$ denote the incoming and outgoing streams at node l, $\dot{m}_{i,l}^k$ and $\dot{m}_{j,l}^k$ are the corresponding mass flow rates, and $h_{i,l}^k$ and $h_{j,l}^k$ are the specific enthalpies of these streams. Thermal losses $\dot{Q}_{l,\text{loss}}$ are excluded from the interconnection nodes but included within individual component models such as the TES.

- **Thermal stratification in TESs:** To ensure realistic stratified temperature profiles in the TES units, a monotonicity condition is enforced. Specifically, the temperature of each layer must be lower than the temperature of any layer above it, i.e., $T_{i+1}^k < T_i^k$, where the layer index i increases from the top to the bottom of the TES. This constraint maintains a top-to-bottom temperature gradient, keeping the upper layers hotter than the lower ones.

- **Thermal integration of STC output:** The heat generated by the STC is stored in TES 1 and TES 2, directed to the layer whose temperature most closely matches the STC outlet. This minimizes mixing and improves the thermal efficiency. A penalty term is added to the objective function (6.16) to minimize the squared temperature deviation between the STC output and TES layer temperatures:

$$\min \sum_{k=1}^{n} \left(\sum_{l=1}^{N} \alpha_{1,l}^k (T_{\text{TES},1,l}^k - T_{\text{STC,out}}^k)^2 + \sum_{l=1}^{N} \alpha_{2,l}^k (T_{\text{TES},2,l}^k - T_{\text{STC,out}}^k)^2 \right). \tag{6.19}$$

Here, $\alpha_{i,l}^k \in [0,1]$ are continuous weights representing the fraction of STC output directed to layer l of TES i at time step k, with the constraint:

$$\sum_{l=1}^{N} \alpha_{i,l}^k = 1 \quad \text{for all } k, i \in \{1,2\}. \tag{6.20}$$

This smooth allocation mechanism eliminates the need for integer variables while effectively guiding the heat flow.

- **Heat demand constraint:** The heat demand required by the process must be met from the top layer of TES 3, subject to both temperature and flow rate requirements:

$$T_{\text{TES,3,out}}^k = T_{\text{demand}}^k, \quad \dot{m}_{\text{TES,3,out}}^k = \dot{m}_{\text{demand}}^k, \tag{6.21}$$

with $T_{\text{demand}}^k \in [88\,^\circ\text{C},\ 90\,^\circ\text{C}]$ and $\dot{m}_{\text{demand}}^k \in [3{,}000\,\text{l/h},\ 3{,}500\,\text{l/h}]$.

Additionally, the underlying optimization problem includes components with different time dynamics by employing two distinct time resolutions. PV, STC, and TES components are updated every 15 minutes to capture fast variations, while the HP operates on an hourly basis to limit frequent switching due to its lower operational flexibility. This is enforced by imposing equality constraints on the inlet and outlet temperatures as well as the inlet mass flow rate of the HP for intermediate 15-minute intervals:

$$T_{\text{c,in}}^k = T_{\text{c,in}}^{k-1}, \quad T_{\text{h,in}}^k = T_{\text{h,in}}^{k-1}, \quad T_{\text{h,out}}^k = T_{\text{h,out}}^{k-1}, \quad \dot{m}_{\text{h,in}}^k = \dot{m}_{\text{h,in}}^{k-1}, \quad \forall k \notin \{1,5,9,\dots\}. \tag{6.22}$$

The upper limit of the set $\{1,5,9,\dots\}$ depends on the total horizon length n of the optimization problem and thus adapts according to the particular use case setup. The resulting formulation is a nonlinear, nonconvex optimization problem with continuously differentiable constraints, avoiding binary or integer variables through relaxation (e.g., $\gamma_{\text{PV}} \in [0,1]$) and smooth approximations (e.g., continuous allocation weights $\alpha_{i,l}^k$ for distributing STC output to TES layers).

The problem requires several inputs: solar irradiance data for both the PV and STC system, electricity price, emission factor, heat demand, and ambient temperature. The electricity price and emission factor in (6.16) are constant throughout the study and known in advance for the upcoming hours. Similarly, the required temperature and mass flow rate for the thermal energy supply process are defined within fixed bounds (see (6.21)) and known for the entire planning period. Moreover, no forecasting model is developed for ambient temperature in this study, as reliable short-term forecasts are available from national meteorological platforms. Due to the high accuracy of these forecasts and the low interpolation error when converting hourly data to 15-minute intervals, additional modeling efforts are not necessary, as already mentioned in Chapter 3. Consequently, the only input that must be forecasted as part of the energy management strategy is the solar irradiance. Similar to wind conditions, publicly available solar irradiance forecasts are typically provided at hourly intervals, whereas the optimization model requires data at a finer

temporal resolution (15 minutes). However, solar irradiance can vary significantly over short time spans due to transient cloud cover and other localized weather effects. As a result, direct interpolation of hourly data may lead to substantial inaccuracies, making forecasting at a finer temporal resolution (15 minutes) essential for reliable energy management.

6.2. Solar Irradiance Forecasting

This section presents the methodology and results for solar irradiance forecasting using data-based models. It introduces the selected forecasting approaches, describes the underlying meteorological dataset, and evaluates the performance of the models using the MAE (introduced in Section 3.2) in both one-step- and multi-step-ahead prediction scenarios across two different seasons and locations in Germany.

6.2.1. Data-Based Models

For solar irradiance forecasting, this work employs a set of data-driven models due to their computational efficiency and strong short-term prediction capabilities, as already demonstrated for the wind speed prediction in Chapter 3. These include statistical methods – such as MLR (Voyant et al., 2017; Sehrawat et al., 2023), MPR (Koller et al., 2019), and ARIMA (Koller et al., 2019; Gallo et al., 2022) – which capture linear and basic nonlinear trends in time series data.

To address the limitations of these models in handling complex patterns, several ML and DL techniques are also used. These include KNN (Voyant et al., 2017), SVR (Voyant et al., 2017; Gallo et al., 2022; Sehrawat et al., 2023), and tree-based models such as HGBR (Sehrawat et al., 2023). DL methods, particularly hybrid architectures like CNN-LSTM (Gallo et al., 2022), further enhance the forecasting accuracy by leveraging both spatial and temporal features.

The selected models – MLR, MPR, ARIMA, KNN, SVR, HGBR, and CNN-LSTM – are widely used in solar irradiance forecasting for their strong predictive performance and have also been applied to wind speed forecasting, as shown in Chapter 3.

6.2.2. Data

This study uses solar irradiance data from the GWS platform, covering a 15-year period from 2010 to 2024, with a temporal resolution of 10 minutes. The dataset includes a comprehensive set of meteorological variables: historical solar irradiance, sunshine duration, time without sun, precipitation, precipitation duration, precipitation height, air temperature at two meters, air pressure at station level, air pressure gradients between two locations at station level, wind speed and wind direction at ten meters as well as the relative humidity at two meters above ground level. Additionally, a pseudo parameter is added as a numerical indicator of the calendar month (e.g., January = 1, February = 2, etc.).

Two geographically distinct locations in Germany – Hamburg in the north and Augsburg in the south – serve as the case studies. The forecasting models are trained and tested on data from the winter and summer seasons of 2024 to account for seasonal variability.

As it is common in environmental datasets, the GWS data contains missing or erroneous entries flagged with the placeholder value -999. These were preprocessed using monotonic cubic interpolation to fill gaps and ensure data consistency. This interpolation was also used to adjust the time resolution to 15-minute intervals, matching the requirements of the optimization framework discussed in Section 6.1.

To improve the forecasting accuracy, the most relevant input features were selected based on the p-value and AIC methods, which have been already introduced in Section 3.2. The final input set includes lagged values of the solar irradiance from the previous four time steps, as well as values from one day and one year ago. Additionally, sunshine duration, temperature, and air pressure from one day before are also included. This leads to the general forecasting model:

$$I_t = f(I_{t-1}, I_{t-2}, I_{t-3}, I_{t-4}, I_{t-\text{day}}, I_{t-\text{year}}, SD_{t-\text{day}}, T_{t-\text{day}}, p_{t-\text{day}}), \tag{6.23}$$

where I is the solar irradiance, SD denotes sunshine duration, T describes the temperature, and p is the air pressure. These predictors are applied across all models presented in Section 6.2.1, except for the ARIMA model, which relies solely on past irradiance values (see Section 3.1.3). The MPR model further enhances the predictive capacity by incorporating nonlinear interaction terms among the most relevant lagged variables. For the second-degree case, the expanded MPR formulation is given by:

$$\begin{aligned} I_t = f(&I_{t-1}, I_{t-3}, I_{t-4}, I_{t-\text{day}}, I_{t-\text{year}}, SD_{t-\text{day}}, T_{t-\text{day}}, p_{t-\text{day}}, I_{t-1}^2, I_{t-1}I_{t-3}, \\ &I_{t-1}I_{t-4}, I_{t-1}I_{t-\text{day}}, I_{t-1}I_{t-\text{year}}, I_{t-3}^2, I_{t-3}I_{t-4}, I_{t-3}I_{t-\text{day}}, I_{t-3}I_{t-\text{year}}, \\ &I_{t-4}I_{t-\text{day}}, I_{t-4}I_{t-\text{year}}, I_{t-\text{day}}^2, I_{t-\text{day}}I_{t-\text{year}}, I_{t-\text{year}}^2). \end{aligned} \tag{6.24}$$

Finally, the forecasting process checks whether it is day or night at each prediction. When it is night, based on sunrise and sunset times, the solar irradiance is set to zero to reflect the absence of solar input during these hours.

6.2.3. Results

This section evaluates the forecasting performance of the models introduced in Section 6.2.1. The ARIMA model used is configured as (0,2,2), a choice guided by patterns observed in the autocorrelation and partial autocorrelation plots (not shown in this thesis). This model indicates that the solar irradiance time series require second-order differencing ($d = 2$) to achieve stationarity, while two MA terms ($q = 2$) are sufficient and no significant AR component ($p = 0$) is present. For the ML and DL models, hyperparameter tuning utilizing RandomizedSearchCV from Sklearn in Python was performed to optimize the predictive performance. To prevent overfitting, a blocked CV approach (see Section 3.2)

was implemented. To further enhance model accuracy, feature scaling was applied where appropriate: normalization was used for KNN and CNN-LSTM, while SVR benefited from standardization. Sunrise and sunset times, considered in the forecasting models, were obtained using the ASTRAL package (Kennedy, 2024). This study focuses on the Rec strategy (see Section 3.3.2) for multi-step-ahead forecasting, as it is a widely used and well-established method in the literature. Moreover, as demonstrated in the wind speed forecasting results presented in Chapter 4, this strategy achieves a good balance between predictive accuracy and computational efficiency. All models were developed using Python 3.12, with the CNN-LSTM model, whose architecture is presented in Appendix A.2.1, utilizing the TENSORFLOW framework, ARIMA implemented via STATSMODELS, and the other ML models constructed using SCIKIT-LEARN. Computations were conducted on an Intel(R) Xeon(R) Gold 5220 CPU running at 2.20 GHz.

As expected, forecast errors tend to grow as the prediction horizon extends, a trend displayed in Fig. 6.2. For one-step-ahead predictions (marked as "1" on the x-axis), most models achieve similar accuracy, though CNN-LSTM and ARIMA exhibit slightly larger errors compared to the rest. However, as the forecast horizon length increases, ARIMA's performance degrades considerably: its MAE rises from 0.26 W/m^2 to 0.46 W/m^2 (about 79 % increase) during the winter week, and from 0.19 W/m^2 to 0.37 W/m^2 (approximately 91 % increase) in the summer week. This happens due to ARIMA's dependence on past error terms (its moving average component), which are unavailable for future steps and thus assumed to be zero – a simplification that weakens its effectiveness for multi-step forecasting. Consequently, ARIMA is best suited for short-term, single-step predictions but struggles with longer forecast horizons, as already shown in the wind speed predictions in Chapter 3. In contrast, models such as MLR, MPR, KNN, SVR, HGBR, and CNN-LSTM demonstrate more consistent error growth across multiple steps, maintaining relatively stable accuracy throughout the forecast period. On average, their MAE increases moderately – from 0.25 W/m^2 to 0.30 W/m^2 (around 20 % increase) in winter, and from 0.17 W/m^2 to

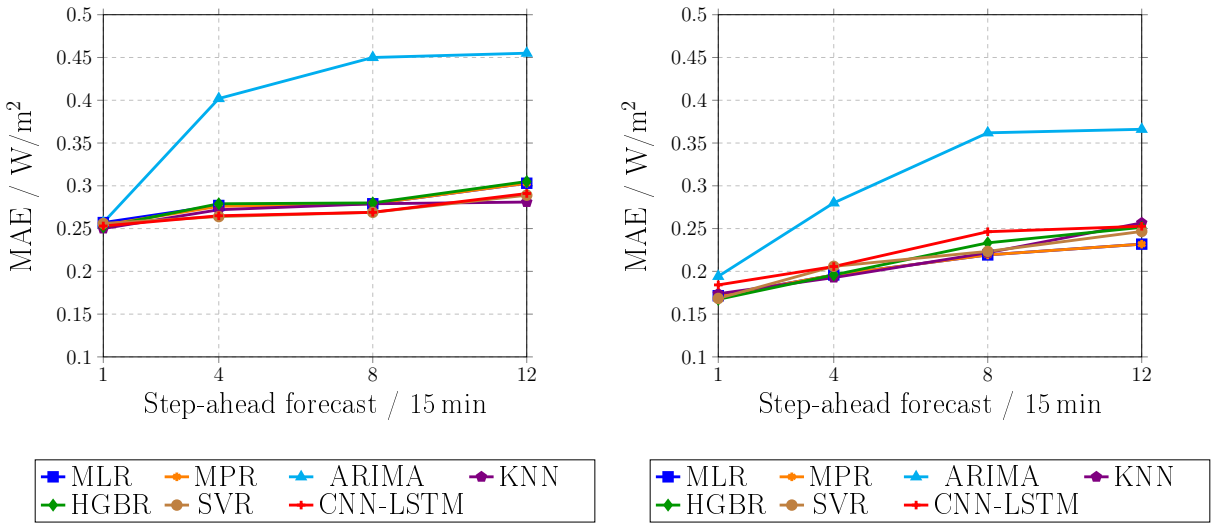

Figure 6.2: Results of solar irradiance forecast using MAE comparison for the tested methods with respect to different multi-step-ahead horizons, for Hamburg in winter (**left**) and Augsburg in summer 2024 (**right**).

0.25 W/m^2 (roughly 47 % increase) during summer. Notably, predictions tend to be more accurate in August (summer) compared to January (winter), with average errors decreasing by up to 30.8 % in the first forecasting step. This seasonal variation can be attributed to higher and more stable solar irradiance during summer months, which provides clearer signals for the models and reduces the influence of noise. In contrast, winter irradiance is generally lower and more variable, making reliable pattern detection more challenging. Another factor contributing to this performance difference is the change in location between Hamburg and Augsburg, as previously noted for wind speed (see Section 3.3.2).

Regarding computational demands, the training times vary widely among the models. ARIMA, SVR, and CNN-LSTM are more resource-intensive, with CNN-LSTM requiring approximately 711 seconds for training, while ARIMA and SVR take around 92 and 613 seconds respectively. Simpler models such as MLR, MPR, HGBR, and KNN train much faster, each completing in under 6 seconds – with MLR being the quickest at just 0.10 seconds. Overall, MLR, HGBR, and KNN offer practical options for solar irradiance forecasting due to their balanced trade-off between accuracy and computational efficiency. Future research could explore the incorporation of filtering techniques to further improve the performance of the models.

6.3. Energy Management

This section presents a detailed description of the methodology used for the energy management in the considered utility system, along with the corresponding results, incorporating the novel BO-IPOPT approach.

6.3.1. Methodology

The main goal of this study is to operate the industrial energy system outlined in Section 6.1 in a way that minimizes both electricity costs from the grid and CO_2 emissions, while accounting for the uncertainties arising from the fluctuating solar energy supply, including both PV and STC sources.

For this purpose, the tool "CoDeOpT" – originally introduced in Kansara and Roldán Serrano (2024) and extended in Kyriakidis et al. (2025a) – serves as the core platform for managing energy systems like the one examined in this study. As mentioned in Section 5.1 and illustrated in Fig. 6.3, CoDeOpT integrates forecasting, system modeling, optimization, and data exchange within a modular architecture tailored for industrial applications. This study introduces a key advancement compared to Section 5.1: the integration of CoDeOpT with the real plant through the "Sinnogenes Middleware" interface. Unlike the previous implementation that relied on OPC UA for communication with the real plant, this middleware was developed and adopted in the context of the European project "Sinnogenes" (Sinnogenes, 2025; Kyriakidis et al., 2025a). As displayed in Fig. 6.3, the middleware enables bidirectional data exchange – allowing CoDeOpT to send optimal setpoints to the plant and receive actual operating data at each iteration of the RHA, thereby enabling closed-loop and adaptive energy management. Although this study does not

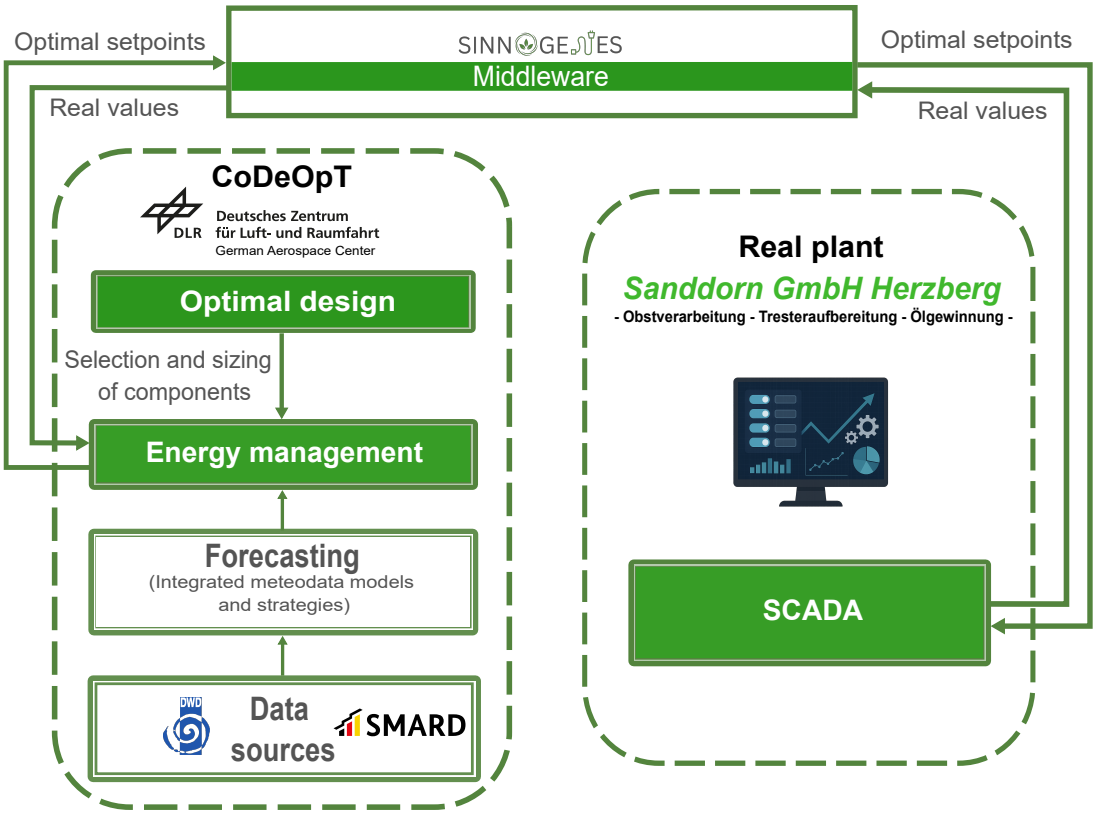

Figure 6.3: Architecture of CoDeOpT, illustrating its communication with the real plant through the interface "SINNOGENES Middleware" for exchanging real measurements and optimal setpoints.

leverage real-time measurements, the infrastructure is in place to support them. As an initial step, the workflow begins with an offline optimization step to determine the system's optimal design. This step has been performed within the scope of the European project "SINNOGENES" and serves as the basis for the models and the system introduced in Section 6.1, guided by technical requirements and engineering expertise.

In this study, the solar irradiance serves as the fluctuating input variable. Data is selected for two representative weeks – one in January and one in August – at a single location called "Herzberg (Elster)" in Germany, reflecting typical periods of low and high solar activity, respectively (see Fig. 6.4). The solar data is sourced from the GWS platform. The electricity price and the CO_2 emission factor for grid electricity are fixed and known in advance (see Section 6.1.3). As the original solar data is provided at a 10-minute resolution, it is resampled to match the 15-minute time steps required by the energy management framework using the monotonic cubic interpolation.

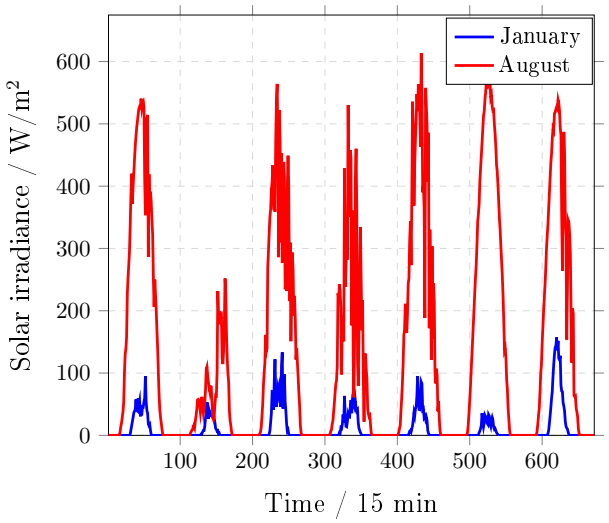

Figure 6.4: Visualization of one-week-scenario input data: solar irradiance for Herzberg (Elster), Germany in winter and summer 2024.

6.3.2. Results

As in Section 5.2, where the influence of various parameters on the system behavior was examined, we now investigate how these factors affect the performance of the energy management system in a real-world application. Specifically, we evaluate how the choice of the optimizer, the allowed CPU time per RHA iteration, the forecasting model for solar energy, and the optimization horizon length impact both operating costs and CO_2 emissions.

We begin by comparing the newly introduced hybrid optimization method, BO-IPOPT, with GA and PSO, IPOPT with single-start (with and without SVR), and the random-based MS-IPOPT. As in Chapter 5, linear programming solvers applied to linearized versions of the system are not considered here, since objective values from linearized models may systematically underestimate or overestimate the true system performance, making direct objective comparisons misleading. However, once real plant data become available, a fully fair comparison between linearized and nonlinear approaches will be possible. It is worth noticing that linearizing the optimization problem for the real-world system, as in the conceptual system, is particularly challenging due to the presence of bilinear and trilinear terms, which require additional auxiliary variables and constraints. All optimization algorithms in this study were implemented in Python 3.12, using the same configurations and parameter settings as described in Section 5.2. All experiments were conducted on a system equipped with an Intel(R) Core(TM) i7-8665U CPU. Due to the stochastic nature of all methods, each configuration was evaluated over 10 independent runs.

A default optimization horizon of 8 time steps (corresponding to 2 hours with a 15-minute resolution) was used, resulting in an optimization problem with 1,064 decision variables. This setting balances having a sufficient number of future steps for effective operational planning with maintaining forecast reliability (≤ 6 h ahead as mentioned in Chapter 3) and a computationally feasible problem size for the time limit

defined in this thesis (approximately 60 seconds per RHA iteration for the optimal setpoint determination). The effect of modifying the horizon length is examined later in this section. Since the global optimum is not known and applying a global deterministic solver (e.g., BARON) becomes computationally impractical for problems of this scale (over 100 decision variables), the best solution found among MS-IPOPT, BO-IPOPT, GA, and PSO with an extended 5-minute CPU time per RHA iteration is used as a reference for the performance comparisons.

The impact of different optimization methods on the system performance is assessed under idealized conditions, assuming perfect knowledge of future inputs. In agreement with the company located in Herzberg, a fixed CPU time of 50 seconds per RHA iteration is used, ensuring that the entire setpoint optimization at each RHA remains within the approximately 60-second computation window defined in this work. This time limit applies to BO-IPOPT, MS-IPOPT, GA, and PSO, which all rely on global or MS strategies and require the full CPU time to explore the search space effectively. In contrast, IPOPT and WS-IPOPT are initialized only once to perform a local optimization and typically converge well before reaching the 50-second limit. This value for the CPU time balances the need for real-time control signal exchange and system response with the high dimensionality of the underlying optimization problem and the associated challenge of solving it efficiently. Compared to the setting in Section 5.2, the chosen CPU time is longer, as the real-world use case analyzed here involves a significantly higher-dimensional optimization task, which requires more time to reach meaningful and feasible solutions. For the same reason, a shorter optimization horizon is adopted here to ensure real-time applicability and maintain computational efficiency.

Fig. 6.5 shows the total operating costs and CO_2 emissions over a representative winter and summer week. Each optimizer was tested across 10 trials, and the results are summarized in box plots. In all scenarios, BO-IPOPT consistently achieves the best median objective function value (up to 7 % lower than the IPOPT-based methods), while satisfying all problem constraints, and it exhibits the narrowest range of results – indicating both high accuracy and robustness in terms of operating costs and CO_2 emissions. During the winter week, BO-IPOPT achieves a median cost of 303.07 €, with a relative error of 2.8 % to the best known value of 294.95 €. MS-IPOPT follows with 309.63 € (5.0 % relative error), then WS-IPOPT at 310.94 € (5.4 % relative error), and IPOPT at 312.54 € (6.0 % relative error). In contrast, GA and PSO produce unrealistically low costs (0.27 € and 0.06 €), corresponding to relative errors of 99.9 % and 100.0 %, respectively, due to constraint violations. These solutions are invalid, with penalties embedded in the objective function reaching magnitudes of 10^{10} – highlighting the inability of GA and PSO to find feasible solutions within the limited time.

In the summer week, with increased solar irradiance enabling more effective use of the PV and STC systems, the best known cost is 182.68 €. BO-IPOPT achieves a median of 187.96 € (2.9 % relative error), followed by MS-IPOPT at 197.86 € (8.3 % relative error), WS-IPOPT at 199.06 € (9.0 % relative error), and IPOPT at 201.98 € (10.6 % relative error). GA and PSO again produce implausible results (0.08 € and 0.25 €), corresponding to relative errors of 100 % and 99.9 % due to infeasibility. It is worth noticing

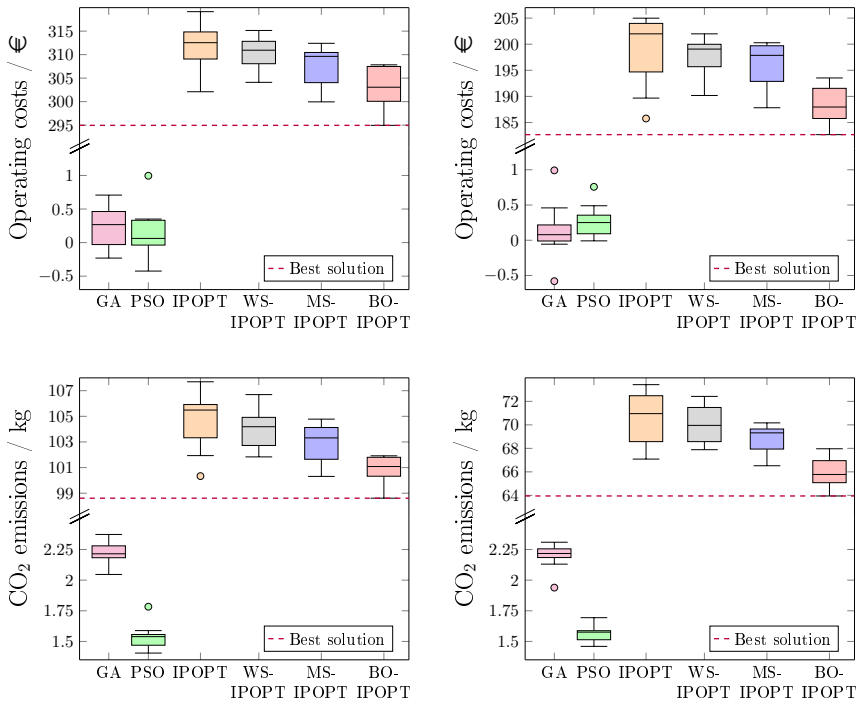

Figure 6.5: Comparison of operating costs (**top**) and CO_2 emissions (**bottom**) over 10 trials for different optimizers in winter (**left**) and summer (**right**) with an optimizer CPU time of 50 seconds per RHA iteration, a 8-step optimization horizon, and assuming that the input data is known. IPOPT and WS-IPOPT require only about six seconds on average at each RHA iteration. GA and PSO show lower operating costs and CO_2 emissions compared to the other optimizers due to constraint violations affecting the solution feasibility. The circles in the box plots represent outliers.

that the increased solar irradiance in the summer week increases variability and operational flexibility, making the optimization problem more challenging and amplifying the differences in the performance of the optimizers compared to the winter week.

CO_2 emissions follow the same pattern. In the winter week, the best known emissions are 98.61 kg. BO-IPOPT results on average in 101.08 kg (2.5 % relative error), MS-IPOPT in 103.32 kg (4.8 % relative error), WS-IPOPT in 104.19 kg (5.7 % relative error), and IPOPT in 105.49 kg (7.0 % relative error). GA and PSO return unrealistically low emission values (2.21 kg and 1.54 kg), corresponding to relative errors of 97.8 % and 98.4 %, respectively, again indicating constraint violations. In summer, BO-IPOPT achieves 65.78 kg (2.9 % relative error), MS-IPOPT 69.32 kg (8.4 % relative error), WS-IPOPT 69.96 kg (9.4 % relative error), and IPOPT 70.96 kg (11.0 % relative error) compared to the best known value of 63.95 kg. GA and PSO again return invalid emission values (2.22 kg and 1.58 kg), corresponding to relative errors of 96.5 % and 97.5 %, respectively.

The superior performance of BO-IPOPT is attributed to its hybrid structure: the BO component explores globally the search space to identify promising regions, while IPOPT ensures fast local convergence with strict constraint handling. As expected, a MS strategy (MS-IPOPT) improves the performance over single-start methods (IPOPT and WS-IPOPT), though it remains less effective than BO-IPOPT due to its

reliance on randomly sampled initial points. Notably, WS-IPOPT tends to produce a narrower range of outcomes than IPOPT, as already observed in the conceptual energy utility system in Chapter 5. This is because WS provides better initial guesses, reducing the optimizer's sensitivity to initialization and leading to more consistent results across trials, even if the median performance is only marginally improved. BO-IPOPT is the only optimizer that was able to consistently reach the best known solutions within the defined CPU time limits – further underscoring its effectiveness for real-time applications. In contrast, GA and PSO prove to be entirely ineffective due to their population-based, stochastic nature, which demands more iterations to converge, making their solutions unfeasible within the limited CPU time of the real-time framework.

Although IPOPT and WS-IPOPT are intended as single-start local optimizers, in practice they do not always converge on the first attempt and must be restarted. For IPOPT, on average, 21 % of all optimization steps in the winter week and 32 % in the summer week required at least one restart. Of these, 20 % and 9 %, respectively, failed more than once. WS-IPOPT performs more reliably: only 12 % of the steps needed a restart in winter and 19 % in summer, with 15 % and 7 % of the unsuccessful runs failing more than once. These results show again that WS improves the convergence robustness, but does not fully eliminate the risk of not converging – particularly under more challenging conditions. The notably higher restart frequency in the summer week again underscores the greater challenge for the optimizer when solar generation becomes more dominant, due to increased variability and operational flexibility. Overall, these findings emphasize the importance of hybrid strategies such as BO-IPOPT, which not only consistently achieve high-quality solutions within the predefined time limits but also provide greater reliability across varying operating conditions. Therefore, BO-IPOPT, the hybrid optimizer, is used exclusively in all subsequent evaluations in this section.

As demonstrated in Section 5.2 for the utility system for steam generation, keeping the optimizer's CPU time as short as possible is crucial in real-time applications to ensure that decisions can be made fast enough to respond effectively to dynamic system conditions. Although a 50-second limit was predefined together with the industrial partner for the present use case, deploying the optimization framework on the real system may require shorter computation times due to stricter control requirements. Therefore, in this section, we analyze how shorter CPU times per RHA iteration affect the performance of BO-IPOPT in terms of both operating costs and CO_2 emissions. Fig. 6.6 shows the optimization results for BO-IPOPT at 20, 30, 40, and 50 seconds CPU time per RHA iteration, evaluated under both winter (left) and summer (right) conditions. As expected, longer CPU times lead to an improved solution quality. In the winter scenario, the average operating costs drop by approximately 0.7 % and CO_2 emissions by about 1.0 % when increasing the time from 20 to 50 seconds. Under summer conditions, the improvements are more significant, with operating costs decreasing by 1.8 % and CO_2 emissions by 2.2 %.

While the average gains in objective values are modest, a more significant effect of the increased CPU time is the narrowing spread of the results, as reflected by the box plots. This reduction in variability, most evident at 50 seconds, demonstrates a marked improvement in the consistency and robustness of the

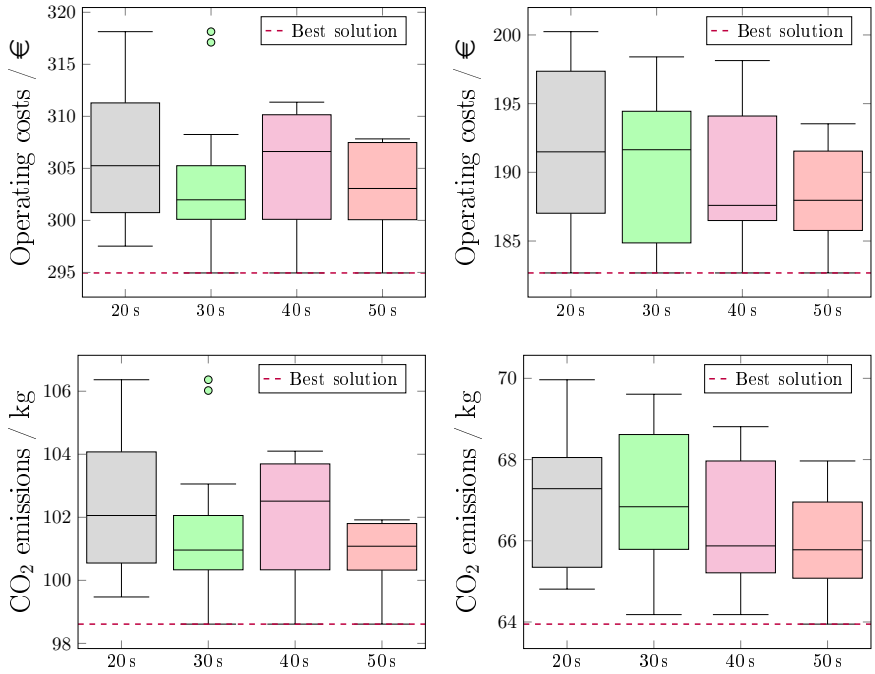

Figure 6.6: Comparison of operating costs (**top**) and CO_2 emissions (**bottom**) over 10 trials for different CPU times of BO-IPOPT in winter (**left**) and summer (**right**) with a 8-step optimization horizon and assuming that the input data is known.

optimization outcomes. Notably, even at shorter CPU times (20 and 30 seconds), the optimizer is able to identify near-optimal solutions, though with greater variability. These findings indicate that while 50 seconds per RHA iteration provides the most reliable and consistent results, shorter CPU times can still be applied if future control requirements necessitate faster RHA iterations. Looking ahead, the optimization framework will be deployed on a Linux-based server connected directly to the real plant. This setup is expected to allow the optimizer to achieve higher accuracy even at shorter CPU times (e.g., 20-40 seconds), or potentially improved performance at the same CPU time (50 seconds), facilitating faster and more robust real-time optimization.

As previously discussed, accounting for uncertainties in input data is essential for capturing realistic operating conditions. To address this, we integrate several solar irradiance forecasting models into the RHA and assess their impact on the system performance. The forecasting models – MLR, HGBR, and KNN – are implemented using a recursive forecasting strategy, selected for their favorable trade-off between prediction accuracy and computational cost, with training times of under 5 seconds (see Section 6.2.3). Additionally, we include a CNN-LSTM model, which, due to its significantly higher computational demands, is trained only once at the beginning of the energy management process rather than at each RHA iteration. This model is considered here because it demonstrated strong predictive performance in Section 6.2.3.

Fig. 6.7 shows the optimization results using a fixed CPU time of 50 seconds and an 8-step optimization horizon per RHA iteration. For both winter (left) and summer (right) conditions, introducing forecasted

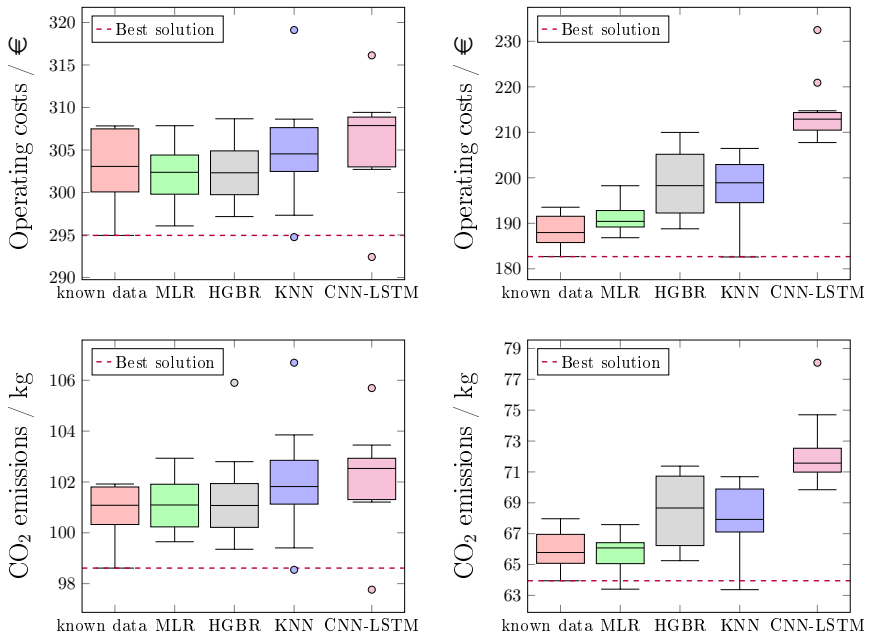

Figure 6.7: Comparison of operating costs (**top**) and CO_2 emissions (**bottom**) over 10 trials for different forecasting models for the solar data using BO-IPOPT with a CPU time of 50 seconds and a 8-step optimization horizon in winter (**left**) and summer (**right**).

solar data from the MLR, HGBR, and KNN models leads to increased operating costs and CO_2 emissions compared to the idealized case with perfect solar knowledge. On average, the highest deviations reach approximately 5.8 % in cost and 4.4 % in emissions. These increases are mainly caused by a systematic underestimation of the solar input, resulting in reduced solar utilization and higher reliance on grid electricity.

Among the tested models, MLR consistently achieves the closest performance to the perfect-data benchmark, with deviations remaining below 1 % across all scenarios, while also requiring minimal computational effort. In contrast, the CNN-LSTM model exhibits higher variability and average values (up to 13.3 % in costs and 8.8 % in emissions) due to its lack of retraining during RHA iterations and limited adaptability to changing conditions. Although periodic offline retraining of the CNN-LSTM model – performed in parallel with the energy management – could potentially preserve high forecasting accuracy, this aspect lies beyond the scope of the present work. These results highlight the effectiveness of simple models like MLR in real-time applications, where fast retraining and high prediction accuracy are essential. As in Section 5.2, MLR again proves to be a highly suitable choice and is therefore used in all subsequent analyses.

The final parameter analyzed in this study is the length of the optimization horizon, which significantly influences the operational strategy within the RHA-based energy management framework. As already mentioned in Section 5.2, the horizon length affects the performance in two key ways. First, longer horizons allow the optimizer to better account for future system conditions – such as fluctuations in renewable generation and electricity prices – resulting in more strategic operation, particularly in managing the

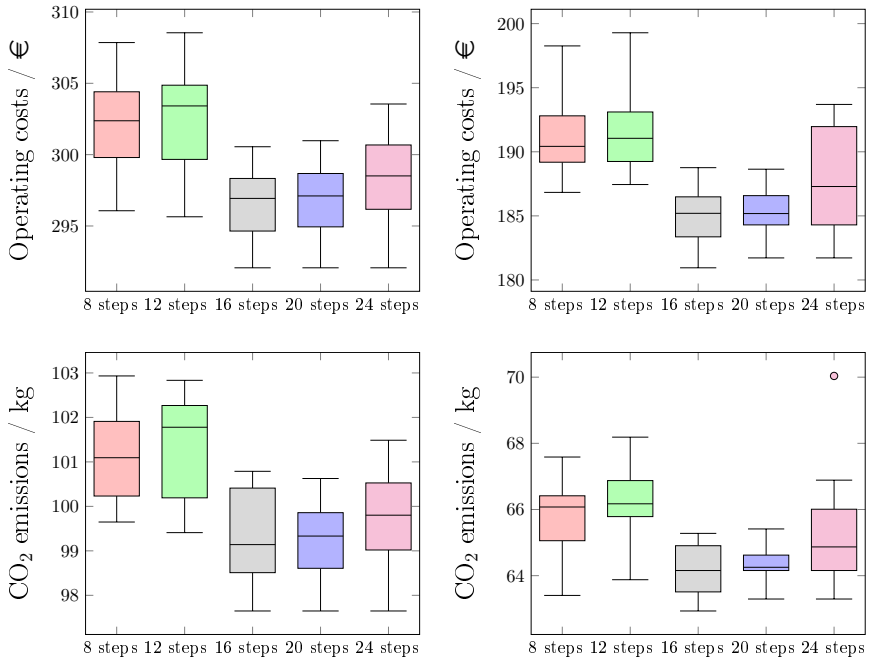

Figure 6.8: Comparison of operating costs (**top**) and CO_2 emissions (**bottom**) over 10 trials for different optimization horizons (in steps) using BO-IPOPT with a CPU time of 50 seconds and the MLR method for the solar data forecast in winter (**left**) and summer (**right**).

thermal energy storage. Second, increasing the horizon also raises the dimensionality of the optimization problem, making it more computationally demanding and more sensitive to forecast errors. Therefore, a careful balance between planning depth and computational feasibility is essential in practice. For the experiments conducted, we test horizons of 8, 12, 16, 20, and 24 steps, corresponding to 2, 3, 4, 5, and 6 hours at a 15-minute resolution, using a fixed computation time of 50 seconds per RHA iteration.

As shown in Fig. 6.8, extending the optimization horizon up to 20 steps leads to reductions of approximately 3 % in both operating costs and CO_2 emissions in winter and summer scenarios. This improvement is attributed to the optimizer's enhanced ability to anticipate future energy availability – particularly from PV and STC systems – and to make more forward-looking decisions. Additionally, the best solution achieved improves (i.e., the objective value decreases) with longer horizons. However, results for the 24-step case show greater variability, indicating increased problem dimensionality and suggesting that the fixed 50-second time limit is insufficient to consistently achieve high-quality solutions. Horizons beyond 24 steps were not evaluated due to the excessive computational demands.

To assess the impact of forecast uncertainty, Fig. 6.9 compares results obtained using forecasted solar data (via MLR) against those using perfect, known values. Even at the longest horizon of 24 steps, the deviations in operating cost and emissions due to forecast errors remain modest – about 1.2 % and 0.7 %, respectively. These differences are in line with those observed at shorter horizons (see Fig. 6.7), suggesting that, for the scenarios investigated, BO-IPOPT maintains strong robustness in the face of prediction

uncertainty. This also confirms that the MLR forecasting model provides sufficiently accurate inputs for reliable real-time optimization in the scenarios considered in this thesis.

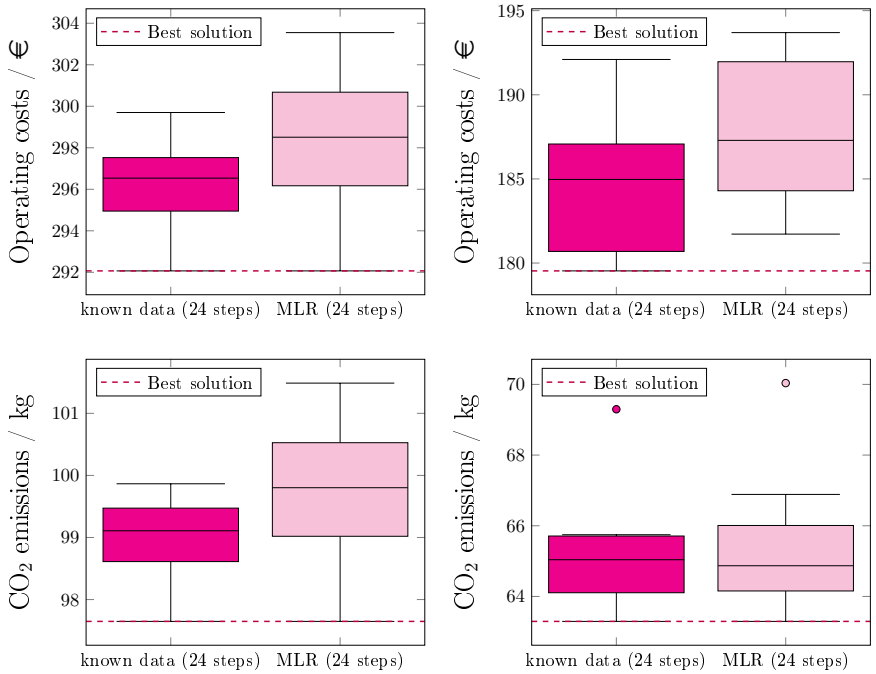

Figure 6.9: Comparison of operating costs (**top**) and CO_2 emissions (**bottom**) over 10 trials using BO-IPOPT with a CPU time of 50 seconds, a 24-step optimization horizon, and the MLR method for the solar data forecast in winter (**left**) and summer (**right**). The results compare the case without forecast uncertainties to the case with uncertainties.

6.4. Key Findings in Energy Management

After analyzing both the conceptual and real-world systems, several key findings emerged regarding the four central parameters of the effective energy management: the choice of optimization algorithm, forecasting model, optimization horizon length, and CPU time. Among the tested optimizers, BO-IPOPT consistently demonstrated superior performance in terms of objective function values compared to the others, even with limited CPU times of up to 50 seconds. This performance advantage became more pronounced as the system dimensionality increased – particularly with the integration of additional renewables and storage units, which introduced greater flexibility and problem size.

In contrast, IPOPT and WS-IPOPT frequently encountered convergence issues, often requiring restarts – sometimes multiple times within a single RHA iteration, which became increasingly common in the real-world problem. In addition to these convergence challenges, both IPOPT and WS-IPOPT exhibited a wide spread in the achieved objective values across several trials. While WS-IPOPT showed improved robustness over IPOPT by reducing both the frequency of convergence failures and the variability of outcomes, these issues were not fully resolved. This behavior suggests that MS strategies become more appropriate as the system complexity increases and the time available per RHA iteration is sufficiently

large (e.g., several seconds), offering greater robustness against local minima and convergence failures. However, BO-IPOPT outperformed the MS algorithm "MS-IPOPT" across all tested scenarios, achieving lower objective function values. This result highlights a key limitation of relying on randomly generated starting points in MS strategies, which often led to suboptimal solutions, particularly in the real-world case. Moreover, the stochastic solvers failed to find feasible solutions under tight time constraints, frequently violating constraints, with the magnitude of violations increasing alongside the system complexity.

Another important observation in energy management is that scenarios with greater amounts of wind or solar power tend to make the optimization problem more challenging due to the increased variability and flexibility in the system. Consequently, the use of a robust and efficient optimizer such as BO-IPOPT becomes increasingly important under these conditions.

In both the conceptual and real-world systems, forecasting accuracy in weather data played a crucial role in effective energy management. Among the tested models, MLR consistently achieved strong performance while requiring minimal computational effort. In both the solar and the more challenging wind forecasting scenarios, MLR resulted in operating cost deviations of less than 1 % compared to the ideal case with perfect knowledge. This makes it especially suitable for real-time applications where fast retraining is necessary. Other models such as HGBR and KNN also performed reasonably well, but they did not offer a significant advantage over MLR. The CNN-LSTM model, despite showing high predictive accuracy in isolated cases, lacked adaptability due to its high computational cost and the absence of retraining during the optimization process, resulting in reduced performance over time. Overall, MLR proved to be the most practical and robust choice for integrating short-term forecasting into the energy management, demonstrating reliable performance for both wind and solar data.

In addition to forecasting, the optimizer's CPU time and the length of the optimization horizon emerged as key factors for achieving effective and robust energy management. The results clearly indicate that as the complexity of the system increases, either longer CPU times must be allowed, or the optimization horizon must be shortened to maintain real-time feasibility. However, excessively short horizons limit the optimizer's ability to predict future system states, particularly in systems with energy storage, where longer horizons enable more strategic planning. On the other hand, very long horizons can increase the risk of forecasting errors accumulating, potentially degrading the optimization performance. Nonetheless, in the investigated cases, BO-IPOPT demonstrated high robustness against forecasting uncertainties, and no significant issues were observed even with extended horizon lengths. Additionally, excessively long CPU times are undesirable because the optimization must operate in real time, imposing strict upper limits on computational duration per iteration depending on each use case.

In the conceptual energy utility system, the optimization framework performed reliably with a fixed CPU time of 15 seconds per RHA iteration and horizon lengths of up to 48 steps (12 hours), with the 24-step horizon offering the best trade-off between predictive planning and computational feasibility. In the real-world case study, which involved a higher number of decision variables, a longer CPU time of 50 seconds per RHA iteration was required to ensure consistent solution quality. Under this setting,

optimization horizons of 16 and 20 steps (5 hours) achieved a good balance between planning depth and computational flexibility. Horizons beyond this point led to greater result variability, indicating that the fixed CPU time was insufficient for stable performance at higher problem sizes ($\geq$ 3,192 decision variables).

Overall, these findings highlight the importance of tuning horizon length and CPU time according to the system's complexity. The trade-offs observed here apply primarily to systems of similar complexity as those studied. For systems with increased complexity, these parameters may need to be adjusted to maintain a balance between predictive performance and computational feasibility. However, the effectiveness of the chosen configuration also depends on the available computational resources. Systems with higher processing power can execute more local searches in parallel and handle longer optimization horizons, enabling improved solution quality and extended planning depth at the same CPU time per RHA iteration. As future work, both the tuning process and the performance evaluation of BO-IPOPT relative to other optimization methods will be extended to systems of greater complexity, where additional challenges may arise due to increased dimensionality and flexibility – particularly in the presence of more diverse energy resources and storage units.

7. Conclusion & Future Work

This chapter summarizes the key findings of this work and discusses directions for future research. First, the main contributions and insights are outlined in the conclusion. Then, potential improvements are highlighted in the future work section.

7.1. Conclusion

This work addressed the growing need for efficient and sustainable energy management in industrial utility systems, which aims to reduce operating costs and CO_2 emissions while integrating variable renewable energy sources. To meet these challenges, a hybrid modeling and optimization framework was developed, combining simulation and experimental data for accurate component representation and integrating BAYESIAN optimization with IPOPT to efficiently solve complex nonlinear problems. The study was guided by two central RQs:

- **RQ 1**: How can the components of industrial energy utility systems be modeled realistically while adapting to new measurements in real time?
- **RQ 2**: How can complex industrial energy utility systems be optimized in real time with high accuracy?

Addressing RQ 1, three modeling approaches were introduced and systematically compared: (1) a transfer-learning-based ANN model, (2) a two-step weighted surrogate model, and (3) a unified surrogate model trained via a weighted cost function. The second approach was further analyzed with ANN and MPR surrogates to assess interpretability and training effort. Testing on two benchmark functions and a HTHP at pilot scale showed that the unified surrogate model consistently delivered the highest accuracy (up to 99.99 %) and robustness, with computational aspects discussed below. Its ability to adaptively weight simulation and experimental data based on data availability and noise characteristics in a unified surrogate model enables reliable performance even with limited or noisy datasets, making it particularly effective in high-dimensional and data-scarce scenarios. In contrast, ANN-based surrogates required significantly more data to achieve comparable accuracy, while the weighted surrogate with MPR worked well in many cases, but its performance degraded significantly when either the simulation-based or the experimental-based surrogate provided poor predictions.

Although the unified model showed increased CPU time in high-dimensional settings ($\geq$ 15 variables), where training exceeded 700 seconds due to the high problem nonlinearity and large amount of data,

this is acceptable when retraining is performed offline before or in parallel with real-time operation. In lower-dimensional problems (< 15 variables) and in the practical HTHP case, the model remained highly computationally efficient. In these cases, it required only 0.04–8 seconds depending on the data volume, enabling real-time retraining for energy management. Overall, these results establish the unified surrogate model as the most capable and generalizable approach among the tested methods, balancing predictive accuracy, robustness, scalability, interpretability, and computational effort.

Addressing RQ 2, this work developed BO-IPOPT, a hybrid optimization method designed to tackle nonlinear, nonconvex, and constrained problems, particularly in higher-dimensional settings ($\gtrsim$ 100 decision variables). The method combines the global search capability of BO with the fast local convergence of IPOPT, enabling more accurate and efficient solutions than state-of-the-art optimization solvers. Several methodological innovations were introduced to enhance the performance and scalability of this framework.

First, instead of performing continuous optimization of the acquisition function, BO-IPOPT evaluates EI on a discrete candidate set. This approach achieves a strong balance between computational cost and convergence speed while preserving the global exploration of BO and better addressing the scalability challenges of high-dimensional problems. Second, unlike traditional approaches that employ multiple surrogate models for the objective function and constraints, constraint handling was simplified in this work by introducing a single surrogate model within an AL formulation. This model efficiently captures both equality and inequality constraints without requiring user-defined hyperparameters, reducing computational effort while maintaining effective global guidance. Third, a first comprehensive analysis of BO-IPOPT's hyperparameters was carried out to enhance the global exploration of BO while eliminating the need for manual tuning of the parameters. Unlike conventional approaches that adjust these parameters dynamically during optimization – often increasing the computational costs – our method employs parameter values derived from prior sensitivity studies. This avoids the need for such adjustments and reduces computational overhead, resulting in a more efficient optimization process. Fourth, the GP surrogate was used primarily as a smooth guiding function rather than for detailed local modeling, showing comparable performance to a simpler linear surrogate and confirming its supporting rather than dominant role in the hybrid method. Finally, an adaptive trust-region mechanism was implemented, dynamically adjusting the search space around each iteration to focus IPOPT on promising regions without losing the exploratory power of BO, reducing highly suboptimal areas and enhancing the convergence speed of the hybrid method.

The effectiveness of BO-IPOPT was demonstrated on test problems as well as two conceptual case studies, each differing in problem size, nonconvexity, and nonlinearity. The hybrid optimizer was compared to advanced BO-based methods – including SCBO, SEGOKPLS, and SEGOKPLS+K – as well as the classical IPOPT-based approach "MS-IPOPT" and the hybrid BOwLS method. Particularly in high-dimensional problems, BO-IPOPT significantly outperformed these state-of-the-art approaches in terms of convergence rate and robustness. The hybrid optimizer achieved up to 120 % lower objective function values within the same CPU time, effectively mitigating the curse of dimensionality commonly

encountered in traditional BO approaches and demonstrating that it is a highly promising method for solving complex optimization problems.

In terms of computational efficiency, BO-IPOPT outperformed BOwLS primarily by eliminating the need for local searches during the initialization of the GP model. This not only reduces the computational overhead but also prevents the surrogate model from being overly restricted to initial local optima. Compared to MS-IPOPT, BO-IPOPT requires more computational effort as the number of samples increases, mainly due to the GP model being rebuilt at each outer iteration. However, across all test and conceptual problems, the CPU time grew approximately linearly with the number of iterations. Experiments with the DED problem, considering 5, 10, and 30 units, further confirmed that BO-IPOPT maintains a roughly linear increase in CPU time with the problem dimensionality. The computational cost of the hybrid method is dominated by the local optimization performed by IPOPT, which accounts for roughly 82–98 % of the total CPU time depending on the problem and number of iterations K. This far exceeds the extra overhead associated with rebuilding the surrogate model and shows BO-IPOPT's scalability and potential applicability to even larger optimization problems.

Finally, the hybrid framework offers flexibility to switch between local solvers, which is particularly advantageous since different algorithms – both commercial and open-source – perform better on different problem types.

In the final part of this work, the BO-IPOPT framework was applied to both a conceptual and a real-world energy management system using a RHA to evaluate its performance in a dynamic operational environment, where short reaction times and rapid adaptation are necessary. The implementation was done in the tool "CoDeOpT", which was enhanced as part of this thesis to support energy management. The conceptual system for RSG, consisting of a WT, HTHP, TES, SG, and grid electricity, was extended in this work to realistically capture operational conditions compared to its original presentation. Key enhancements include temperature-dependent heat capacities, thermal losses in the TES model, ramp constraints, and four OPs to maintain the required pressure levels throughout the thermal oil loop. Peak demand charges were also considered to penalize high electricity consumption during predefined intervals. In contrast to previous studies, which relied solely on simulation data for modeling the HTHP, the model here was developed using the hybrid approach combining simulation and experimental data, improving the accuracy and extrapolative generalization of the surrogate. The real-world energy utility system, introduced here for the first time, represents a case study of a food and cosmetics industry facility in Germany and includes a STC, PV, HP, grid electricity, and three stratified TES units. The optimization problems in both studies were formulated with a 15-minute time resolution, consistent with typical intraday trading. Components such as the HTHP, HP, and SG operated on an hourly basis to prevent rapid thermal changes that could negatively affect the system performance and equipment lifespan. The two studies aimed to assess the optimization method under realistic conditions, including variable renewable generation, high-dimensional decision spaces, and the need for real-time adaptation. Several key insights emerged

from this analysis regarding the choice of optimizers, forecasting models, optimization horizon length, and CPU time.

Across all scenarios, BO-IPOPT consistently delivered superior solution accuracy and robustness compared to state-of-the-art solvers, achieving up to 35 % lower objective function values at the same CPU time. Even under tight CPU limits of up to 50 seconds per RHA iteration, BO-IPOPT maintained high performance. These CPU limits align with the practical requirement defined at the beginning of this thesis (see Chapter 1), where a maximum CPU time of approximately 60 seconds per RHA iteration was allocated for determining an optimal setpoint to ensure sufficient time for control execution and feedback adaptation within each 15-minute time frame. The addition of renewable energy sources and storage units in the real-world case compared to the conceptual system introduced higher dimensionality and operational flexibility, further highlighting the method's ability to effectively manage complex, high-dimensional problems. In contrast, IPOPT and WS-IPOPT frequently faced convergence issues, often requiring multiple restarts within a single RHA iteration. While WS-IPOPT offered some improvement in robustness over IPOPT, both methods still showed considerable variability in the achieved objective values. MS-based strategies provided partial mitigation against local minima but were outperformed by BO-IPOPT in terms of solution quality and robustness in all cases, demonstrating the limitations of relying on randomly generated starting points in complex real-world settings. Moreover, stochastic solvers under limited CPU time violated operational constraints, with violations becoming more severe as the system's dimensionality increased. It is worth noticing that scenarios with higher shares of wind or solar generation increase the operational variability of the system, making the optimization task more challenging. This makes it more difficult for conventional solvers to find reliable solutions, emphasizing the importance of a robust and efficient optimizer like BO-IPOPT to maintain high-quality and consistent decision-making under these conditions.

While the optimizer defines the quality of the decisions, its performance is inherently linked to the accuracy of the input forecasts. Poor predictions can weaken even the most advanced algorithms, especially in systems driven by variable renewables. In this context, the forecasting accuracy proved to be a critical factor for effective energy management. Uncertainties were explicitly considered in wind and solar data, as their high variability makes accurate forecasting challenging, and forecast errors can directly impact the system operation. Electricity prices, although inherently variable in the conceptual system, were treated as deterministic in this work due to their many dependencies, which make accurate forecasting very complex. In the real-world system, the electricity price was considered fixed and fully known, eliminating the need for any forecasting. Similarly, heat demand and ambient temperature were treated as deterministic to avoid the need for additional forecasting models. Heat demand was modeled either as a constant value in the conceptual system or within a predefined range in the real-world case. Ambient temperature, which varies far less than wind or solar data, was represented as a known time series without uncertainty, as the minor errors of reliable short-term forecasts have only a negligible impact on operational decisions compared to fully deterministic cases. Among the forecasting models and strategies tested, the simple MLR with

the Rec strategy consistently delivered robust predictions with minimal computational overhead. This allowed the optimizer to integrate forecasts in real time, resulting in deviations in operating costs and CO_2 emissions of up to 1 % compared to the ideal case with perfect knowledge. Other models such as HGBR and KNN offered reasonable performance but did not substantially improve over MLR, while the CNN-LSTM model, despite achieving high accuracy in standalone benchmark tests, lacked adaptability due to high computational requirements and the inability to retrain at each RHA iteration in energy management. These findings underscore MLR as a practical and reliable choice for real-time forecasting of wind and solar data in energy management applications.

In addition to forecasting accuracy, the CPU time per RHA iteration and the length of the optimization horizon proved to be decisive factors for effective and reliable energy management. The study showed that, as systems become more complex, the optimizer needs either more time per RHA iteration or a shorter look-ahead period to keep calculations feasible in real time. If the horizon is too short, the optimizer can not anticipate future conditions, which is especially problematic in systems that rely on storage units, where strategic planning is essential. At the opposite extreme, very long horizons increase the risk that forecast errors will accumulate, reducing the quality of the results. Despite these challenges, BO-IPOPT showed strong resistance to forecasting uncertainties, delivering stable, high-quality solutions even when longer horizons were used. In addition, practical limits on CPU time must always be respected, since industrial systems require the optimizer to respond quickly and reliably.

In the conceptual case study, good performance was achieved with only 15 seconds of CPU time per RHA iteration, and horizons up to 48 steps (12 hours) could be used. A 24-step horizon provided the best balance, offering sufficient foresight to achieve lower operating costs than shorter horizons while avoiding the higher computational effort of longer ones. In contrast, the real-world case involved higher dimensionality and a larger set of decision variables. Here, a longer computation time of 50 seconds per iteration was necessary, and shorter horizons of 16–20 steps (4-5 hours) proved most effective. Extending the horizon further caused more variability in the solutions, indicating that the fixed computation time was insufficient to handle the larger problem size consistently.

Overall, these findings underscore the necessity of carefully tuning both the optimization horizon and CPU time per RHA iteration according to the system's complexity. While the specific trade-offs observed apply directly to the systems studied, larger or more complex systems may require further adjustments to maintain a balance between predictive accuracy and computational practicality. Furthermore, the effectiveness of this configuration is inherently linked to the available computational resources, as more powerful hardware can either support longer horizons or improve the solution quality within the same time window. Together with the unified surrogate modeling approach, BO-IPOPT forms a comprehensive framework that enables high-quality, real-time decision-making, offering a promising approach for high-dimensional and complex energy management systems.

7.2. Future Work

While this work has demonstrated the potential of the proposed modeling and optimization approach, there remain several avenues to further advance the two methodologies, improve their performance, and expand their practical impact. On the modeling side, future work could explore alternative optimization strategies for the unified cost function currently solved with L-BFGS-B. Approaches such as MS techniques or hybrid methods combining BO with L-BFGS-B could be investigated to potentially improve convergence and robustness. For higher-dimensional problems, parallelizing multiple local searches in the optimization method could be employed to reduce computational time and enhance exploration of the solution space. Since black-box models were not the focus of this work for building up surrogates, future research could explore alternative black-box methods, such as GP and RBF, to investigate whether they provide improvements in accuracy, robustness, scalability, and computational efficiency. Finally, the methodology should be applied to other real-world systems to further assess its extrapolative generalization and practical applicability.

On the optimization side, BO-IPOPT could be further refined and tested in a broader set of benchmark and real-world problems to ensure that the method remains effective under different levels of nonlinearity, nonconvexity, and dimensionality. This includes exploring advanced acquisition strategies such as LogEI (Ament et al., 2023) or compositional functions (Grosnit et al., 2021), which could be investigated to determine whether they accelerate the convergence in high-dimensional settings. Likewise, incorporating batch selection methods such as q-EI (Wang et al., 2020b) could help increase diversity in sampling, reduce the risk of redundant evaluations, and thereby support faster convergence. Although no convergence issues related to multiple candidates collapsing to the same minimum were observed in this study, q-EI may still bring benefits by further improving diversity and sampling efficiency. Additional investigation of hyperparameter settings in more diverse cases would help improve the performance and stability of BO-IPOPT. Future work could also explore ways to formally quantify the complexity of optimization problems, e.g., via metrics for nonlinearity, nonconvexity, and dimensionality, to better understand how problem characteristics affect the algorithm performance (Priesmann et al., 2019).

Another promising direction is to extend BO-IPOPT to handle mixed-integer optimization problems, as many real-world systems involve discrete decision variables, broadening the framework's relevance to an even wider range of industrial and operational contexts. Moreover, the evaluation with different local solvers (IPOPT, XPRESS, and CONOPT) in the hybrid method revealed that the solver effectiveness strongly depends on the underlying problem structure. This suggests that future research should systematically investigate problem-dependent solver performance and develop guidelines or adaptive strategies for selecting the most suitable local method within the hybrid framework. Finally, extending the framework to black-box optimization problems, where gradients are unavailable or expensive to compute, would be highly valuable, as such problems are common in scientific, engineering, and data-driven domains. A key challenge in this context is that IPOPT relies on gradient information, so considering efficient

strategies like the adjoint approach (Chen et al., 2020) to efficiently compute gradients would greatly enhance BO-IPOPT's flexibility and broaden its applicability to a wider range of real-world scenarios.

In energy management, future work should focus on improving robustness, applicability, and operational performance under more realistic conditions. In the conceptual energy utility system for RSG, a key next step is to investigate the impact of uncertainties in electricity prices, which are particularly relevant for applications such as day-ahead and intraday market trading. Extending the study to include additional uncertainties in variable heat demand and ambient temperature, would further test the framework's robustness. Filtering techniques like KALMAN filtering, moving average, or SAVITZKY-GOLAY filter (Durbin and Koopman, 2012) could be employed in both energy utility systems to reduce noise in input data and assess whether they improve the performance of the forecasting models.

For the real-world energy utility system, future work should aim at validating the framework in operational settings. Implementing BO-IPOPT in a Software- and Hardware-in-the-Loop setup or directly on the plant infrastructure would allow real-time transmission of setpoints to system components, with sensor feedback enabling dynamic model updating and evaluation of closed-loop behavior under realistic operating conditions. Such an implementation would provide critical insight into the practical feasibility, reliability, and effectiveness of the methodology in real industrial environments. In the case of the conceptual system where no physical plant exists, a Software-in-the-Loop setup can be used to validate real-time optimization, setpoint exchange, and closed-loop performance, whereas a Hardware-in-the-Loop setup is not feasible. For the individual components of the real-world system, the hybrid modeling approach developed in this work can be applied as soon as sufficient experimental data become available, further enhancing the model accuracy and reliability. Additionally, once real plant data are accessible, linearized models can also be implemented, and linear programming solvers can be compared with nonlinear programming methods by evaluating their performance on the same system, allowing a fair and meaningful assessment. Furthermore, ramp constraints based on measured component limitations could be added to enhance the operational realism. Moreover, future work could involve integrating time-varying electricity prices, allowing the optimizer to reduce operating costs and enhance operational flexibility. As part of future investigations, the influence of different scalar weight values in the objective function could be explored to better understand the trade-offs between economic and environmental objectives and to identify optimal weighting strategies under realistic system conditions.

Finally, while the current hybrid optimization approach scales well for the systems investigated, its performance in significantly larger or more complex systems remains to be evaluated. Future studies should therefore explore larger industrial configurations, higher-dimensional decision spaces, and broader uncertainty scenarios, including variations in demand profiles, renewable generation, and electricity prices, to assess extrapolative generalization and operational reliability.

Bibliography

Aasim, S. Singh, and A. Mohapatra. Repeated wavelet transform based ARIMA model for very short-term wind speed forecasting. *Renewable Energy*, 136:758–768, 2019. doi:https://doi.org/10.1016/j.renene.2019.01.031.

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. https://www.tensorflow.org/. Last checked: 20/11/2025.

A. Alkesaiberi, F. Harrou, and Y. Sun. Efficient Wind Power Prediction Using Machine Learning Methods: A Comparative Study. *Energies*, 15:1–24, 2022. doi:10.3390/en15072327.

S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 20577–20612, 2023. https://papers.nips.cc/paper_files/paper/2023/hash/419f72cbd568ad62183f8132a3605a2a-Abstract-Conference.html.

N. H. An and D. T. Anh. Comparison of Strategies for Multi-step-Ahead Prediction of Time Series Using Neural Network. In *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pages 142–149, 2015. doi:10.1109/ACOMP.2015.24.

N. Andrei. *Continuous Nonlinear Optimization for Engineering Applications in GAMS Technology*, volume 121 of *Springer Optimization and Its Applications*. Springer Cham, 2017. doi:10.1007/978-3-319-58356-3.

S. Ariafar, J. Coll-Font, D. Brooks, and J. Dy. ADMMBO: Bayesian Optimization with Unknown Constraints using ADMM. *Journal of Machine Learning Research*, 20:1–26, 2019. https://jmlr.org/papers/v20/18-227.html.

D. Arias, A. Gavilan, and R. Muren. PUMPING POWER PARASITICS IN PARABOLIC TROUGH SOLAR FIELDS. In *In Proceedings of the 15th International SolarPACES Symposium*, pages 1–8,

2009. https://www.researchgate.net/publication/303718533_PUMPING_POWER_PARASITICS_IN_PARABOLIC_TROUGH_SOLAR_FIELDS.

J. S. Arora, O. A. Elwakeil, A. I. Chahande, and C. C. Hsieh. Global optimization methods for engineering applications: A review. *Structural optimization*, 9:137–159, 1995. doi:https://doi.org/10.1007/BF01743964.

M. Asim, W. Mashwani, Ö. Yeniay, M. A. Jan, N. Tairan, H. Hussian, and G.-G. Wang. Hybrid Genetic Algorithms for Global Optimization Problems. *Hacettepe Journal of Mathematics and Statistics*, 47: 539–551, 2018. https://ieeexplore.ieee.org/document/485836.

P. Attaviriyanupap, H. Kita, E. Tanaka, and J. Hasegawa. A hybrid EP and SQP for dynamic economic dispatch with nonsmooth fuel cost function. *IEEE Transactions on Power Systems*, 17:411 – 416, 2002. doi:10.1109/TPWRS.2002.1007911.

S. Azadi and A. Karimi-Jashni. Verifying the performance of artificial neural network and multiple linear regression in predicting the mean seasonal municipal solid waste generation rate: A case study of Fars province, Iran. *Waste management*, 48:14–23, 2016. doi:https://doi.org/10.1016/j.wasman.2015.09.034.

R. Balamurugan and S. Subramanian. An Improved Differential Evolution Based Dynamic Economic Dispatch with Nonsmooth Fuel Cost Function. *Journal of Electrical Systems*, 3:151–161, 2007. https://www.ingentaconnect.com/content/doaj/11125209/2007/00000003/00000003/art00004?crawler=true.

M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, pages 1–34, 2020. http://arxiv.org/abs/1910.06403.

S. Barhmi, O. Elfatni, and I. Belhaj. Forecasting of wind speed using multiple linear regression and artificial neural networks. *Energy Systems*, 11:935–946, 2020. doi:https://doi.org/10.1007/s12667-019-00338-y.

N. Bartoli, T. Lefebvre, S. Dubreuil, R. Olivanti, R. Priem, N. Bons, J. Martins, and J. Morlier. Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. *Aerospace Science and Technology*, 90:85–102, 2019. doi:https://doi.org/10.1016/j.ast.2019.03.041.

F. Bauelos-Ruedas, C. Angeles-Camacho, and Sebastin. Methodologies Used in the Extrapolation of Wind Speed Data at Different Heights and Its Impact in the Wind Energy Resource Assessment in a Region. In G. O. Suvire, editor, *Wind Farm - Technical Regulations, Potential Estimation and Siting Assessment*. InTech, 2011. doi:10.5772/20669.

P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–119, 2013. doi:10.1017/S0962492913000032.

J. Berk, V. Nguyen, S. Gupta, S. Rana, and S. Venkatesh. Exploration Enhanced Expected Improvement for Bayesian Optimization. In M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 621–637, 2019. doi:https://doi.org/10.1007/978-3-030-10928-8_37.

D. Berrar. Cross-Validation. In S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 542–545. Academic Press, Oxford, 2019. doi:https://doi.org/10.1016/B978-0-12-809633-8.20349-X.

D. Bertsimas and M. Sim. The Price of Robustness. *Operations Research*, 52:35–53, 2004. doi:10.1287/opre.1030.0065.

M. Binois, D. Ginsbourger, and O. Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization*, 76:69–90, 2020. doi:10.1007/s10898-019-00839-1.

K. Bio Gassi and M. Baysal. Analysis of a linear programming-based decision-making model for microgrid energy management systems with renewable sources. *International Journal of Energy Research*, 46: 7495–7518, 2022. doi:https://doi.org/10.1002/er.7656.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Science & Business Media, New York, NY, 2011. doi:https://doi.org/10.1007/978-1-4614-0237-4.

A. Bischi, L. Taccari, E. Martelli, E. Amaldi, G. Manzolini, P. Silva, S. Campanari, and E. Macchi. A detailed MILP optimization model for combined cooling, heat and power system operation planning. *Energy*, 74:12–26, 2014. doi:https://doi.org/10.1016/j.energy.2014.02.042.

A. Bischi, L. Taccari, E. Martelli, E. Amaldi, G. Manzolini, P. Silva, S. Campanari, and E. Macchi. A rolling-horizon optimization algorithm for the long term operational scheduling of cogeneration systems. *Energy*, 184:73–90, 2019. doi:https://doi.org/10.1016/j.energy.2017.12.022.

M. A. Bouhlel, N. Bartoli, R. G. Regis, A. Otsmane, and J. Morlier. Efficient global optimization for high-dimensional constrained problems by using the kriging models combined with the partial least squares method. *Engineering Optimization*, 50:2038–2053, 2018. doi:https://doi.org/10.1080/0305215X.2017.1419344.

G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden Day, San Francisco, 1970. Revised Edition.

G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, 5th edition, 1994. ISBN 978-1-118-67502-1. https://www.wiley.com/en-us/Time+Series+Analysis%3A+Forecasting+and+Control%2C+5th+Edition-p-9781118675021.

L. Breiman. Random Forests. *Machine Learning*, 45:5–32, 2001. doi:https://doi.org/10.1023/A:1010933404324.

E. Brochu, V. M. Cora, and N. de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. 2010. http://arxiv.org/abs/1012.2599.

N. Cadau, A. De Lorenzi, A. Gambarotta, M. Morini, and M. Rossi. Development and Analysis of a Multi-Node Dynamic Model for the Simulation of Stratified Thermal Energy Storage. *Energies*, 12: 1–22, 2019. doi:10.3390/en12224275.

Canadian Solar. HiKu5 Mono Technical Information Sheet, 2022. https://cdn.enfsolar.com/z/pp/acc60dbda3e4ed35/5fe2bced1c3d5.pdf. Last checked: 20/11/2025.

Z. Cao, L. Wang, and X. Hei. A global-best guided phase based optimization algorithm for scalable optimization problems and its application. *Journal of Computational Science*, 25:38–49, 2018. doi:https://doi.org/10.1016/j.jocs.2018.02.001.

D. Carvalho, A. Rocha, M. Gómez-Gesteira, and C. Santos. A sensitivity study of the WRF model in wind simulation for an area of high wind energy. *Environmental Modelling & Software*, 33:23–34, 2012. doi:https://doi.org/10.1016/j.envsoft.2012.01.019.

R. Chelouah and P. Siarry. Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, 148: 335–348, 2003. doi:https://doi.org/10.1016/S0377-2217(02)00401-0. Sport and Computers.

L. Chen, H. Qiu, L. Gao, C. Jiang, and Z. Yang. Optimization of expensive black-box problems via Gradient-enhanced Kriging. *Computer Methods in Applied Mechanics and Engineering*, 362:1–21, 2020. doi:https://doi.org/10.1016/j.cma.2020.112861.

R. Cheng and Y. Jin. A Competitive Swarm Optimizer for Large Scale Optimization. *IEEE Transactions on Cybernetics*, 45:191–204, 2015. doi:10.1109/TCYB.2014.2322602.

I. Cherki, A. Chaker, Z. Djidar, N. Khalfallah, and F. Benzergua. A Sequential Hybridization of Genetic Algorithm and Particle Swarm Optimization for the Optimal Reactive Power Flow. *Sustainability*, 11: 1–12, 2019. doi:https://doi.org/10.3390/su11143862.

S. S. Chong, A. Abdul Aziz, S. W. Harun, H. Arof, and S. Shamshirband. Application of multiple linear regression, central composite design, and ANFIS models in dye concentration measurement and prediction using plastic optical fiber sensor. *Measurement*, 74:78–86, 2015. doi:https://doi.org/10.1016/j.measurement.2015.06.019.

C. Corinaldesi, D. Schwabeneder, G. Lettner, and H. Auer. A rolling horizon approach for real-time trading and portfolio optimization of end-user flexibilities. *Sustainable Energy, Grids and Networks*, 24: 1–10, 2020. doi:https://doi.org/10.1016/j.segan.2020.100392.

C. Cortes and V. N. Vapnik. Support-Vector Networks. *Machine Learning*, 20:273–297, 2004. doi:http://dx.doi.org/10.1007/BF00994018.

A. Cozad, N. V. Sahinidis, and D. C. Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60:2211–2227, 2014. doi:https://doi.org/10.1002/aic.14418.

Deutscher Wetterdienst. Climate data center, 2025. https://opendata.dwd.de/climate_environment/CDC/. Last checked: 20/11/2025.

T. G. Dietterich. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000. doi:https://doi.org/10.1007/3-540-45014-9_1.

DLR Solar Research. Greenius Manual, 2024. https://www.dlr.de/de/sf/forschung-und-transfer/forschungsdienstleistungen/simulation-und-wirtschaftlichkeitsbewertung/copy_of_greenius/greenius-support. Last checked: 20/11/2025.

A. J. Domingo, F. Carlo Garcia, M. L. Salvaña, N. J. C. Libatique, and G. L. Tangonan. Short Term Wind Speed Forecasting: A Machine Learning Based Predictive Analytics. In *TENCON 2018 - 2018 IEEE Region 10 Conference*, pages 1948–1953, 2018. doi:10.1109/TENCON.2018.8650287.

A. W. Dowling, R. Kumar, and V. M. Zavala. A multi-scale optimization framework for electricity market participation. *Applied Energy*, 190:147–164, 2017. doi:https://doi.org/10.1016/j.apenergy.2016.12.081.

G. V. Drisya, P. Valsaraj, K. Asokan, and K. S. Kumar. Wind speed forecast using random forest learning method. *International Journal on Computer Science and Engineering (IJCSE)*, 9:362–367, 2017. doi:https://doi.org/10.48550/arXiv.2203.14909.

A. S. Drud. CONOPT: A Large-Scale GRG Code. *ORSA Journal on Computing*, 6:207–216, 1994. doi:https://doi.org/10.1287/ijoc.6.2.207.

J. Durbin and S. J. Koopman. Filtering, Smoothing and Forecasting. In *Time Series Analysis by State Space Methods*, Oxford Statistical Science Series. Oxford University Press, 2nd edition, 2012. https://doi.org/10.1093/acprof:oso/9780199641178.003.0004.

Eastman Chemical Company. Therminol VP-1 Technical bulletin, 2025. https://www.eastman.com/Literature_Center/T/TF9141.pdf. Last checked: 20/11/2025.

Energynest. Prozesswärme mit thermischen Speichern dekarbonisieren, 2025. https://energy-nest.com/de/. Last checked: 20/11/2025.

F. G. Erdinç. Rolling horizon optimization based real-time energy management of a residential neighborhood considering PV and ESS usage fairness. *Applied Energy*, 344:1–13, 2023. doi:https://doi.org/10.1016/j.apenergy.2023.121275.

D. Eriksson and M. Poloczek. Scalable Constrained Bayesian Optimization. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 730–738, 2021. https://proceedings.mlr.press/v130/eriksson21a.html.

D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local Bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 1–12, 2019. https://proceedings.neurips.cc/paper/2019/hash/6c990b7aca7bc7058f5e98ea909e924b-Abstract.html.

Fair Isaac Corporation. FICO Xpress Optimization, 2025. https://www.fico.com/en/products/fico-xpress-optimization. Last checked: 20/11/2025.

R. Floors and M. Nielsen. Estimating Air Density Using Observations and Re-Analysis Outputs for Wind Energy Purposes. *Energies*, 12:1–12, 2019. doi:10.3390/en12112038.

P. S. Foundation. Multiprocessing module. https://docs.python.org/3/library/multiprocessing.html. Last checked: 20/11/2025, 2023.

Free OPC UA Contributors. Python OPC-UA, 2023. https://python-opcua.readthedocs.io/en/latest/. Last checked: 20/11/2025.

F. N. Fritsch and R. E. Carlson. Monotone Piecewise Cubic Interpolation. *SIAM Journal on Numerical Analysis*, 17:238–246, 1980. doi:https://doi.org/10.1137/0905021.

R. Gallo, M. Castangia, A. Macii, E. Macii, E. Patti, and A. Aliberti. Solar radiation forecasting with deep learning techniques integrating geostationary satellite images. *Engineering Applications of Artificial Intelligence*, 116:1–16, 2022. doi:https://doi.org/10.1016/j.engappai.2022.105493.

GAMS. Python API, 2025. https://www.gams.com/latest/docs/API_PY_OVERVIEW.html. Last checked: 20/11/2025.

GAMS Development Corporation. General Algebraic Modeling System (GAMS), 35 Distribution, 2025. https://www.gams.com/download/. Last checked: 20/11/2025.

Y. Gao, T. Yu, and J. Li. Bayesian Optimization with Local Search. In G. Nicosia, V. Ojha, E. La Malfa, G. Jansen, V. Sciacca, P. Pardalos, G. Giuffrida, and R. Umeton, editors, *Machine Learning, Optimization, and Data Science*, pages 350–361, 2020. doi:https://doi.org/10.1007/978-3-030-64580-9_30.

J. R. Gardner, M. J. Kusner, X. Zhixiang, K. Q. Weinberger, and J. P. Cunningham. Bayesian optimization with inequality constraints. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *PMLR*, pages 937–945, 2014. https://proceedings.mlr.press/v32/gardner14.html.

M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian Optimization with Unknown Constraints. In *Conference on Uncertainty in Artificial Intelligence*, page 250–259, 2014. https://dl.acm.org/doi/10.5555/3020751.3020778.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006. doi:10.1007/s10994-006-6226-1.

E. Ghysels and H. Qian. Estimating MIDAS regressions via OLS with polynomial parameter profiling. *Econometrics and Statistics*, 9:1–16, 2019. doi:https://doi.org/10.1016/j.ecosta.2018.02.001.

R. B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020. http://bobby.gramacy.com/surrogates/.

R. B. Gramacy, G. A. Gray, S. Le Digabel, H. K. H. Lee, P. Ranjan, G. Wells, and S. M. Wild. Modeling an Augmented Lagrangian for Improved Blackbox Constrained Optimization. *Technometrics*, 58, 2016. doi:10.1080/00401706.2015.1014065.

S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh. Bayesian Optimization for Adaptive Experimental Design: A Review. *IEEE Access*, 8:13937–13948, 2020. doi:10.1109/ACCESS.2020.2966228.

A. Grosnit, A. I. Cowen-Rivers, R. Tutunov, R.-R. Griffiths, J. Wang, and H. Bou-Ammar. Are we forgetting about compositional optimisers in Bayesian optimisation? *Journal of Machine Learning Research*, 22:1–78, 2021. https://dl.acm.org/doi/abs/10.5555/3546258.3546418.

J. Hao, W. Ye, L. Jia, G. Wang, and J. Allen. Building surrogate models for engineering problems by integrating limited simulation data and monotonic engineering knowledge. *Advanced Engineering Informatics*, 49:1–15, 2021. doi:https://doi.org/10.1016/j.aei.2021.101342.

W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3:219–260, 2011. doi:https://doi.org/10.1007/s12532-011-0026-8.

C. He, R. Cheng, Y. Tian, X. Zhang, K. C. Tan, and Y. Jin. Paired Offspring Generation for Constrained Large-Scale Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 25: 448–462, 2021. doi:10.1109/TEVC.2020.3047835.

D. He, G. Dong, F. Wang, and Z. Mao. Optimization of dynamic economic dispatch with valve-point effect using chaotic sequence based differential evolution algorithms. *Energy Conversion and Management*, 52:1026–1032, 2011. doi:https://doi.org/10.1016/j.enconman.2010.08.031.

Z. He, Y. Chen, and Y. Zang. Wind Speed Forecasting Based on Phase Space Reconstruction and a Novel Optimization Algorithm. *Sustainability*, 16:1–29, 2024. doi:10.3390/su16166945.

A. Hebbal, M. Balesdent, L. Brevault, N. Melab, and E.-G. Talbi. Deep Gaussian process for multi-objective Bayesian optimization. *Optimization and Engineering*, page 1809–1848, 2023. doi:https://doi.org/10.1007/s11081-022-09753-0.

J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani. Predictive Entropy Search for Bayesian Optimization with Unknown Constraints. In *International Conference on Machine Learning*, page 1699–1707, 2015. https://jmhl.org/wp-content/uploads/2015/05/pesc-icml2015.pdf.

H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, I. Rozum, D. Schepers, A. Simmons, C. Soci, D. Dee, and J.-N. Thépaut. ERA5 hourly data on single levels from 1940 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS), 2023. DOI: https://doi.org/10.24381/cds.adbb2d47. Last checked: 20/11/2025.

P. Hietaharju, M. Ruusunen, and K. Leiviskä. Enabling Demand Side Management: Heat Demand Forecasting at City Level. *Materials*, 12:1–17, 2019. doi:10.3390/ma12020202.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. doi:https://doi.org/10.7551/mitpress/1090.001.0001.

J. Hu, J. Wang, and G. Zeng. A hybrid forecasting approach applied to wind speed time series. *Renewable Energy*, 60:185–194, 2013. doi:http://dx.doi.org/10.1016/j.renene.2013.05.012.

N. Hu. The Limitations of Traditional PID Controllers and Modern Optimization Methods. *Applied and Computational Engineering*, 147:238–244, 2025. doi:10.54254/2755-2721/2025.22912.

K. D. Humbird, J. L. Peterson, J. Salmonson, and B. K. Spears. Cognitive simulation models for inertial confinement fusion: Combining simulation and experimental data. *Physics of Plasmas*, 28:1–8, 2021. doi:10.1063/5.0041907.

H. Imran, N. M. Al-Abdaly, M. H. Shamsa, A. Shatnawi, M. Ibrahim, and K. A. Ostrowski. Development of Prediction Model to Predict the Compressive Strength of Eco-Friendly Concrete Using Multivariate Polynomial Regression Combined with Stepwise Method. *Materials*, 15, 2022. doi:10.3390/ma15010317.

M. Z. Jacobson and V. Jadhav. World estimates of PV optimal tilt angles and ratios of sunlight incident upon tilted and tracked PV panels relative to horizontal panels. *Solar Energy*, 169:55–66, 2018. doi:https://doi.org/10.1016/j.solener.2018.04.030.

S. Jeong and S. Obayashi. Efficient global optimization (EGO) for multi-objective problem and data mining. In *IEEE Congress on Evolutionary Computation*, volume 3, pages 2138–2145, 2005. doi:10.1109/CEC.2005.1554959.

J. Jeßberger, C. Arpagaus, F. Heberle, L. Brendel, S. Bertsch, and D. Brüggemann. Experimental investigations of upscaling effects of high-temperature heat pumps with R1233zd(E). *International Journal of Refrigeration*, 164:243–256, 2024. doi:https://doi.org/10.1016/j.ijrefrig.2024.04.023.

P. Jiang, Q. Shanshan, J. Wu, and B. Sun. Time Series Analysis and Forecasting for Wind Speeds Using Support Vector Regression Coupled with Artificial Intelligent Algorithms. *Mathematical Problems in Engineering*, 2015:1–14, 2015. doi:10.1155/2015/939305.

Y. Jiang and G. Huang. Short-term wind speed prediction: Hybrid of ensemble empirical mode decomposition, feature selection and error correction. *Energy Conversion and Management*, 144: 340–350, 2017. doi:https://doi.org/10.1016/j.enconman.2017.04.064.

D. Kahaner, C. Moler, and S. Nash. *Numerical methods and software*. Prentice-Hall, Inc., USA, 1989. ISBN 0136272584. https://dl.acm.org/doi/abs/10.5555/61926.

T. Kallabis. Rolling-Horizon Optimization As a Speed-Up Method - Assessment Using the Electricity System Model JMM. Technical report, HEMF Working Paper No. 06/2020, 2020. Available at SSRN: https://ssrn.com/abstract=3684538.

R. Kansara and M. I. Roldán Serrano. Coupled Design and Operation Optimization for Decarbonization of Industrial Energy Systems Using an Open-Source In-House Tool. *Eng*, 5:3033–3048, 2024. doi:10.3390/eng5040158.

I. J. Karassik, J. P. MessinaPaul, and C. C. Heald. *Pump Handbook*. McGraw-Hill, 4th edition, 2007. ISBN 9780071460446. https://www.accessengineeringlibrary.com/content/book/9780071460446.

V. Kelner, F. Capitanescu, O. Léonard, and L. Wehenkel. A hybrid optimization technique coupling an evolutionary and a local search algorithm. *Journal of Computational and Applied Mathematics*, 215: 448–456, 2008. doi:https://doi.org/10.1016/j.cam.2006.03.048.

S. Kennedy. Astral: Python calculations for the position of the sun and moon, 2024. https://github.com/sffjunkie/astral. Last checked: 20/11/2025.

M. Khalid. Smart grids and renewable energy systems: Perspectives and grid integration challenges. *Energy Strategy Reviews*, 51:1–26, 2024. doi:https://doi.org/10.1016/j.esr.2024.101299.

D. L. King, S. Gonzalez, G. M. Galbraith, and W. E. Boyson. Performance Model for Grid-Connected Photovoltaic Inverters. Technical Report SAND2007-5036,

Sandia National Laboratories, 2007. https://energy.sandia.gov/wp-content/gallery/uploads/Performance-Model-for-Grid-Connected-Photovoltaic-Inverters.pdf.

M. Koller, J. Feldmaier, and K. Diepold. A Comparison of Prediction Algorithms and Nexting for Short Term Weather Forecasts. *CoRR*, abs/1903.07512, 2019. doi:https://doi.org/10.48550/arXiv.1903.07512.

O. Kramer. *K-Nearest Neighbors*, pages 13–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-38652-7_2.

L. Kyriakidis and M. Bähr. ENERGY MANAGEMENT OF AN INDUSTRIAL POWER-TO-HEAT SYSTEM USING A HYBRID OPTIMIZATION ALGORITHM. In *Proceedings of ECOS 2025: 38th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems (ECOS2025)*, 2025. Accepted, https://www.scopus.com/pages/publications/105037454568.

L. Kyriakidis and A. Thapa. Hybrid Approach Based on Simulation and Experimental Data for Surrogate-Based Modeling: A Case Study of a High-Temperature Heat Pump. In *Proceedings of ASME 2025: International Mechanical Engineering Congress and Exposition (IMECE2025)*, pages 1–10, 2025. doi:https://doi.org/10.1115/IMECE2025-167746.

L. Kyriakidis, M. A. Mendez, and M. Bähr. A hybrid algorithm based on Bayesian optimization and Interior Point OPTimizer for optimal operation of energy conversion systems. *Energy*, 312:1–10, 2024. doi:https://doi.org/10.1016/j.energy.2024.133416.

L. Kyriakidis, R. Kansara, and M. I. Roldán Serrano. Energy Management of Industrial Energy Systems via Rolling Horizon and Hybrid Optimization: A Real-Plant Application in Germany. *Energies*, 18: 1–29, 2025a. doi:10.3390/en18153977.

L. Kyriakidis, B. Martin, and M. A. Mendez. Enhanced Hybrid Algorithm based on Bayesian Optimization and Interior Point OPTimizer for Constrained Optimization. *Optimization and Engineering*, pages 1–52, 2025b. doi:https://doi.org/10.1007/s11081-025-09975-y.

V. Laitsos, G. Vontzos, D. Bargiotas, A. Daskalopulu, and L. H. Tsoukalas. Data-Driven Techniques for Short-Term Electricity Price Forecasting through Novel Deep Learning Approaches with Attention Mechanisms. *Energies*, 17(7):1–27, 2024. doi:10.3390/en17071625.

R. Lam and K. Willcox. Lookahead Bayesian Optimization with Inequality Constraints. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, page 1888–1898, 2017. https://proceedings.neurips.cc/paper/2017/hash/83f97f4825290be4cb794ec6a234595f-Abstract.html.

R. Lam, D. Allaire, and K. Willcox. Multifidelity Optimization using Statistical Surrogate Modeling for Non-Hierarchical Information Sources. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pages 1–21, 2015. doi:10.2514/6.2015-0143.

G. Lan, J. M. Tomczak, D. M. Roijers, and A. Eiben. Time efficiency in optimization with a bayesian-Evolutionary algorithm. *Swarm and Evolutionary Computation*, 69:1–14, 2022. doi:https://doi.org/10.1016/j.swevo.2021.100970.

J. Larson, M. Menickelly, and S. M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28: 287–404, 2019. doi:10.1017/S0962492919000060.

L. Lasdon and J. C. Plummer. Multistart algorithms for seeking feasibility. *Computers and Operations Research*, 35:1379–1393, 2008. doi:https://doi.org/10.1016/j.cor.2006.08.008.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1:541–551, 1989. doi:10.1162/neco.1989.1.4.541.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998. doi:10.1109/5.726791.

B. Letham, B. Karrer, G. Ottoni, and E. Bakshy. Constrained Bayesian Optimization with Noisy Experiments. *Bayesian Analysis*, 14(2):495–519, 2019. doi:10.1214/18-BA1110.

S. Li, J. Yang, W. Song, and A. Chen. A Real-Time Electricity Scheduling for Residential Home Energy Management. *IEEE Internet of Things Journal*, 6:2602–2611, 2019. doi:10.1109/JIOT.2018.2872463.

T. Li, M. Hua, and X. Wu. A Hybrid Cnn-Lstm Model for Forecasting Particulate Matter (PM2.5). *IEEE Access*, 8:26933–26940, 2020. doi:10.1109/ACCESS.2020.2971348.

J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. C. Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, IEEE Congress on Evolutionary Computation (CEC), 2006. http://al-roomi.org/benchmarks/cec-database/cec-2006.

Q. Long and C. Wu. A hybrid method combining genetic algorithm and Hooke-Jeeves method for constrained global optimization. *Journal of Industrial and Management Optimization*, 10:1279–1296, 2014. doi:10.3934/jimo.2014.10.1279.

C. Lu and J. A. Paulson. No-Regret Bayesian Optimization with Unknown Equality and Inequality Constraints using Exact Penalty Functions. *IFAC-PapersOnLine*, 55:895–902, 2022. doi:https://doi.org/10.1016/j.ifacol.2022.07.558.

H. Lund, S. Werner, R. Wiltshire, S. Svendsen, J. E. Thorsen, F. Hvelplund, and B. V. Mathiesen. 4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems. *Energy*, 68:1–11, 2014. doi:https://doi.org/10.1016/j.energy.2014.02.089.

J. Luo, Z. Chen, and Y. Zheng. A gradient-based method assisted by surrogate model for robust optimization of turbomachinery blades. *Chinese Journal of Aeronautics*, 35:1–7, 2022. doi:https://doi.org/10.1016/j.cja.2021.07.019.

H. Lv, X. Chen, and X. Zeng. Optimization of micromixer with cantor fractal baffle based on simulated annealing algorithm. *Chaos, Solitons and Fractals*, 148:1–9, 2021. doi:https://doi.org/10.1016/j.chaos.2021.111048.

T. Ma, J. Wu, and L. Hao. Energy flow modeling and optimal operation analysis of the micro energy grid based on energy hub. *Energy Conversion and Management*, 133:292–306, 2017. doi:https://doi.org/10.1016/j.enconman.2016.12.011.

M. Malu, G. Dasarathy, and A. Spanias. Bayesian Optimization in High-Dimensional Spaces: A Brief Survey. In *2021 12th International Conference on IISA*, pages 1–8, 2021. doi:10.1109/IISA52424.2021.9555522.

A. Marina, S. Spoelstra, H. Zondag, and A. Wemmers. An estimation of the European industrial heat pump market potential. *Renewable and Sustainable Energy Reviews*, 139:1–14, 2021. doi:https://doi.org/10.1016/j.rser.2020.110545.

R. Martí, J. A. Lozano, A. Mendiburu, and L. Hernando. Multi-start Methods. In R. Martí, P. M. Pardalos, and M. G. C. Resende, editors, *Handbook of Heuristics*, pages 155–175, 2018. doi:10.1007/978-3-319-07124-4_1.

R. Martí, R. Aceves, M. T. León, J. M. Moreno-Vega, and A. Duarte. Intelligent Multi-Start Methods. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 221–243, 2019. doi:https://doi.org/10.1007/978-3-319-91086-4_7.

G. Mavrotas, K. Florios, and D. Vlachou. Energy planning of a hospital using Mathematical Programming and Monte Carlo simulation for dealing with uncertainty in the economic parameters. *Energy Conversion and Management*, 51:722–731, 2010. doi:https://doi.org/10.1016/j.enconman.2009.10.029.

K. McBride and K. Sundmacher. Overview of Surrogate Modeling in Chemical Process Engineering. *Chemie Ingenieur Technik*, 91:228–239, 2019. doi:https://doi.org/10.1002/cite.201800091.

F. Ming, W. Gong, D. Li, L. Wang, and L. Gao. A Competitive and Cooperative Swarm Optimizer for Constrained Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 27:1313–1326, 2023. doi:10.1109/TEVC.2022.3199775.

Modelica. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, 2025. https://modelica.org. Last checked: 20/11/2025.

C. Moler. *Numerical Computing with MATLAB*. Society for Industrial and Applied Mathematics, 2004. ISBN 978-0-89871-660-3. doi:https://doi.org/10.1137/1.9780898717952.

M. A. Mquqwana and S. Krishnamurthy. Particle Swarm Optimization for an Optimal Hybrid Renewable Energy Microgrid System under Uncertainty. *Energies*, 17:1–21, 2024. doi:10.3390/en17020422.

P. A. Murtaugh. In defense of P values. *Ecology*, 95:611–617, 2014. doi:https://doi.org/10.1890/13-0590.1.

Mustafa Qamaruddin. Cross-Validation Strategies for Time Series Forecasting, 2019. https://medium.com/sci-net/cross-validation-strategies-for-time-series-forecasting-9e6cfab91f60. Last checked: 20/11/2025.

N. Müller and L. G. Frechette. Performance Analysis of Brayton and Rankine Cycle Microsystems for Portable Power Generation. In *ASME International Mechanical Engineering Congress and Exposition*, pages 513–522, 2002. doi:10.1115/IMECE2002-39628.

A. Nayebi, A. Munteanu, and M. Poloczek. A Framework for Bayesian Optimization in Embedded Subspaces. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4752–4761, 2019. https://proceedings.mlr.press/v97/nayebi19a.html.

A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinko. A Comparison of Complete Global Optimization Solvers. *Mathematical Programming*, 103:335–356, 2005. doi:10.1007/s10107-005-0585-4.

T. T. Ngo, A. Sadollah, and J. H. Kim. A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems. *Journal of Computational Science*, 13: 68–82, 2016. doi:https://doi.org/10.1016/j.jocs.2016.01.004.

J. Nocedal and S. J. Wright. *Numerical Optimization*, volume 2 of *Springer Series in Operations Research and Financial Engineering*. Springer New York, NY, 2006. doi:https://doi.org/10.1007/978-0-387-40065-5.

L. Olatomiwa, S. Mekhilef, M. Ismail, and M. Moghavvemi. Energy management strategies in hybrid renewable energy systems: A review. *Renewable and Sustainable Energy Reviews*, 62:821–835, 2016. doi:https://doi.org/10.1016/j.rser.2016.05.040.

D. E. Olivares, J. D. Lara, C. A. Cañizares, and M. Kazerani. Stochastic-Predictive Energy Management System for Isolated Microgrids. *IEEE Transactions on Smart Grid*, 6:2681–2693, 2015. doi:10.1109/TSG.2015.2469631.

S. Pan, J. Jian, and L. Yang. A hybrid MILP and IPM approach for dynamic economic dispatch with valve-point effects. *International Journal of Electrical Power & Energy Systems*, 97:290–298, 2018. doi:https://doi.org/10.1016/j.ijepes.2017.11.004.

S. Pan, J. Jian, H. Chen, and L. Yang. A full mixed-integer linear programming formulation for economic dispatch with valve-point effects, transmission loss and prohibited operating zones. *Electric Power Systems Research*, 180, 2020. doi:https://doi.org/10.1016/j.epsr.2019.106061.

B. K. Panigrahi, Y. Shi, and M.-H. Lim, editors. *Handbook of Swarm Intelligence: Concepts, Principles and Applications*, volume 8. Springer, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-17390-5.

A. Parisio, E. Rikos, and L. Glielmo. A Model Predictive Control Approach to Microgrid Operation Optimization. *IEEE Transactions on Control Systems Technology*, 22:1813–1827, 2014. doi:10.1109/TCST.2013.2295737.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011. https://jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

J. Pelamatti, R. L. Riche, C. Helbert, and C. Blanchet-Scalliet. Coupling and selecting constraints in Bayesian optimization under uncertainties. *Optimization and Engineering*, 25:373–412, 2024. doi:https://doi.org/10.1007/s11081-023-09807-x.

V. Picheny. A Stepwise uncertainty reduction approach to constrained global optimization. In *17th International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 787–795, 2014. https://proceedings.mlr.press/v33/picheny14.html.

V. Picheny, R. B. Gramacy, S. Wild, and S. L. Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *Proceedings of the 30th International Conference on NIPS*, page 1443–1451, 2016. https://papers.nips.cc/paper_files/paper/2016/hash/31839b036f63806cba3f47b93af8ccb5-Abstract.html.

H. Pieper, T. Ommen, J. Jensen, B. Elmegaard, and W. Markussen. Comparison of COP estimation methods for large-scale heat pumps in energy planning tools. In *Proceedings of ECOS 2019: 32nd International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems*, pages 1–19, 2019. https://orbit.dtu.dk/en/publications/comparison-of-cop-estimation-methods-for-large-scale-heat-pumps-i.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007. https://assets.cambridge.org/97805218/80688/frontmatter/9780521880688_frontmatter.pdf.

R. Priem, N. Bartoli, and Y. Diouane. On the Use of Upper Trust Bounds in Constrained Bayesian Optimization Infill Criteria. *AIAA Aviation 2019 Forum*, pages 1–10, 2019. doi:https://doi.org/10.2514/6.2019-2986.

R. Priem, N. Bartoli, Y. Diouane, and A. Sgueglia. Upper trust bound feasibility criterion for mixed constrained Bayesian optimization with application to aircraft design. *Aerospace Science and Technology*, 105:105980, 2020. doi:https://doi.org/10.1016/j.ast.2020.105980.

R. Priem, N. Bartoli, Y. Diouane, S. Dubreuil, and P. Saves. High-dimensional efficient global optimization using both random and supervised embeddings. In *AIAA AVIATION 2023 Forum*, pages 1–19, 2023. doi:https://doi.org/10.2514/6.2023-4448.

J. Priesmann, L. Nolting, and A. Praktiknjo. Are complex energy system models more accurate? An intra-model comparison of power system optimization models. *Applied Energy*, 255:1–16, 2019. doi:https://doi.org/10.1016/j.apenergy.2019.113783.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. 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, 2019. doi:https://doi.org/10.1016/j.jcp.2018.10.045.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005. doi:https://doi.org/10.7551/mitpress/3206.001.0001.

V. Ravikumar Pandi and B. K. Panigrahi. Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm. *Expert Systems with Applications*, 38:8509–8514, 2011. doi:https://doi.org/10.1016/j.eswa.2011.01.050.

R. G. Regis and C. A. Shoemaker. A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions. *INFORMS Journal on Computing*, 19:497–509, 2007. doi:10.1287/ijoc.1060.0182.

A. P. Roberts, A. A. M. Rahat, D. S. Jarman, J. E. Fieldsend, and G. R. Tabor. Multi-objective Bayesian shape optimization of an industrial hydrodynamic separator using unsteady Eulerian-Lagrangian simulations. *Optimization and Engineering*, pages 1–32, 2024. doi:https://doi.org/10.1007/s11081-024-09907-2.

H. Rodriguez, M. Medrano, L. M. Rosales, G. P. Peñuñuri, and J. J. Flores. Multi-step forecasting strategies for wind speed time series. In *2020 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, volume 4, pages 1–6, 2020. doi:10.1109/ROPEC50909.2020.9258743.

Saiken Pumps. Thermal oil pumps, 2025. https://www.saikenpump.com/High-Temperature-Circulation-Thermal-Oil-Centrifugal-Pump.html. Last checked: 20/11/2025.

M. Sakawa and K. Yauchi. Floating Point Genetic Algorithms for Nonconvex Nonlinear Programming Problems: Revised GENOCOP III. *Electronics and Communications in Japan Part 3*, 83:1–9, 2000. doi:https://doi.org/10.1002/(SICI)1520-6440(200008)83:8<1::AID-ECJC1>3.0.CO;2-Y.

F. Samanipour and J. Jelovica. Adaptive repair method for constraint handling in multi-objective genetic algorithm based on relationship between constraints and variables. *Applied Soft Computing*, 90:1–15, 2020. doi:https://doi.org/10.1016/j.asoc.2020.106143.

J. K. Sankaran. A note on resolving infeasibility in linear programs by constraint relaxation. *Operations Research Letters*, 13(1):19–20, 1993. doi:https://doi.org/10.1016/0167-6377(93)90079-V.

M. Santhosh, C. Venkaiah, and D. V. Kumar. Short-term wind speed forecasting approach using Ensemble Empirical Mode Decomposition and Deep Boltzmann Machine. *Sustainable Energy, Grids and Networks*, 19:1–16, 2019. doi:https://doi.org/10.1016/j.segan.2019.100242.

D. Santra, A. Mukherjee, K. Sarker, and S. K. Mondal. Dynamic economic dispatch using hybrid metaheuristics. *Journal of Electrical Systems and Information Technology*, 7:1–30, 2020. doi:https://doi.org/10.1186/s43067-020-0011-2.

K. Sareen, B. K. Panigrahi, T. Shikhola, and R. Sharma. An imputation and decomposition algorithms based integrated approach with bidirectional LSTM neural network for wind speed prediction. *Energy*, 278:1–26, 2023. doi:https://doi.org/10.1016/j.energy.2023.127799.

A. P. Sari, H. Suzuki, T. Kitajima, T. Yasuno, D. A. Prasetya, and N. Nachrowie. Prediction Model of Wind Speed and Direction using Convolutional Neural Network - Long Short Term Memory. In *2020 IEEE International Conference on Power and Energy (PECon)*, pages 356–361, 2020. doi:10.1109/PECon48942.2020.9314474.

M. J. Sasena, P. Y. Papalambros, and P. Goovaerts. The use of surrogate modeling algorithms to exploit disparities in function computation time within simulation-based optimization. In G. D. Cheng, Y. Gu, S. Liu, and Y. Wang, editors, *The Fourth World Congress of Structural and Mutlidisciplinary Optimisation*, pages 1–6, 2001. https://www.researchgate.net/publication/2367848_The_Use_of_Surrogate_Modeling_Algorithms_to_Exploit_Disparities_in_Function_Computation_Time_within_Simulation-Based_Optimization.

P. Saves, E. Nguyen Van, N. Bartoli, T. Lefebvre, C. David, S. Defoort, Y. Diouane, and J. Morlier. Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design. In *AIAA SCITECH 2022 Forum*, pages 1–22, 2023. doi:10.2514/6.2022-0082.

P. Saves, R. Lafage, N. Bartoli, Y. Diouane, J. Bussemaker, T. Lefebvre, J. T. Hwang, J. Morlier, and J. R. R. A. Martins. SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes. *Advances in Engineering Sofware*, 188:1–16, 2024. doi:https://doi.org/10.1016/j.advengsoft.2023.103571.

N. Sehrawat, S. Vashisht, and A. Singh. Solar irradiance forecasting models using machine learning techniques and digital twin: A case study with comparison. *International Journal of Intelligent Networks*, 4:90–102, 2023. doi:https://doi.org/10.1016/j.ijin.2023.04.001.

V. Shanmugarjah, N. W. A. Lidula, and A. D. Rajapakse. A Comparative Analysis of Rule-Based and Optimization-Based Energy Management for a Campus Microgrid in Sri Lanka. In *2024 International Conference on Sustainable Energy: Energy Transition and Net-Zero Climate Future (ICUE)*, pages 1–6, 2024. doi:10.1109/ICUE63019.2024.10795646.

J. Silvente, G. M. Kopanos, E. N. Pistikopoulos, and A. Espuña. A rolling horizon optimization framework for the simultaneous energy supply and demand planning in microgrids. *Applied Energy*, 155:485–501, 2015. doi:https://doi.org/10.1016/j.apenergy.2015.05.090.

N. Singh and S. R. Mohanty. A Review of Price Forecasting Problem and Techniques in Deregulated Electricity Markets. *Journal of Power and Energy Engineering*, 3:1–19, 2015. doi:10.4236/jpee.2015.39001.

S. Sivasubramani and K. Swarup. Hybrid SOA–SQP algorithm for dynamic economic dispatch with valve-point effects. *Energy*, 35:5031–5036, 2010. doi:https://doi.org/10.1016/j.energy.2010.08.018.

Skipper Seabold and Josef Perktold. Statsmodels: Econometric and Statistical Modeling with Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 92 – 96, 2010. doi:10.25080/Majora-92bf1922-011.

SMARD. Marktdaten, 2025. https://www.smard.de/home/downloadcenter/download-marktdaten/?downloadAttributes=%7B%22selectedCategory%22:3,%22selectedSubCategory%22:false,%22selectedRegion%22:false,%22selectedFileType%22:false%7D. Last checked: 20/11/2025.

Solar Keymark Database. Annex to Solar Keymark Certificate, 2025. https://www.duurzaamloket.nl/DBF/PDF_Downloads/DS_305.pdf. Last checked: 20/11/2025.

R. M. Solgi. geneticalgorithm 1.0.2, 2020. https://pypi.org/project/geneticalgorithm/. Last checked: 20/11/2025.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. https://jmlr.org/papers/v15/srivastava14a.html.

STEAG System Technologies. Ebsilon®professional, 2025. https://help.ebsilon.com/EN/EBSILON_Professional_Documentation.html. Last checked: 20/11/2025.

Z. Su, J. Wang, H. Lu, and G. Zhao. A new hybrid model optimized by an intelligent optimization algorithm for wind speed forecasting. *Energy Conversion and Management*, 85:443–452, 2014. doi:https://doi.org/10.1016/j.enconman.2014.05.058.

T. Tao, P. Shi, H. Wang, L. Yuan, and S. Wang. Performance Evaluation of Linear and Nonlinear Models for Short-Term Forecasting of Tropical-Storm Winds. *Applied Sciences*, 11:1–17, 2021. doi:10.3390/app11209441.

M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*, volume 65 of *Nonconvex Optimization and Its Applications*. Springer New York, NY, 2002. doi:10.1007/978-1-4757-3532-1.

Sinnogenes. Storage INNovations for Green ENErgy Systems (SINNOGENES). EU Horizon Europe Grant Agreement No. 101096992, 2025. https://sinnogenes.eu/. Last checked: 20/11/2025.

The scikit-optimize contributors. Scikit-Optimize: Sequential Model-Based Optimization, 2020. https://scikit-optimize.github.io/stable/. Last checked: 20/11/2025.

C. Thiel and V. Klapperich. Heatcrete: Ein Spezialbeton für die Hochtemperatur-Wärmespeicherung, 2019. https://www.heidelbergcement.de/de/system/files_force/assets/document/energiewandel-regenerative-energiequellen\-heatcrete-energynest-waermespeicher-waermetauschersaeulen-nlt28-2017.pdf. Last checked: 20/11/2025.

Y. Tian, X. Zheng, X. Zhang, and Y. Jin. Efficient Large-Scale Multiobjective Optimization Based on a Competitive Swarm Optimizer. *IEEE Transactions on Cybernetics*, 50:3696–3708, 2020. doi:10.1109/TCYB.2019.2906383.

A. P. Tran and P. Stathopoulos. Dynamic simulation and experimental validation of a high-temperature Brayton heat pump. *Applied Thermal Engineering*, pages 1–42, 2025. doi:https://doi.org/10.1016/j.applthermaleng.2025.126536.

Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí. Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization. *INFORMS Journal on Computing*, 19:328–340, 2007. doi:http://dx.doi.org/10.2139/ssrn.886559.

N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, and E. Pesch. Genetic Local Search Algorithms for the Travelling Salesman Problem. In *International Conference on Parallel Problem Solving from Nature*, pages 109–116, 1990. doi:10.1007/BFb0029740.

G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, 2009. ISBN 978-1-4414-1269-0. https://dl.acm.org/doi/book/10.5555/1593511.

Vestas. 4MW Platform Brochure: V136-4.2 MW, 2025. https://www.vestas.com/en/products/4-mw-platform/v150-4_2_mw. Last checked: 20/11/2025.

Viessmann Deutschland GmbH. Dampferzeuger - Effiziente Anlagen bis 31,5 t/h, 2018. https://www.viessmann.at/de/produkte/grosskessel/vitomax-hs.html. Last checked: 20/11/2025.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, and A. Fouilloy. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569–582, 2017. doi:https://doi.org/10.1016/j.renene.2016.12.095.

W. Wagner and A. Pruss. The IAPWS formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *Journal of Physical and Chemical Reference Data*, 31: 387–535, 2002. doi:10.1063/1.1461829.

J. V. Walden, M. Bähr, A. Glade, J. Gollasch, A. P. Tran, and T. Lorenz. Nonlinear operational optimization of an industrial power-to-heat system with a high temperature heat pump, a thermal energy storage and wind energy. *Applied Energy*, 344, 2023. doi:https://doi.org/10.1016/j.apenergy.2023.121247.

X. Wan. Influence of feature scaling on convergence of gradient iterative algorithm. *Journal of Physics: Conference Series*, 1213:1–5, 2019. doi:10.1088/1742-6596/1213/3/032021.

H. Wang, S. Chen, and L. Luo. A diffusion algorithm based on P systems for continuous global optimization. *Journal of Computational Science*, 44, 2020a. doi:https://doi.org/10.1016/j.jocs.2020.101112.

J. Wang, Q. Zhou, H. Jiang, and R. Hou. Short-Term Wind Speed Forecasting Using Support Vector Regression Optimized by Cuckoo Optimization Algorithm. *Mathematical Problems in Engineering*, 2015:1–13, 2015a. doi:10.1155/2015/619178.

J. Wang, Q. Zhou, H. Jiang, and R. Hou. Short-Term Wind Speed Forecasting Using Support Vector Regression Optimized by Cuckoo Optimization Algorithm. *Mathematical Problems in Engineering*, 2015:1–13, 2015b. doi:https://doi.org/10.1155/2015/619178.

J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Parallel Bayesian Global Optimization of Expensive Functions. *Operations Research*, 68:1850–1865, 2020b. doi:https://doi.org/10.1287/opre.2019.1966.

J. Wang, H. Zhang, Q. Li, and A. Ji. Design and research of hybrid forecasting system for wind speed point forecasting and fuzzy interval forecasting. *Expert Systems with Applications*, 209:1–16, 2022. doi:https://doi.org/10.1016/j.eswa.2022.118384.

Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016. https://dl.acm.org/doi/10.5555/3013558.3013569.

A. G. Watson and R. J. Barnes. Infill sampling criteria to locate extremes. *Mathematical Geology*, 27:589 – 608, 1995. doi:10.1007/BF02093902.

World Resources Institute & World Business Council for Sustainable Development. *The Greenhouse Gas Protocol: A Corporate Accounting and Reporting Standard*. World Resources Institute & World Business Council for Sustainable Development, Washington, DC, revised edition, 2004. https://ghgprotocol.org/corporate-standard.

H. Wu and Z. Xu. Fuzzy Logic in Decision Support: Methods, Applications and Future Trends. *International Journal of Computers, Communications & Control*, 16, 2021. doi:10.15837/ijccc.2021.1.4044.

Q. Wu, F. Guan, C. Lv, and Y. Huang. Ultra-short-term multi-step wind power forecasting based on CNN-LSTM. *IET Renewable Power Generation*, 15:1019–1029, 2021. doi:10.1049/rpg2.12085.

Y. Wu, B. Sicard, and S. A. Gadsden. Physics-informed machine learning: A comprehensive review on applications in anomaly detection and condition monitoring. *Expert Systems with Applications*, 255: 1–40, 2024. doi:https://doi.org/10.1016/j.eswa.2024.124678.

A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 2002. URL https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=71f09bd46a0da1a4a98fb2c227cc4f63f16fa75d.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006. doi:https://doi.org/10.1007/s10107-004-0559-y.

D. Xu, Q. Wu, B. Zhou, C. Li, L. Bai, and S. Huang. Distributed Multi-Energy Operation of Coupled Electricity, Heating, and Natural Gas Networks. *IEEE Transactions on Sustainable Energy*, 11: 2457–2469, 2020. doi:10.1109/TSTE.2019.2961432.

A. Yadav and K. Deep. An efficient co-swarm particle swarm optimization for non-linear constrained optimization. *Journal of Computational Science*, 5:258–268, 2014. doi:https://doi.org/10.1016/j.jocs.2013.05.011.

Q. Yang, G. Huang, T. Li, Y. Xu, and J. Pan. A novel short-term wind speed prediction method based on hybrid statistical-artificial intelligence model with empirical wavelet transform and hyper-parameter optimization. *Journal of Wind Engineering and Industrial Aerodynamics*, 240:1–19, 2023. doi:https://doi.org/10.1016/j.jweia.2023.105499.

Z. Yang and S. Dong. A novel decomposition-based approach for non-stationary hub-height wind speed modelling. *Energy*, 283:1–27, 2023. doi:https://doi.org/10.1016/j.energy.2023.129081.

A. A. A. Zapata, E. G. Suarez, and J. A. V. Florez. Application of VRP Techniques to the Allocation of Resources in an Electric Power Distribution System. *Journal of Computational Science*, 35:102–109, 2019. doi:https://doi.org/10.1016/j.jocs.2019.06.010.

A. Zhadan, A. Martemyanov, A. Allahverdyan, and et al. Linearization method for MINLP energy optimization problems. *Scientific Reports*, 15:1–14, 2025. doi:10.1038/s41598-025-11380-5.

Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang. A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review*, 55:59–110, 2022. doi:10.1007/s10462-021-10042-y.

X. Zhang, X. Zheng, R. Cheng, J. Qiu, and Y. Jin. A competitive mechanism based multi-objective particle swarm optimizer with fast convergence. *Information Sciences*, 427:63–76, 2018. doi:https://doi.org/10.1016/j.ins.2017.10.037.

Étienne Cuisinier, P. Lemaire, B. Penz, A. Ruby, and C. Bourasseau. New rolling horizon optimization approaches to balance short-term and long-term decisions: An application to energy planning. *Energy*, 245:1–22, 2022. doi:https://doi.org/10.1016/j.energy.2021.122773.

D. Šturek and S. Lazarova-Molnar. Surrogate modeling: Review and opportunities for expert knowledge integration. *Procedia Computer Science*, 257:826–833, 2025. doi:https://doi.org/10.1016/j.procs.2025.03.106.

List of Figures

List of Tables

A. Appendix

The appendix provides additional materials that support the research presented in this thesis. These materials include detailed data, extended methodological explanations, supplementary figures and tables, and any other relevant information that is too lengthy to be included in the main sections.

A.1. Cubic Interpolation

This section provides the mathematical foundation for the one-dimensional monotonic and two-dimensional cubic interpolation.

A.1.1. 1-D Monotonic Cubic Interpolation

In this work, the Piecewise Cubic HERMITE Interpolating Polynomial (PCHIP) (Fritsch and Carlson, 1980; Kahaner et al., 1989; Moler, 2004) is used for cubic interpolation. PCHIP is a method that preserves the shape and monotonicity of the input data. This approach ensures that the interpolant maintains the monotonic behavior of the original data points, avoiding unrealistic overshoots or oscillations that can occur with other interpolation methods. The general formula for PCHIP on an interval $[x_i, x_{i+1}]$ is:

$$p_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \tag{A.1}$$

where the coefficients a_i, b_i, c_i, and d_i are defined as:

$$a_i = \frac{m_i + m_{i+1} - 2s_i}{(x_{i+1} - x_i)^2}, \tag{A.2}$$

$$b_i = \frac{3s_i - 2m_i - m_{i+1}}{x_{i+1} - x_i}, \tag{A.3}$$

$$c_i = m_i, \tag{A.4}$$

$$d_i = y_i, \tag{A.5}$$

with m_i describing the slopes and the secant slopes s_i defined as:

$$s_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}. \tag{A.6}$$

To preserve monotonicity, the slopes m_i are determined as follows: If the secant slopes s_{i-1}, s_i have a different sign or either of them equals zero, then x_i is a discrete local minimum or maximum, and m_i is set to 0. If the secant slopes have the same sign and the two intervals have the same length, then m_i is calculated as the harmonic mean of the two discrete slopes:

$$\frac{1}{m_i} = \frac{1}{2}\Big(\frac{1}{s_{i-1}} + \frac{1}{s_i}\Big). \tag{A.7}$$

In the case, where s_{i-1} and s_i have the same sign, but the two intervals have different lengths, then m_i is determined as a weighted harmonic mean, with weights determined by the lengths of the two intervals:

$$\frac{w_1 + w_2}{m_i} = \frac{w_1}{s_{i-1}} + \frac{w_2}{s_i}, \tag{A.8}$$

with:

$$w_1 = 2(x_{i+1} - x_i) + x_i - x_{i-1} = 2x_{i+1} - x_i - x_{i-1}, \tag{A.9}$$

$$w_2 = x_{i+1} - x_i + 2(x_i - x_{i-1}) = x_{i+1} + x_i - 2x_{i-1}. \tag{A.10}$$

A.1.2. 2-D Cubic Interpolation

The bicubic interpolation (Press et al., 2007) approximates a function $f(x, y)$ over a rectangular cell defined by the points:

$$(x_i, y_j), \quad (x_{i+1}, y_j), \quad (x_i, y_{j+1}), \quad (x_{i+1}, y_{j+1}). \tag{A.11}$$

The interpolation is represented by a third-degree polynomial in both variables as follows:

$$p(x, y) = \sum_{m=0}^{3} \sum_{n=0}^{3} a_{mn}(x - x_i)^m (y - y_j)^n. \tag{A.12}$$

To uniquely determine the sixteen coefficients a_{mn}, the method requires knowledge of the function values and derivatives at the four corner points of the cell. These include:

Function values:

$$f_{ij} = f(x_i, y_j), \tag{A.13}$$

$$f_{i+1,j} = f(x_{i+1}, y_j), \tag{A.14}$$

$$f_{i,j+1} = f(x_i, y_{j+1}), \tag{A.15}$$

$$f_{i+1,j+1} = f(x_{i+1}, y_{j+1}). \tag{A.16}$$

Partial derivatives with respect to x:

$$f_{x,ij} = \frac{\partial f}{\partial x}(x_i, y_j), \tag{A.17}$$

$$f_{x,i+1,j} = \frac{\partial f}{\partial x}(x_{i+1}, y_j), \tag{A.18}$$

$$f_{x,i,j+1} = \frac{\partial f}{\partial x}(x_i, y_{j+1}), \tag{A.19}$$

$$f_{x,i+1,j+1} = \frac{\partial f}{\partial x}(x_{i+1}, y_{j+1}). \tag{A.20}$$

Partial derivatives with respect to y:

$$f_{y,ij} = \frac{\partial f}{\partial y}(x_i, y_j), \tag{A.21}$$

$$f_{y,i+1,j} = \frac{\partial f}{\partial y}(x_{i+1}, y_j), \tag{A.22}$$

$$f_{y,i,j+1} = \frac{\partial f}{\partial y}(x_i, y_{j+1}), \tag{A.23}$$

$$f_{y,i+1,j+1} = \frac{\partial f}{\partial y}(x_{i+1}, y_{j+1}). \tag{A.24}$$

Mixed partial derivatives:

$$f_{xy,ij} = \frac{\partial^2 f}{\partial x \partial y}(x_i, y_j), \tag{A.25}$$

$$f_{xy,i+1,j} = \frac{\partial^2 f}{\partial x \partial y}(x_{i+1}, y_j), \tag{A.26}$$

$$f_{xy,i,j+1} = \frac{\partial^2 f}{\partial x \partial y}(x_i, y_{j+1}), \tag{A.27}$$

$$f_{xy,i+1,j+1} = \frac{\partial^2 f}{\partial x \partial y}(x_{i+1}, y_{j+1}). \tag{A.28}$$

These sixteen known quantities form a vector $\boldsymbol{f}$, which is related to the vector of coefficients $\boldsymbol{a}$ by a linear system:

$$\boldsymbol{f} = X\boldsymbol{a}. \tag{A.29}$$

Here, X is a 16×16 matrix encoding the constraints from the corner function values and derivatives. The coefficients $\boldsymbol{a}$ are then found by inverting this matrix:

$$\boldsymbol{a} = X^{-1}\boldsymbol{f}. \tag{A.30}$$

Once $\boldsymbol{a}$ is known, the polynomial can be evaluated for any (x, y) within the cell, yielding a smooth approximation of the original function.

A.2. Wind Speed Forecasting

This section presents the architectures of the LSTM and CNN-LSTM models used in this work, along with additional results for one-step and multi-step-ahead wind speed forecasting.

A.2.1. LSTM & CNN-LSTM Architecture

Fig. A.1 displays the architectures of the LSTM and hybrid CNN-LSTM models considered in this work. The figure on the left illustrates the LSTM model, which consists of two LSTM layers, each followed by a Dropout layer. Dropout is used to mitigate overfitting, a common challenge in deep neural networks (Li et al., 2020), by randomly setting input units to zero during training based on a predefined probability, as suggested by Srivastava et al. (2014). The model concludes with a fully connected (Dense) layer to generate the final output. On the right, the architecture of the hybrid CNN-LSTM model is depicted. The CNN component is responsible for feature extraction, consisting of two one-dimensional convolutional layers followed by a max pooling layer, which selects the maximum value within a defined region. To model the temporal dependencies in the time series, two LSTM layers are incorporated, again using

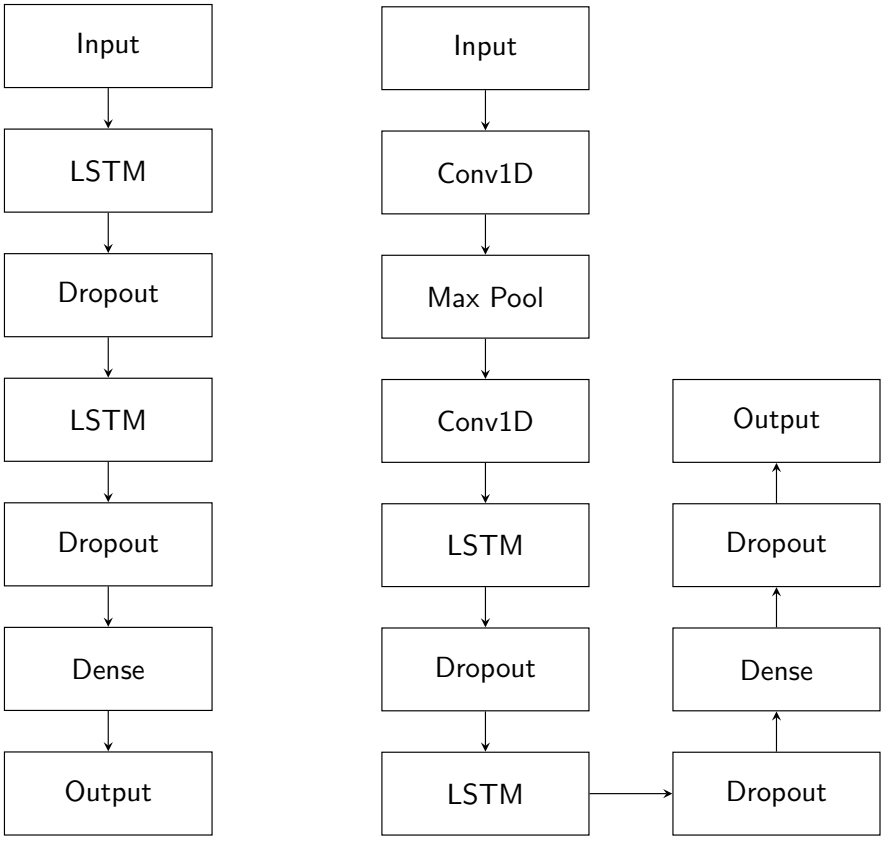

Figure A.1: Network structure of the LSTM model (**left**) and hybrid CNN-LSTM model (**right**).

Dropout layers to address overfitting. Finally, a fully connected (Dense) layer integrates the extracted features and produces the final output.

A.2.2. Results

The results (R^2, RMSE, MAE) of one-step-ahead wind speed forecast are presented in Tables A.1-A.7 for Cottbus (summer 2021) and for Düsseldorf, Augsburg, and Hamburg in both winter and summer 2021. Figs. A.2-A.5 illustrate the forecasting results using the methods described in Section 3.1 for one-step-, four-step-, eight-step-, and twelve-step-ahead wind speed forecasting, exemplified for Düsseldorf (summer).

For a fair comparison, the proposed methods are trained only once at the beginning, as methods like ARIMA, SVR, and ANN are computationally expensive (see Section 3.3.1). After one, four, eight, and twelve steps, the methods update their training data with real data without retraining and continue forecasting. However, as shown in Tables A.8 and A.9, retraining MLR, MPR, KNN, and HGBR after each prediction horizon does not significantly impact the model accuracy.

This study also investigates the influence of various multi-step forecasting strategies on MLR, MPR, HGBR, and KNN, as presented in Tables A.10-A.15, exemplified for a two-day dataset in Cottbus (winter) and Düsseldorf (summer). In the DirMO strategy (for forecasting steps eight and twelve), different entries correspond to results obtained using various numbers of blocks: two blocks for a four-step horizon, two or three blocks for an eight-step horizon, and two, three, or four blocks for a twelve-step horizon. As shown in the tables, increasing the number of blocks leads to higher total CPU time – including forecasting model training and executing the strategy – on the two-day datasets without improving the model accuracy. It is also worth noticing that longer forecast horizons in the Rec, MIMO, and DirMO strategies require less CPU time than shorter horizons, as seen in Tables A.10-A.15. This is primarily because the forecasting models need to be retrained fewer times for longer horizons, reducing the overall computational cost. All results in this section are analyzed in detail in Section 3.3.

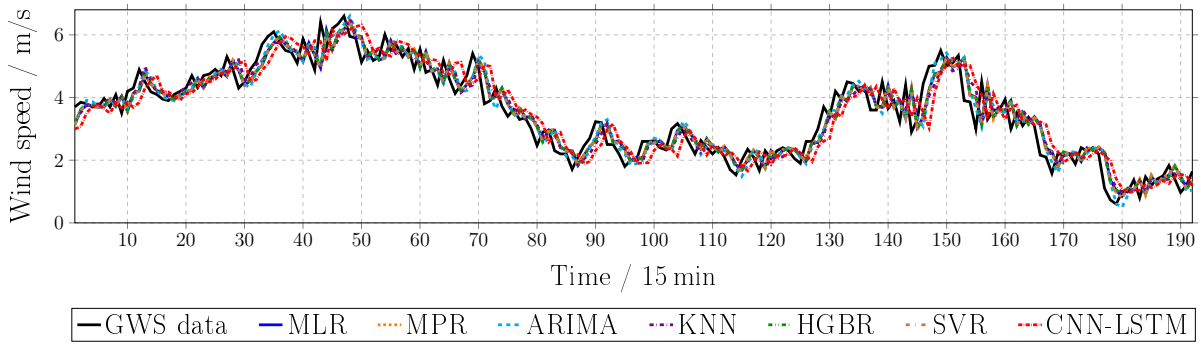

Figure A.2: Results of one-step-ahead wind speed forecast, exemplified for Düsseldorf in summer 2021.

Table A.1: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Cottbus in summer 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	87.63	0.48	0.35
MPR	87.69	0.48	0.35
ARIMA	85.41	0.52	0.39
KNN	86.35	0.50	0.37
RFR	87.57	0.48	0.35
ETR	87.66	0.48	0.35
HGBR	87.65	0.48	0.35
SVR	87.71	0.48	0.35
LSTM	83.72	0.53	0.40
CNN-LSTM	86.02	0.51	0.38

Table A.2: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Düsseldorf in winter 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	91.17	0.64	0.45
MPR	91.18	0.64	0.45
ARIMA	92.87	0.62	0.43
KNN	89.54	0.69	0.50
RFR	91.04	0.64	0.45
ETR	91.11	0.64	0.45
HGBR	91.02	0.64	0.45
SVR	91.23	0.64	0.45
LSTM	86.66	0.73	0.54
CNN-LSTM	88.34	0.71	0.52

Table A.3: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Düsseldorf in summer 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	89.94	0.64	0.46
MPR	89.94	0.64	0.46
ARIMA	89.95	0.64	0.46
KNN	88.29	0.69	0.50
RFR	89.90	0.64	0.46
ETR	89.81	0.64	0.46
HGBR	89.86	0.64	0.46
SVR	89.87	0.64	0.46
LSTM	81.68	0.79	0.59
CNN-LSTM	86.13	0.75	0.55

Table A.4: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Augsburg in winter 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	88.83	0.63	0.45
MPR	88.87	0.63	0.45
ARIMA	85.98	0.71	0.52
KNN	86.99	0.69	0.50
RFR	88.72	0.63	0.45
ETR	88.90	0.63	0.45
HGBR	88.77	0.63	0.45
SVR	88.91	0.63	0.45
LSTM	82.91	0.76	0.57
CNN-LSTM	84.73	0.74	0.55

Table A.5: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Augsburg in summer 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	87.08	0.64	0.46
MPR	87.12	0.64	0.46
ARIMA	86.42	0.66	0.47
KNN	85.33	0.68	0.49
RFR	86.94	0.64	0.46
ETR	87.03	0.64	0.46
HGBR	87.05	0.64	0.46
SVR	87.18	0.64	0.46
LSTM	82.44	0.74	0.55
CNN-LSTM	83.19	0.73	0.54

Table A.6: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Hamburg in winter 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	91.57	0.61	0.44
MPR	91.67	0.61	0.44
ARIMA	92.02	0.60	0.43
KNN	90.41	0.65	0.48
RFR	91.53	0.61	0.44
ETR	91.59	0.61	0.44
HGBR	91.58	0.61	0.44
SVR	91.65	0.61	0.44
LSTM	84.44	0.76	0.57
CNN-LSTM	88.54	0.74	0.55

Table A.7: Comparative analysis (R^2, RMSE, MAE) of one-step-ahead wind speed forecast results for the presented methods, exemplified for Hamburg in summer 2021.

Methods	R^2 (%)	RMSE (m/s)	MAE (m/s)
MLR	90.69	0.64	0.46
MPR	90.78	0.64	0.46
ARIMA	88.49	0.69	0.51
KNN	89.41	0.68	0.50
RFR	90.56	0.64	0.46
ETR	90.60	0.64	0.46
HGBR	90.54	0.64	0.46
SVR	90.76	0.64	0.46
LSTM	86.08	0.73	0.55
CNN-LSTM	87.95	0.72	0.54

Table A.8: MAE comparison of wind speed forecast results for MLR, MPR, HGBR, and KNN with respect to different multi-step-ahead horizons without and with retrain them after the forecast horizons, exemplified for Cottbus in winter 2021.

Methods	Metric	MLR	MPR	HGBR	KNN
one-step-ahead	MAE (without retrain) (m/s)	0.23	0.23	0.24	0.23
	MAE (with retrain) (m/s)	0.23	0.23	0.24	0.23
four-step-ahead	MAE (without retrain) (m/s)	0.31	0.31	0.30	0.30
	MAE (with retrain) (m/s)	0.31	0.31	0.30	0.30
eight-step-ahead	MAE (without retrain) (m/s)	0.37	0.37	0.32	0.35
	MAE (with retrain) (m/s)	0.37	0.37	0.32	0.35
twelve-step-ahead	MAE (without retrain) (m/s)	0.45	0.45	0.38	0.39
	MAE (with retrain) (m/s)	0.45	0.45	0.38	0.39

Table A.9: MAE comparison of wind speed forecast results for MLR, MPR, HGBR, and KNN with respect to different multi-step-ahead horizons without and with retrain them after the forecast horizons, exemplified for Düsseldorf in summer 2021.

Methods	Metric	MLR	MPR	HGBR	KNN
one-step-ahead	MAE (without retrain) (m/s)	0.38	0.38	0.38	0.38
	MAE (with retrain) (m/s)	0.38	0.38	0.38	0.38
four-step-ahead	MAE (without retrain) (m/s)	0.52	0.52	0.48	0.51
	MAE (with retrain) (m/s)	0.52	0.52	0.48	0.51
eight-step-ahead	MAE (without retrain) (m/s)	0.66	0.66	0.62	0.66
	MAE (with retrain) (m/s)	0.66	0.66	0.62	0.66
twelve-step-ahead	MAE (without retrain)(m/s)	0.71	0.71	0.68	0.68
	MAE (with retrain) (m/s)	0.71	0.71	0.68	0.68

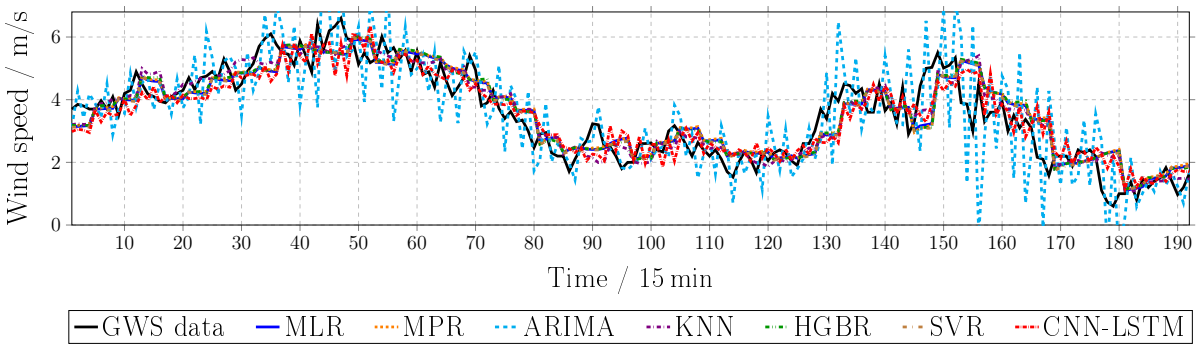

Figure A.3: Results of four-step-ahead wind speed forecast, exemplified for Düsseldorf in summer 2021.

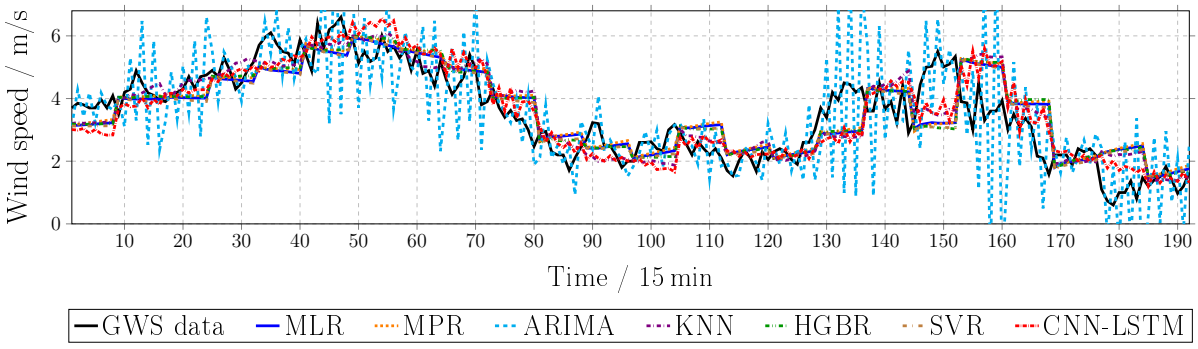

Figure A.4: Results of eight-step-ahead wind speed forecast, exemplified for Düsseldorf in summer 2021.

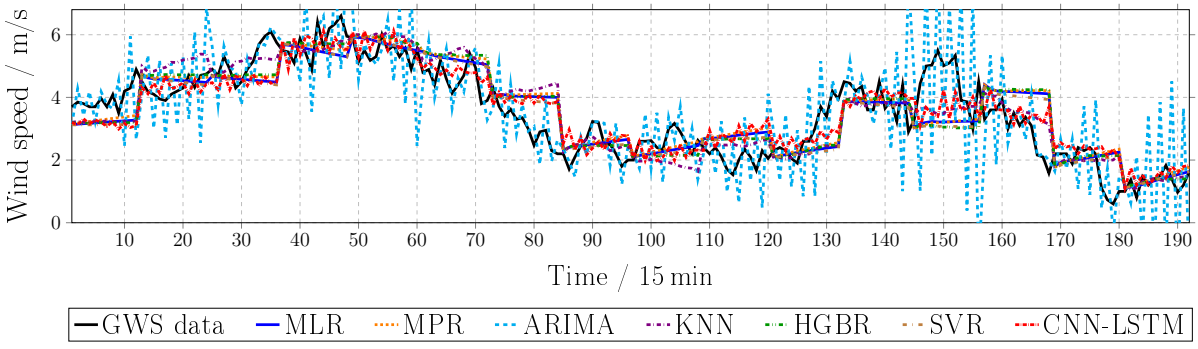

Figure A.5: Results of twelve-step-ahead wind speed forecast, exemplified for Düsseldorf in summer 2021.

Table A.10: Comparative analysis (RMSE, MAE, CPU time) of four-step-ahead wind speed forecast results for different forecasting strategies applied to MLR, MPR, HGBR, and KNN, exemplified for Cottbus in winter 2021. CPU time refers to the total time required for training the forecasting model and executing the forecasting strategy.

Strategy	Metric	MLR	MPR	HGBR	KNN
Rec	RMSE (m/s)	0.38	0.38	0.37	0.37
	MAE (m/s)	0.31	0.31	0.30	0.30
	CPU time (s)	4	13	102	132
Dir	RMSE (m/s)	0.38	0.38	0.38	0.38
	MAE (m/s)	0.31	0.31	0.30	0.30
	CPU time (s)	15	46	378	496
MIMO	RMSE (m/s)	0.38	0.38	0.38	0.38
	MAE (m/s)	0.31	0.31	0.30	0.30
	CPU time (s)	7	17	175	230
DirRec	RMSE (m/s)	0.38	0.38	0.38	0.38
	MAE (m/s)	0.31	0.31	0.30	0.30
	CPU time (s)	17	55	423	560
DirMO	RMSE (m/s)	0.38	0.38	0.38	0.38
	MAE (m/s)	0.31	0.31	0.30	0.30
	CPU time (s)	8	23	201	260

Table A.11: Comparative analysis (RMSE, MAE, CPU time) of eight-step-ahead wind speed forecast results for different forecasting strategies applied to MLR, MPR, HGBR, and KNN, exemplified for Cottbus in winter 2021. CPU time refers to the total time required for training the forecasting model and executing the forecasting strategy. In the DirMO strategy, the different entries in the CPU time correspond to results obtained using two and three blocks.

Strategy	Metric	MLR	MPR	HGBR	KNN
Rec	RMSE (m/s)	0.46	0.46	0.42	0.45
	MAE (m/s)	0.37	0.37	0.32	0.35
	CPU time (s)	3	7	76	102
Dir	RMSE (m/s)	0.46	0.46	0.42	0.45
	MAE (m/s)	0.37	0.37	0.32	0.35
	CPU time (s)	15	46	378	496
MIMO	RMSE (m/s)	0.46	0.46	0.42	0.45
	MAE (m/s)	0.37	0.37	0.32	0.35
	CPU time (s)	5	12	125	160
DirRec	RMSE (m/s)	0.46	0.46	0.42	0.45
	MAE (m/s)	0.37	0.37	0.32	0.35
	CPU time (s)	17	55	423	560
DirMO	RMSE (m/s)	0.46	0.46	0.42	0.45
	MAE (m/s)	0.37	0.37	0.32	0.35
	CPU time (s)	6/10	15/25	150/252	200/324

Table A.12: Comparative analysis (RMSE, MAE, CPU time) of twelve-step-ahead wind speed forecast results for different forecasting strategies applied to MLR, MPR, HGBR, and KNN, exemplified for Cottbus in winter 2021. CPU time refers to the total time required for training the forecasting model and executing the forecasting strategy. In the DirMO strategy, the different entries in the CPU time correspond to results obtained using two, three, and four blocks.

Strategy	Metric	MLR	MPR	HGBR	KNN
Rec	RMSE (m/s)	0.52	0.52	0.45	0.46
	MAE (m/s)	0.45	0.45	0.38	0.39
	CPU time (s)	2	6	51	68
Dir	RMSE (m/s)	0.52	0.52	0.45	0.46
	MAE (m/s)	0.45	0.45	0.38	0.39
	CPU time (s)	15	46	378	496
MIMO	RMSE (m/s)	0.52	0.52	0.45	0.46
	MAE (m/s)	0.45	0.45	0.38	0.39
	CPU time (s)	4	9	98	130
DirRec	RMSE (m/s)	0.52	0.52	0.45	0.46
	MAE (m/s)	0.45	0.45	0.38	0.39
	CPU time (s)	17	55	423	560
DirMO	RMSE (m/s)	0.52	0.52	0.45	0.46
	MAE (m/s)	0.45	0.45	0.38	0.39
	CPU time (s)	6/7/8	12/17/19	151/175/202	199/230/266

Table A.13: Comparative analysis (RMSE, MAE, CPU time) of four-step-ahead wind speed forecast results for different forecasting strategies applied to MLR, MPR, HGBR, and KNN, exemplified for Düsseldorf in summer 2021. CPU time refers to the total time required for training the forecasting model and executing the forecasting strategy.

Strategy	Metric	MLR	MPR	HGBR	KNN
Rec	RMSE (m/s)	0.65	0.65	0.62	0.64
	MAE (m/s)	0.52	0.52	0.48	0.51
	CPU time (s)	4	13	98	134
Dir	RMSE (m/s)	0.65	0.65	0.62	0.64
	MAE (m/s)	0.52	0.52	0.48	0.51
	CPU time (s)	16	46	399	530
MIMO	RMSE (m/s)	0.65	0.65	0.62	0.64
	MAE (m/s)	0.52	0.52	0.48	0.51
	CPU time (s)	7	17	177	227
DirRec	RMSE (m/s)	0.65	0.65	0.62	0.64
	MAE (m/s)	0.52	0.52	0.48	0.51
	CPU time (s)	17	56	429	564
DirMO	RMSE (m/s)	0.65	0.65	0.62	0.64
	MAE (m/s)	0.52	0.52	0.48	0.51
	CPU time (s)	8	24	203	261

Table A.14: Comparative analysis (RMSE, MAE, CPU time) of eight-step-ahead wind speed forecast results for different forecasting strategies applied to MLR, MPR, HGBR, and KNN, exemplified for Düsseldorf in summer 2021. CPU time refers to the total time required for training the forecasting model and executing the forecasting strategy. In the DirMO strategy, the different entries in the CPU time correspond to results obtained using two and three blocks.

Strategy	Metric	MLR	MPR	HGBR	KNN
Rec	RMSE (m/s)	0.83	0.83	0.79	0.83
	MAE (m/s)	0.66	0.66	0.62	0.66
	CPU time (s)	3	8	76	102
Dir	RMSE (m/s)	0.83	0.83	0.79	0.83
	MAE (m/s)	0.66	0.66	0.62	0.66
	CPU time (s)	16	46	399	530
MIMO	RMSE (m/s)	0.83	0.83	0.79	0.83
	MAE (m/s)	0.66	0.66	0.62	0.66
	CPU time (s)	6	12	148	201
DirRec	RMSE (m/s)	0.83	0.83	0.79	0.83
	MAE (m/s)	0.66	0.66	0.62	0.66
	CPU time (s)	17	56	429	564
DirMO	RMSE (m/s)	0.83	0.83	0.79	0.83
	MAE (m/s)	0.66	0.66	0.62	0.66
	CPU time (s)	7/9	16/26	176/229	233/296

Table A.15: Comparative analysis (RMSE, MAE, CPU time) of twelve-step-ahead wind speed forecast results for different forecasting strategies applied to MLR, MPR, HGBR, and KNN, exemplified for Düsseldorf in summer 2021. CPU time refers to the total time required for training the forecasting model and executing the forecasting strategy. In the DirMO strategy, the different entries in the CPU time correspond to results obtained using two, three, and four blocks.

Strategy	Metric	MLR	MPR	HGBR	KNN
Rec	RMSE (m/s)	0.89	0.89	0.87	0.87
	MAE (m/s)	0.71	0.71	0.68	0.68
	CPU time (s)	2	7	51	68
Dir	RMSE (m/s)	0.89	0.89	0.87	0.87
	MAE (m/s)	0.71	0.71	0.68	0.68
	CPU time (s)	16	46	399	530
MIMO	RMSE (m/s)	0.89	0.89	0.87	0.87
	MAE (m/s)	0.71	0.71	0.68	0.68
	CPU time (s)	5	11	123	172
DirRec	RMSE (m/s)	0.89	0.89	0.87	0.87
	MAE (m/s)	0.71	0.71	0.68	0.68
	CPU time (s)	17	56	429	564
DirMO	RMSE (m/s)	0.89	0.89	0.87	0.87
	MAE (m/s)	0.71	0.71	0.68	0.68
	CPU time (s)	6/7/8	13/17/19	145/174/203	194/237/274

A.3. Numerical Optimization

This section provides supplementary data, algorithms, and results related to Chapter 4 that are not included in the main text.

A.3.1. AL Framework with Slack Variables

AL methods are iterative procedures (Picheny et al., 2016), where the following subproblem, derived from (4.3), is solved using the current parameter sequence $\left(\boldsymbol{\lambda}_{\boldsymbol{g}}^{j-1}, \boldsymbol{\lambda}_{\boldsymbol{h}}^{j-1}, \rho^{j-1}\right)$:

$$\min_{\boldsymbol{x}\in\Omega} \left\{ u\left(\boldsymbol{x}, \boldsymbol{\lambda}_{\boldsymbol{g}}^{j-1}, \boldsymbol{\lambda}_{\boldsymbol{h}}^{j-1}, \rho^{j-1}, \boldsymbol{s}(\boldsymbol{x})\right) \right\}, \tag{A.31}$$

with the slack variable vector in (4.4) being dependent on $\boldsymbol{x}$. The LAGRANGE multiplier vectors $\boldsymbol{\lambda_g}$, $\boldsymbol{\lambda_h}$ for the inequality and equality constraints, the penalty parameter ρ, and the slack variables $\boldsymbol{s}$ are updated based on Algorithm 3 depending on the solution of (A.31). The vector $\boldsymbol{\epsilon} \geq \mathbf{0}$ (line 7) serves as a threshold for equality constraints and is set to $\mathbf{10^{-2}}$, following the recommendation of Picheny et al. (2016). Solving (A.31) requires an initialization of the parameter sequence. Here, we adopt the settings proposed in Gramacy (2020), where $\boldsymbol{\lambda}_{\boldsymbol{g}}^0 = \boldsymbol{\lambda}_{\boldsymbol{h}}^0 = \mathbf{0}$, and the initial ρ^0 is determined using the following rule:

$$\rho^0 = \frac{\min\limits_{i=1,\dots,N_0} \left\{ \sum\limits_{w=1}^{q} \max(0, g_w(\boldsymbol{x}_i))^2 + \sum\limits_{r=1}^{p} h_r^2(\boldsymbol{x}_i) : \exists w, r, v_w(\boldsymbol{x}_i) = 0, v_r(\boldsymbol{x}_i) = 0 \right\}}{2 \min\limits_{i=1,\dots,N_0} \left\{ f(\boldsymbol{x}_i) : \forall w, r, v_w(\boldsymbol{x}_i) = 1, v_r(\boldsymbol{x}_i) = 1 \right\}}. \tag{A.32}$$

The vector $\boldsymbol{v}(\boldsymbol{x}_i)$ denotes the logical vector, which determines the validity of $\boldsymbol{x}$ in a zero-slack setting. If the w^{th} inequality constraint is fulfilled, i.e., $g_w(\boldsymbol{x}) \leq 0$, let $v_w(\boldsymbol{x}) = 1$ for $w = 1, \dots, q$ and if the r^{th} equality constraint is fulfilled, i.e., $|h_r(\boldsymbol{x})| \leq \epsilon$, let $v_r(\boldsymbol{x}) = 1$ for $r = 1, \dots, p$; otherwise let $v_w(\boldsymbol{x}) = v_r(\boldsymbol{x}) = 0$. If the denominator for the initial design is not defined, i.e., the initial design has no valid values, then the median of $f(\boldsymbol{x}_i)$ is used in the denominator instead. Otherwise, if the initial design has no invalid values, i.e., the numerator is not defined, then $\rho^0 = 1$. To approximately solve (A.31) (line 3) at each outer iteration K, the BO solver is terminated after a single iteration, as recommended in Picheny et al. (2016). In other words, at each K, the best $\boldsymbol{x}$ found so far is chosen from the set of points D, aligning with the search conducted by the acquisition function at each outer iteration.

When BO and the AL framework are integrated with a local deterministic solver such as IPOPT, as proposed in this work, the parameter update process is significantly simplified. Let $j = 0$: since $\boldsymbol{\lambda}_{\boldsymbol{g}}^0 = \boldsymbol{\lambda}_{\boldsymbol{h}}^0 = \mathbf{0}$ and a local optimum within D_1 satisfies either $g_w(\boldsymbol{x}_1^\star) = 0$ or $g_w(\boldsymbol{x}_1^\star) < 0$ (neglecting here rounding errors), it follows that $s_w(\boldsymbol{x}_1^\star) = 0$ or $s_w(\boldsymbol{x}_1^\star) = -g_w(\boldsymbol{x}_1^\star)$ for $w = 1, \dots, q$. Consequently, this leads to $\lambda_{w,\boldsymbol{g}}^1 = 0$ for all w (see line 5). This also holds for $j > 0$, meaning that the LAGRANGE multipliers for the inequality constraints remain constant throughout the optimization process, i.e., $\lambda_{w,\boldsymbol{g}}^{j+1} = 0$ for all w, j. The same applies to the LAGRANGE multipliers for the equality constraints. Since local optima satisfy

Algorithm 3 AL method with slack variables

Input: Initial LAGRANGE multiplier vectors $\boldsymbol{\lambda}_{\boldsymbol{g}}^0$, $\boldsymbol{\lambda}_{\boldsymbol{h}}^0 \geq 0$; initial penalty parameter $\rho^0 > 0$
1: $j = 0$
2: **while** $j < K$ **do**
3: Let $\boldsymbol{x}^{j+1} \in D_{j+1}$ approximately solve (A.31)
4: Compute $s_w(\boldsymbol{x}^{j+1}) = \max\left\{0, -\lambda_{w,\boldsymbol{g}}^j \rho^j - g_w(\boldsymbol{x}^{\boldsymbol{j+1}})\right\}$ for $w = 1{:}q$
5: Set $\lambda_{w,\boldsymbol{g}}^{j+1} = \lambda_{w,\boldsymbol{g}}^j + \frac{1}{\rho^j}\left(g_w(\boldsymbol{x}^{j+1}) + s_w(\boldsymbol{x}^{j+1})\right)$ for $w = 1{:}q$
6: Set $\lambda_{r,\boldsymbol{h}}^{j+1} = \lambda_{r,\boldsymbol{h}}^j + \frac{1}{\rho^j} h_r(\boldsymbol{x}^{j+1})$ for $r = 1{:}p$
7: **if** $\boldsymbol{g}(\boldsymbol{x}^{j+1}) \leq \mathbf{0}$ and $|\boldsymbol{h}(\boldsymbol{x}^{j+1})| \leq \boldsymbol{\epsilon}$ **then**
8: Set $\rho^{j+1} = \rho^j$
9: **else**
10: Set $\rho^{j+1} = \frac{1}{2}\rho^j$
11: **end if**
12: $j := j + 1$
13: **end while**

$h_r(\boldsymbol{x}_1^\star) = 0$ for $r = 1, \ldots, p$ (neglecting rounding errors), it follows that $\lambda_{r,\boldsymbol{h}}^1 = 0$ and therefore, $\lambda_{r,\boldsymbol{h}}^{j+1} = 0$ (see line 6). As a result, ρ^{j+1} remains constant at each outer iteration K since all constraints are satisfied at local optima. These simplifications hold due to the use of local solvers such as IPOPT, which leads (4.3) to be transformed to $u(\boldsymbol{x}_k^\star, \boldsymbol{s}(\boldsymbol{x}_k^\star)) \equiv f(\boldsymbol{x}_k^\star)$ for $k = z+1, \ldots, z+N_{\text{bc}}$ at each K. In simple terms, the objective function value for the AL technique at each local minimum is equal to the local minimum provided by IPOPT (line 9 of Algorithm 2), since $\boldsymbol{\lambda}_{\boldsymbol{g}} = \boldsymbol{\lambda}_{\boldsymbol{h}} = \mathbf{0}$ and therefore, $\boldsymbol{s}(\boldsymbol{x}_k^\star) = -\boldsymbol{g}(\boldsymbol{x}_k^\star)$.

A.3.2. MS-IPOPT

This section presents the algorithm for the classical random MS-IPOPT (see Algorithm 4). Each MS outer iteration follows two main steps: First, a starting point is randomly generated, and then a local search is performed from that point. For higher N_{bc} values, the algorithm is adjusted accordingly in line 2 to generate the appropriate number of random starting points.

Algorithm 4 MS-IPOPT

Input: Number of outer iterations K
1: **while** $j < K$ **do**
2: Generate a random point $\boldsymbol{x}_i$ in Ω
3: Solve $[y_i^\star, \boldsymbol{x}_i^\star] = \text{IPOPT}(\boldsymbol{x}_i)$
4: **end while**
5: **return** $y_{\text{min}} = \min\{y_j^\star\}_{j=1}^K$

A.3.3. Hybrid Method BOwLS

The recently introduced hybrid BOwLS method (Gao et al., 2020) shares similarities with our proposed BO-IPOPT approach. From an algorithmic perspective, the implementation of BOwLS (see Algorithm 5)

differs from BO-IPOPT (see Algorithm 2) in two key steps. However, there are additional technical distinctions between the two methods, which are presented below.

Original BOwLS

In Gao et al. (2020), the integration of local search within the BO framework aims to effectively determine the starting points for the local solver, thereby constructing an appropriate MS formulation. The BOwLS approach differs from our method in two key aspects: First, instead of using randomly selected points, BOwLS constructs the initial GPR based on the results of local searches (see line 2 of Algorithm 5). This approach carries the risk of an overly restricted sampling region for the initial GPR, potentially slowing down the optimizer convergence – an issue discussed in Appendix A.3.7. Second, both the initial and updated GPR model (see lines 3 and 10) are trained using the objective function values $\boldsymbol{y}^\star$ of the local optima but are paired with the initial points before applying the local solver, i.e., $\boldsymbol{x}$ instead of $\boldsymbol{x}^\star$. As a result, the GPR does not approximate the original function directly but rather a modified version that shares the same known minima. This approach introduces the risk of relying on infeasible solutions, making the surrogate model's training process more challenging. For further details, the reader is referred to the original work (Gao et al., 2020).

From a theoretical perspective, the BOwLS method was initially designed for unconstrained optimization problems. Specifically, it employs the conjugate gradient method from the SCIPY package as a local search strategy to optimize the objective function f without accounting for any constraints. However, this significantly restricts its applicability, as most real-world optimization problems involve constraints. Moreover, as previously discussed, integrating constraints into the BO framework introduces both theoretical and practical challenges. The latter is particularly intricate and is a key focus of our work.

Constraint Handling in BOwLS

Since this study focuses on constrained optimization problems as defined in (4.1), the BOwLS method must be adapted to handle both equality and inequality constraints. To ensure a fair comparison, IPOPT is used in place of the original local solver (conjugate gradient method). Within the BO component of BOwLS, two common techniques are available for managing constraints (see Section 4.2.1): incorporating constraints as penalty terms in the objective function (4.2) or employing the AL framework (4.3).

However, the AL approach is not well-suited for BOwLS and results in reduced optimizer performance compared to the penalty term method. This can be attributed to the fact that the AL framework updates its parameters based on the best $\boldsymbol{x}$ value in D from the previous outer iteration K (see Algorithm 3). In BOwLS, this value is not necessarily a local minimum, making it unlikely that constraints are satisfied. Consequently, the LAGRANGE multipliers $\lambda_{w,\boldsymbol{g}}^{j+1}$ and $\lambda_{r,\boldsymbol{h}}^{j+1}$ (for $w = 1, \ldots, q$ and $r = 1, \ldots, p$) do not vanish during the optimization process. As a result, the condition $u(\boldsymbol{x}_k, \boldsymbol{s}(\boldsymbol{x}_k)) \neq f(\boldsymbol{x}_k^\star)$ holds for $k = z+1, \ldots, z+N_{\text{bc}}$ (in line 9 of Algorithm 5), since $\boldsymbol{\lambda}_{\boldsymbol{g}}, \boldsymbol{\lambda}_{\boldsymbol{h}} \neq \mathbf{0}$ and thus $\boldsymbol{s}(\boldsymbol{x}_k) \neq -\boldsymbol{g}(\boldsymbol{x}_k)$. This

Algorithm 5 Adapted Hybrid Method BOwLS

Input: Length scale l; regularization term α; exploration ξ; number of initialization points N_0; number of candidates N_{c} considered in EI; number of best candidates N_{bc}; penalty weight vectors $\boldsymbol{\delta}, \boldsymbol{\tau}$; number of outer iterations K

Generate an initial set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}\}$ in Ω
Solve $[y^\star_{i_0}, \boldsymbol{x}^\star_{i_0}] = \text{LS}(\boldsymbol{x}_{i_0})$ for $i_0 = 1{:}N_0$
Let $D_0 = \{(\boldsymbol{x}_{i_0}, y^\star_{i_0})\}_{i_0=1}^{N_0}$
Construct a GPR $\hat{u}_0$ from D_0
$z = N_0, j = 0$
while $j < K$ **do**
    Generate a new group of points $\{\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_{N_{\text{c}}}\}$ in Ω
    $\{\boldsymbol{x}_{z+1}, \ldots, \boldsymbol{x}_{z+N_{\text{bc}}}\} \in \underset{\boldsymbol{x} \in D_j}{\arg\max}\, \text{EI}(\bar{\boldsymbol{x}}_i | \hat{u}_z)$ for $i = 1{:}N_{\text{c}}$
    Solve $[y^\star_k, \boldsymbol{x}^\star_k] = \text{LS}(\boldsymbol{x}_k)$ for $k = z+1{:}z+N_{\text{bc}}$
    $D_{j+1} = D_j \cup \{(\boldsymbol{x}_{z+1}, y^\star_{z+1}), \ldots, (\boldsymbol{x}_{z+N_{\text{bc}}}, y^\star_{z+N_{\text{bc}}})\}$
    Update GPR $\hat{u}_{j+1}$ from D_{j+1}
    $z := z + N_{\text{bc}}, j := j + 1$
end while
return $y_{\text{min}} = \min\{y^\star_j\}_{j=1}^{KN_{\text{bc}}}$

leads to significant parameter fluctuations and instability in (4.3), negatively impacting the convergence of the optimizer.

In contrast, BO-IPOPT, which also employs the AL framework, does not suffer from this issue (as discussed in Section A.3.1) because its parameter updates are based on local solutions. Given these observations, we recommend using the penalty term approach (4.2) for BOwLS. This method is adopted in our comparisons with BO-IPOPT and MS-IPOPT across all experiments in this study.

A.3.4. Input Data for DED and RSG

The optimization problems presented in Sections 4.3.3 and 4.3.4 require input data. For both problems, a typical one-day operation is considered, with a time step of one hour, dividing the day into 24 intervals. For the DED problem (4.22)-(4.25), we distinguish between the use of 5, 10, and 30 units. The scenario input data for the 5- and 10-unit system is taken from Santra et al. (2020), with the load demand and unit data described in Fig. A.6 and Tables A.16 and A.17, respectively. The data for the 30-unit system are generated by simply tripling the 10-unit system data. The required data for the WT power production and the grid electricity price for the RSG problem (4.26)-(4.35) are shown in Fig. A.6.

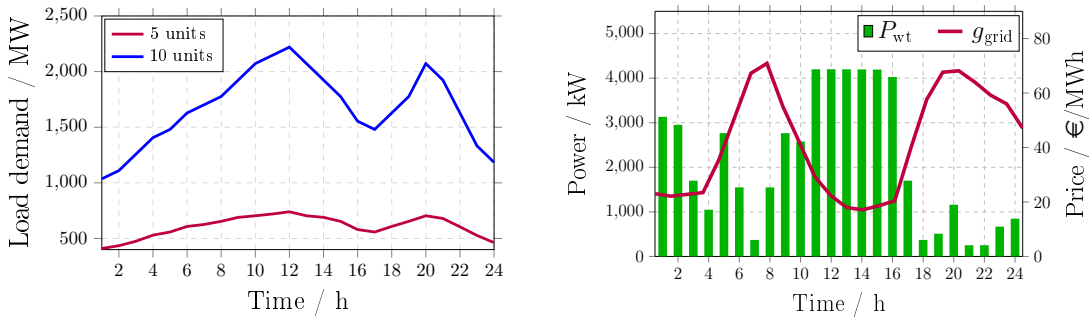

Figure A.6: Visualization of 24 h-scenario input data: load demand for the 5- and 10-unit DED problem (**left**) as well as P_{WT} (WT power output) and g_{grid} (electricity price) for the RSG problem (**right**).

Table A.16: Scenario unit data for the 5-unit DED problem (4.22)-(4.25) taken from Santra et al. (2020). Currency values are given in USD, represented by the $ sign for brevity.

Units	a ($/(MW^{2}h))	b ($/(MWh))	c ($/h)	e ($/h)	f (rad/MW)	P_{min} (MW)	P_{max} (MW)	DR (MW/h)	UR (MW/h)
Unit 1	0.0080	2.0	25	100	0.042	10	75	30	30
Unit 2	0.0030	1.8	60	140	0.040	20	125	30	30
Unit 3	0.0012	2.1	100	160	0.038	30	175	40	40
Unit 4	0.0010	2.0	120	180	0.037	40	250	50	50
Unit 5	0.0015	1.8	40	200	0.035	50	300	50	50

Table A.17: Scenario unit data for the 10-unit DED problem (4.22)-(4.25) taken from Santra et al. (2020). Currency values are given in USD, represented by the $ sign for brevity.

Units	a ($/(MW^{2}h))	b ($/(MWh))	c ($/h)	e ($/h)	f (rad/MW)	P_{min} (MW)	P_{max} (MW)	DR (MW/h)	UR (MW/h)
Unit 1	0.00043	21.60	958.20	450	0.041	150	470	80	80
Unit 2	0.00063	21.05	1313.6	600	0.036	135	460	80	80
Unit 3	0.00039	20.81	604.97	320	0.028	73	340	80	80
Unit 4	0.00070	23.90	471.60	260	0.052	60	300	50	50
Unit 5	0.00079	21.62	480.29	280	0.063	73	243	50	50
Unit 6	0.00056	17.87	601.75	310	0.048	57	160	50	50
Unit 7	0.00211	16.51	502.70	300	0.086	20	130	30	30
Unit 8	0.00480	23.23	639.40	340	0.082	47	120	30	30
Unit 9	0.10908	19.58	455.60	270	0.098	20	80	30	30
Unit 10	0.00951	22.54	692.40	380	0.094	55	55	30	30

A.3.5. BO-IPOPT: Fixed vs. Adaptive Length Scale

In this section, the performance of BO-IPOPT using two different strategies for the length scale in the GP model is compared: a fixed value ($l = 100$) versus an adaptive one. In the adaptive approach, the length scale is optimized at each outer iteration using the L-BFGS-B optimizer initialized from 20 different points. The results are presented in Fig. A.7. While the adaptive length scale can enhance the convergence rate by better fitting the local structure of the objective function, it also leads to a substantial

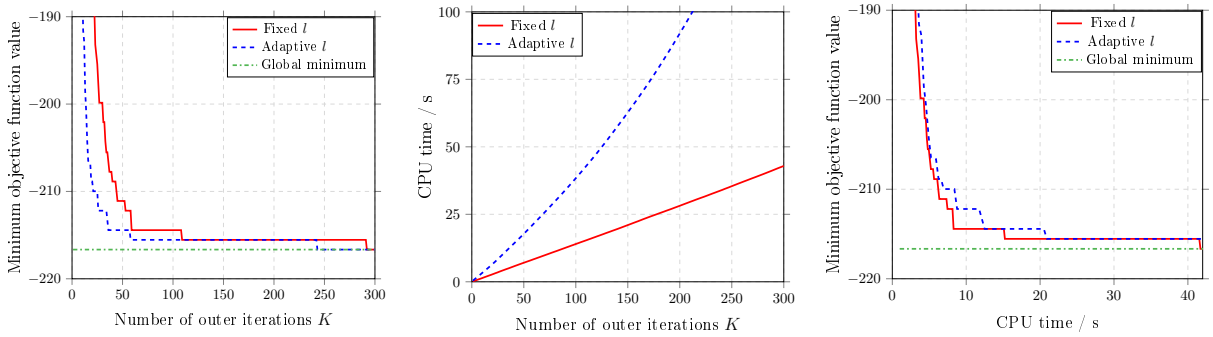

Figure A.7: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for fixed and adaptive length scale l as a function of the number of outer iterations K.

increase in computational time. In contrast, using a fixed length scale provides global guidance in the BO part of the hybrid method and significantly reduces the CPU time, resulting in slightly better overall solver performance. This trade-off becomes even more critical in higher-dimensional settings, where repeated hyperparameter tuning imposes a much greater computational burden – further motivating the use of fixed hyperparameters in this study.

A.3.6. Linear Model in BO-IPOPT: Parameter Study and Setting

To ensure an effective application of the BO-IPOPT framework with a linear surrogate model, a parameter study was conducted to evaluate the influence of key parameters on the optimization performance. The parameters under investigation are the weighting factor w_u for the quality criterion, the initial number of samples N_0, the number of candidates N_c, and the batch size N_bc. The parameter study has been conducted analogously to the GP model case, considering the constrained ACKLEY function and the DED problem with 5 units. The simple test case has been excluded, as variations of the parameters in the GP model already showed little impact on the BO-IPOPT performance for that problem. This allows focusing on more representative and challenging problems to better understand the influence of the parameters in realistic settings.

The results of the parameter study, illustrated in Fig. A.8, show that the choice of w_u has a notable impact on the convergence behavior of the optimization process. Specifically, higher values of w_u lead to faster convergence toward the global or best-known solution. This finding indicates that prioritizing the predicted objective value (through v_u) over the distance criterion (through v_d) results in more efficient exploitation of the surrogate model. However, w_u is intentionally kept slightly below 1, i.e., $w_u = 0.9$, to preserve a small but nonnegligible contribution from the distance criterion, promoting a minimal level of exploration and avoiding convergence to suboptimal regions. For this reason, values greater than 0.9 were not further considered in this study. Additionally, the option of using a monotonic increase in w_u – starting

from 0.1 and increasing to 0.9 – was also tested, with the intention of encouraging more exploration initially and more exploitation later in the process. However, this approach did not show good convergence.

Furthermore, variations in N_0, N_c, and N_{bc} (see Figs. A.9-A.11) show similar trends to those observed in the GP model case discussed in Section 4.4.3, resulting in comparable convergence performance and CPU time. Therefore, the parameters $N_0 = 10$, $N_c = 500$, and $N_{bc} = 2$ were selected for BO-IPOPT with the linear model in the comparison with the hybrid method including the GP model in Section 4.4.4, providing a reasonable balance between computational effort and convergence rate.

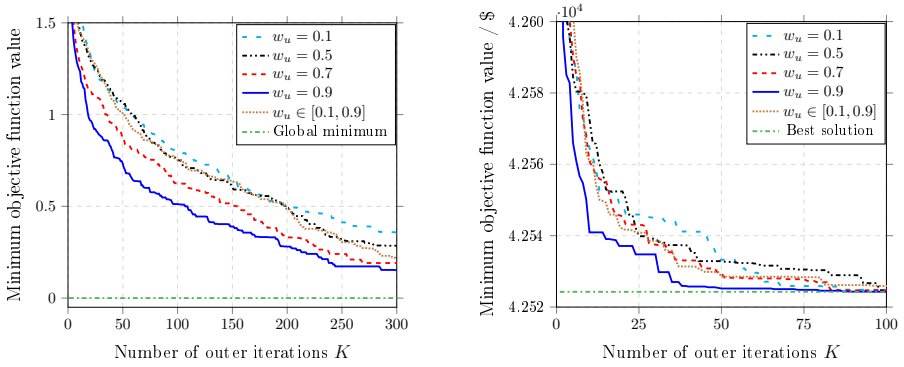

Figure A.8: Optimization results for the constrained Ackley function (4.20)-(4.21) (**left**) and the DED problem (4.22)-(4.25) with 5 units (**right**) using BO-IPOPT with a linear surrogate model: comparison of the averaged minimum objective function value over 100 and 50 trials for different weights w_u as a function of the number of outer iterations K.

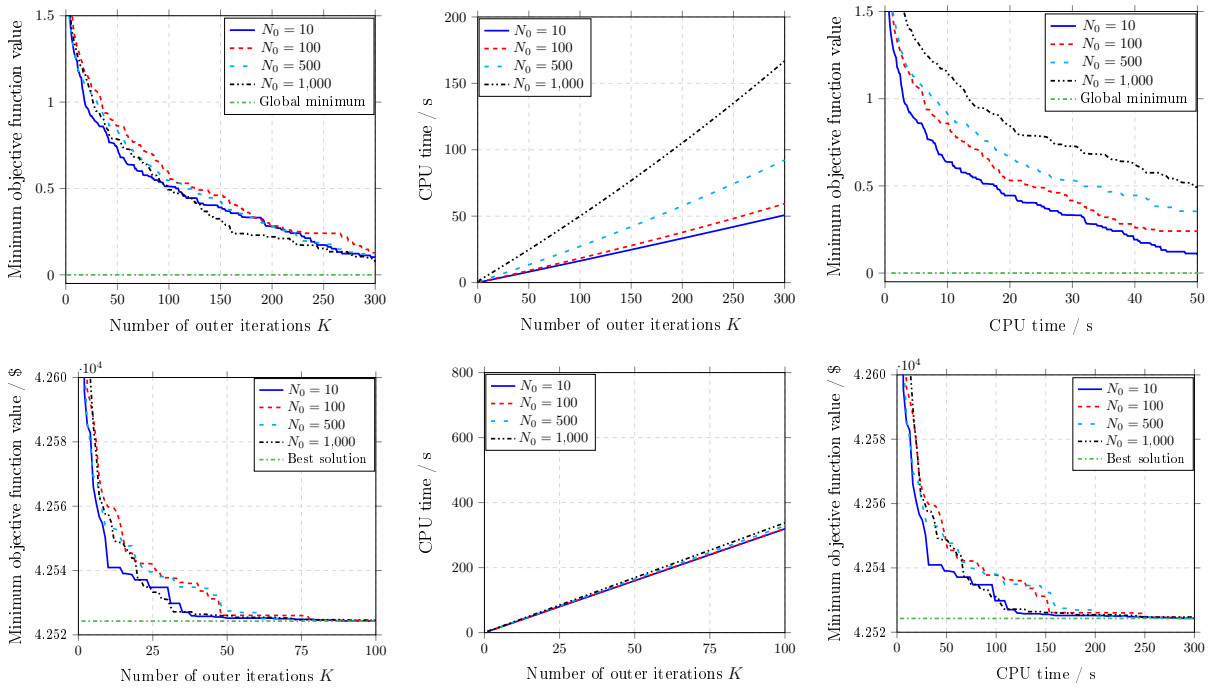

Figure A.9: Optimization results for the the constrained Ackley function (4.20)-(4.21) (**top**) and the DED problem (4.22)-(4.25) with 5 units (**bottom**) using BO-IPOPT with a linear surrogate model: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 and 50 trials for different numbers of initialization points N_0 as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

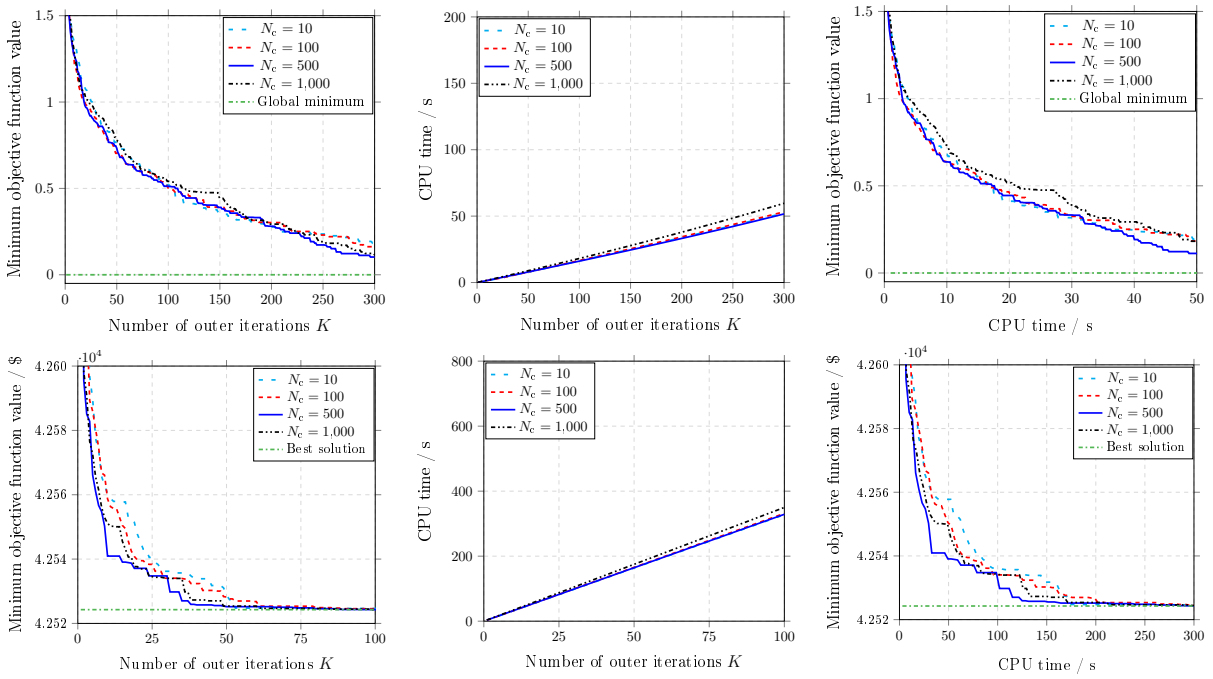

Figure A.10: Optimization results for the the constrained Ackley function (4.20)-(4.21) (**top**) and the DED problem (4.22)-(4.25) with 5 units (**bottom**) using BO-IPOPT with a linear surrogate model: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 and 50 trials for different numbers of candidates N_c as a function of the number of outer K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

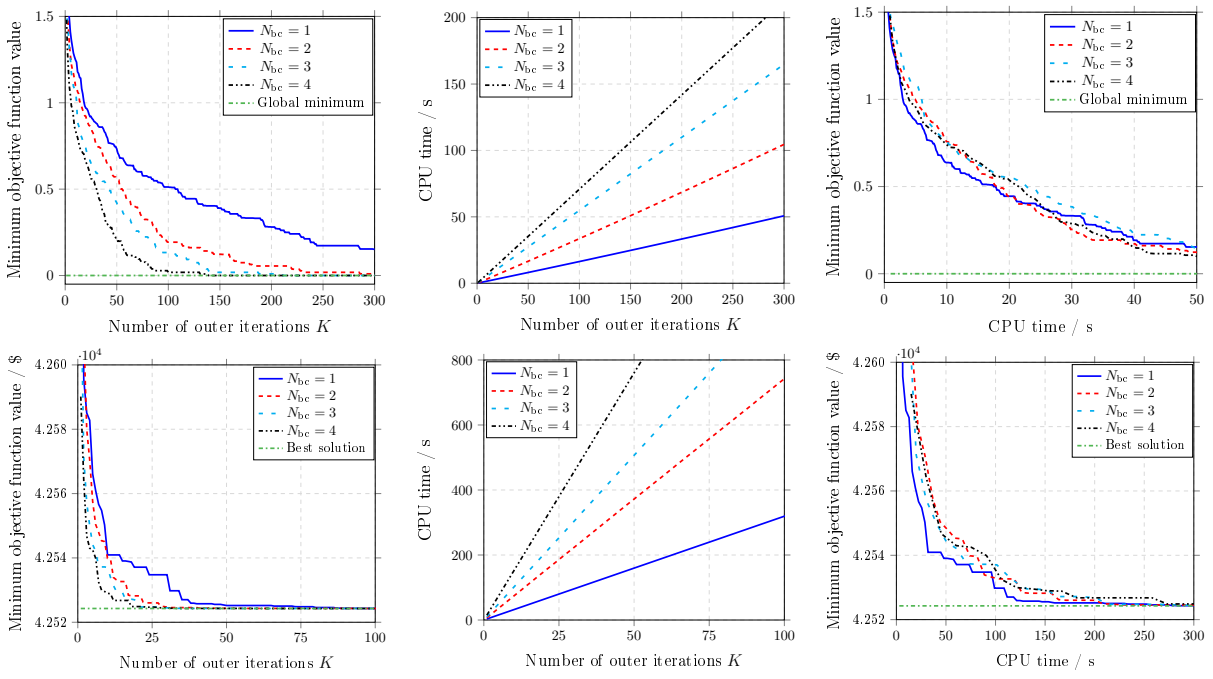

Figure A.11: Optimization results for the constrained Ackley function (4.20)-(4.21) (**top**) and the DED problem (4.22)-(4.25) with 5 units (**bottom**) using BO-IPOPT with a linear surrogate model: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**) and the respective solver performance (**right**) over 100 and 50 trials for different numbers of best candidates N_{bc} as a function of the number of outer iterations K. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

A.3.7. BO-IPOPT: Random vs. Local Initialization Points

A key aspect that has not yet been addressed in this work is how the initialization points used to build the initial GPR are selected. In designing the hybrid BO-IPOPT, random-based initialization points, denoted as N_0, are suggested to enable the GP model to explore the search space more broadly and enhance the optimizer's convergence speed. This approach differs from the BOwLS method, which relies on local search-based initialization for the initial GPR. To highlight the advantages of the strategy considered in BO-IPOPT, the constrained ACKLEY function is used as a case study. As illustrated in Fig. A.12, BO-IPOPT initialized with random points ($N_0 = 10$) achieves better performance (see right-hand figure) compared to the hybrid method based on local minima. As expected, even increasing the number of local minima fails to improve the performance of the optimizer significantly, as the initial GPR remains constrained to a limited region. This, in turn, impacts the selection of the next best candidates, resulting in slower convergence for the optimizer.

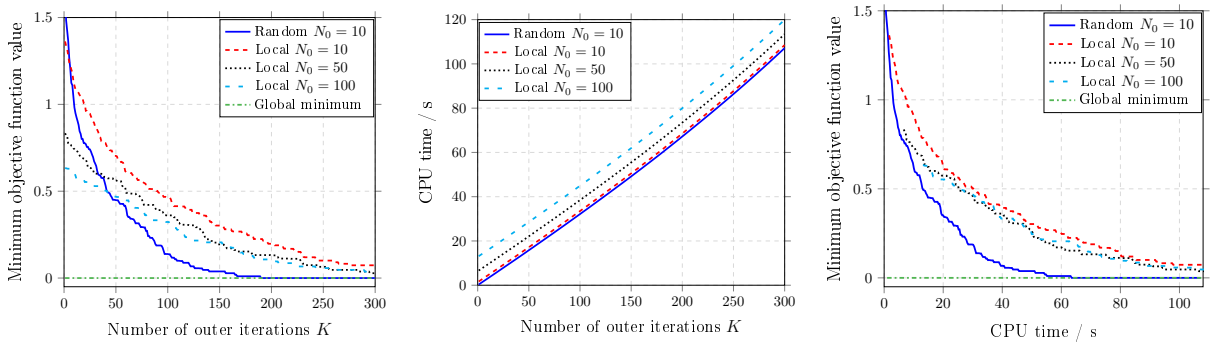

Figure A.12: Optimization results for the constrained ACKLEY function (4.20)-(4.21) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for random and local initialization points N_0 as a function of the number of outer iterations K.

A.3.8. Results of Simple Test Problem

This section presents the results of the simple test problem, highlighting the influence of the hyperparameters in BO-IPOPT on both the convergence rate and computational costs of the optimizer, along with a comparison of BO-IPOPT, BOwLS, and MS-IPOPT.

The variation of the hyperparameters l, α, and ξ in BO-IPOPT has a minimal impact on the solver performance, primarily due to the low dimension and complexity of the problem, as shown in Fig. A.13. While the influence of the parameters N_0, N_c, and N_{bc} is slightly more pronounced (see Figs. A.14-A.16), this test problem is not ideal for evaluating the effect of these hyperparameters, as the global minimum is reached very quickly for all the parameter settings.

As seen in Fig. A.17, the hybrid method BOwLS demonstrates a slightly better performance than the other two methods, particularly in the first outer iterations. This advantage can be attributed to the fact that,

due to the relatively low dimensionality and simplicity of the problem, the initial GP model in BOwLS is initialized with local minima that are already positioned near the global minimum. However, despite this early lead, all three methods converge to the optimal solution within a similar CPU time (approximately 6 seconds). This is because BOwLS requires higher computational costs at each iteration, offsetting its early gains.

It is worth noticing that when the complexity of the optimization problem increase (see Section 4.4.7), the convergence speed of BOwLS is noticeably reduced. In more complex scenarios, the method's initial advantage weakens, and it converges more slowly toward the global optimum compared to its performance in simpler problems.

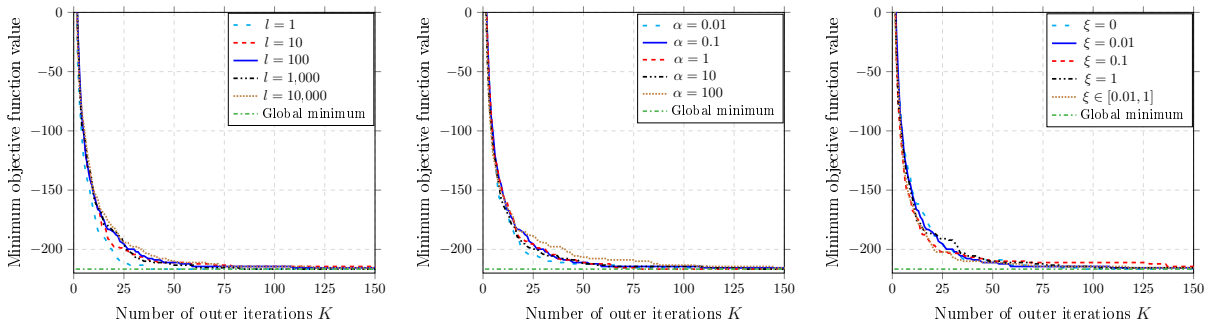

Figure A.13: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value over 100 trials for different length scales l (**left**), regularization terms α (**middle**), and exploration terms ξ (**right**) as a function of the number of outer iterations K.

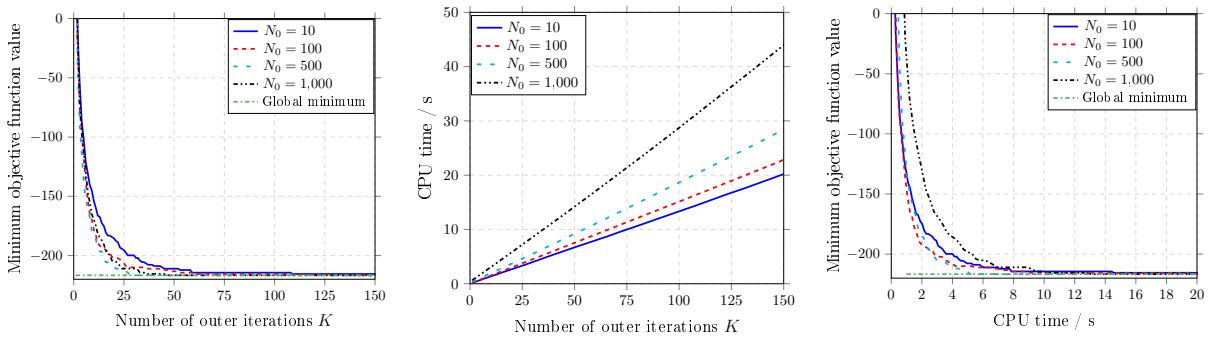

Figure A.14: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for different numbers of initialization points N_0 as a function of the number of outer iterations K.

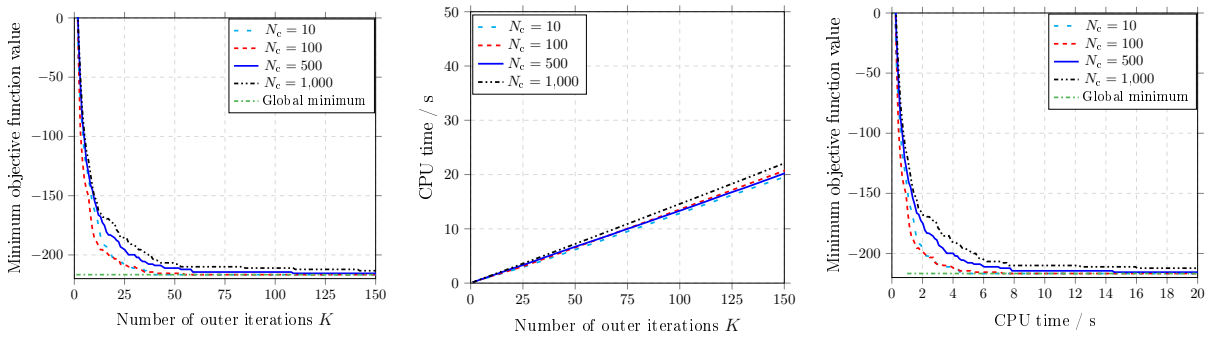

Figure A.15: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for different numbers of candidates N_c as a function of the number of outer iterations K.

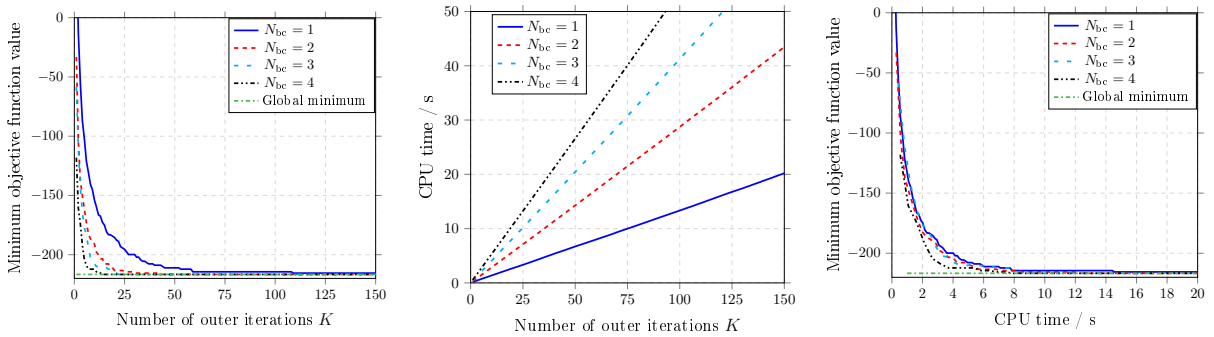

Figure A.16: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for different numbers of best candidates N_{bc} as a function of the number of outer iterations K.

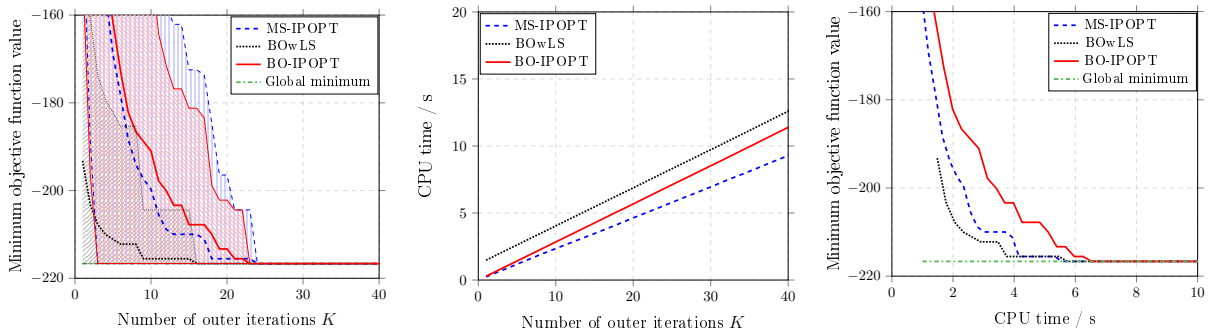

Figure A.17: Optimization results for the simple test problem (4.11)-(4.19): comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves (in the left figure), which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the minimum objective function value to/from the averaged minimum objective function value and describe the worst and best possible convergence rate of each optimizer.

A.3.9. Constraint Handling in BO-IPOPT: Single GP Model vs. Independent GP Models

Traditional BO builds separate GP models for both the objective function and constraints, as outlined in Section 4.2.1. Here, we assess the impact of using a single surrogate model for both the objective and constraints in BO-IPOPT versus employing independent GP models. This comparison is conducted on a simple test problem and the constrained ACKLEY function. When using separate GP models, the EI criterion is adjusted by weighting it according to the probability of satisfying each constraint (Gelbart et al., 2014; Gardner et al., 2014).

Figs. A.18 and A.19 show that while independent GP models improve the convergence rate of BO-IPOPT, they substantially increase the computational overhead, particularly in the simple test case, which contains more constraints than the ACKLEY function. As the problem's dimension and the number of constraints grow (e.g., in DED and RSG problems), the additional computational burden becomes even more pronounced. To mitigate this, we adopt a single GP model for both the objective function and all constraints in BO-IPOPT throughout this study. This choice not only improves the computational

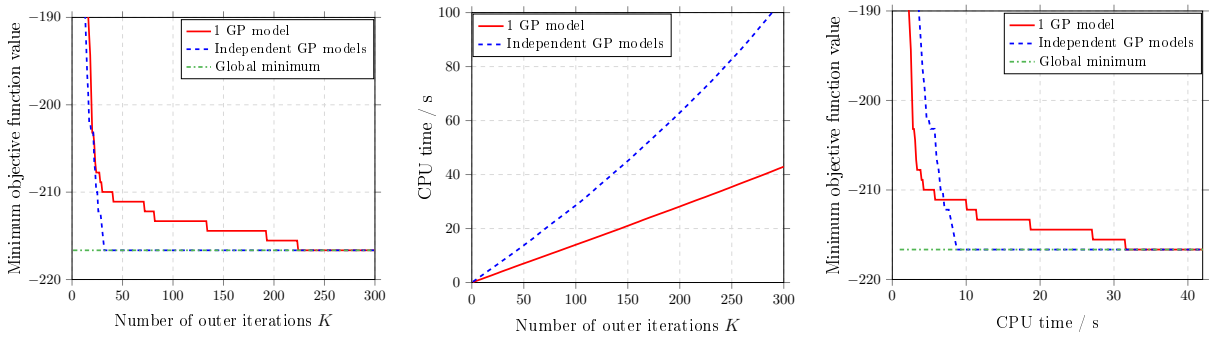

Figure A.18: Optimization results for the simple test problem (4.11)-(4.19) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for single GP model and independent GP models as a function of the number of outer iterations K.

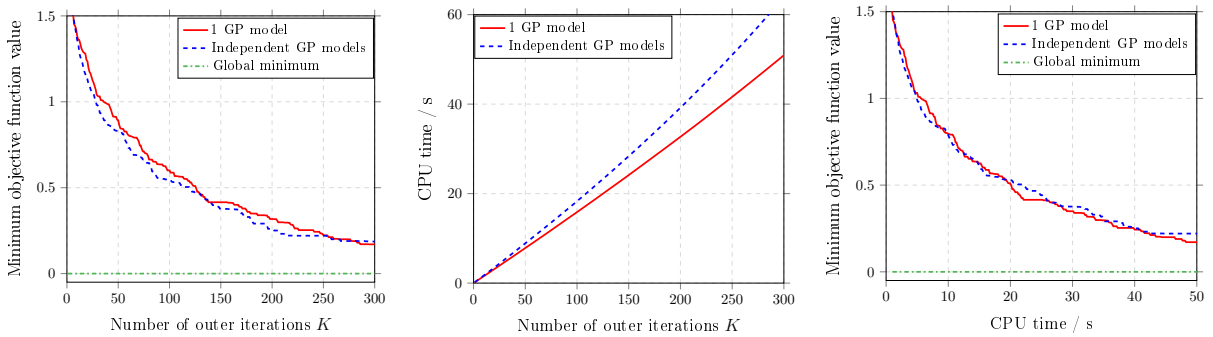

Figure A.19: Optimization results for the constrained ACKLEY function (4.20)-(4.21) using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 100 trials for single GP model and independent GP models as a function of the number of outer iterations K.

efficiency but also aligns with the role of BO in providing global guidance, where the surrogate model does not need to be highly precise but should effectively guide the search toward promising regions.

A.3.10. Reformulation of DED problem

As mentioned in Section 4.4.2, the nonsmooth nature of the DED problem requires reformulation when using gradient-based solvers like IPOPT. This reformulation, which introduces additional slack variables as suggested in Pan et al. (2018), enlarges the original DED problem (with 5 units). In contrast, the BO framework can be directly applied to the original problem without any need for such reformulation. However, since BO does not provide information about the slack variables, these must either be randomly initialized or set to zero when using IPOPT. In this work, these variables are randomly initialized. This raises the question of whether incorporating the same reformulation in the BO part would improve BO-IPOPT's performance by providing IPOPT with better starting values for the slack variables. As shown in Fig. A.20, applying the reformulation to both parts of BO-IPOPT increases the CPU time without significantly improving the convergence, ultimately resulting in poorer performance. Furthermore, increasing the problem size, such as adding more units to the DED problem, exacerbates the issue, as a larger matrix must be inverted at each iteration, further increasing the computational costs. Therefore, in this study, we use the original DED problem formulation for the BO part of BO-IPOPT.

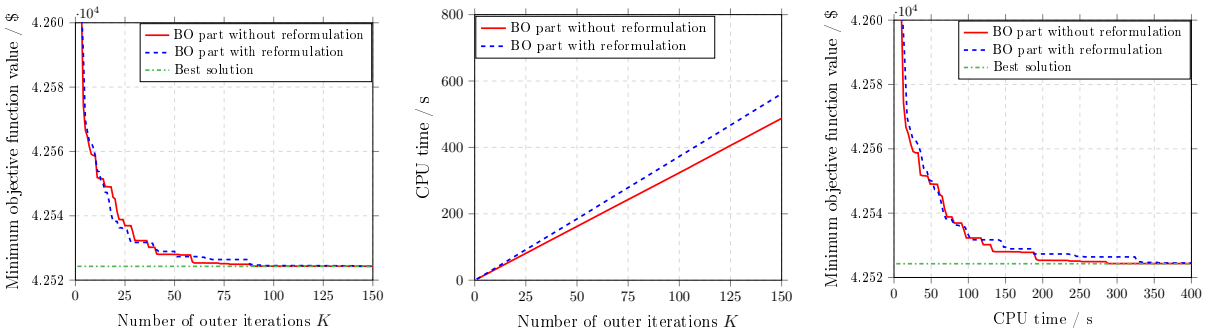

Figure A.20: Optimization results for the DED problem (4.22)-(4.25) with 5 units using BO-IPOPT: comparison of the averaged minimum objective function value (**left**), the averaged CPU time (**middle**), and the respective solver performance (**right**) over 50 trials by reformulating the problem in both parts of BO-IPOPT (i.e., BO and IPOPT) and only in IPOPT as a function of the number of outer iterations K. Currency values are given in USD, represented by the $ sign for brevity.

A.3.11. Results over the total number of function evaluations

This section presents the minimum objective function value over the total number of function evaluations across all benchmark problems, with a comparative analysis of BO-IPOPT, MS-IPOPT, and BOwLS. It should be noted that, while function evaluations in BO involve only evaluating the objective function at a given point, in gradient-based solvers like IPOPT each evaluation includes the objective function, constraints and their Jacobian, and the Hessian of the Lagrangian. For the hybrid methods,

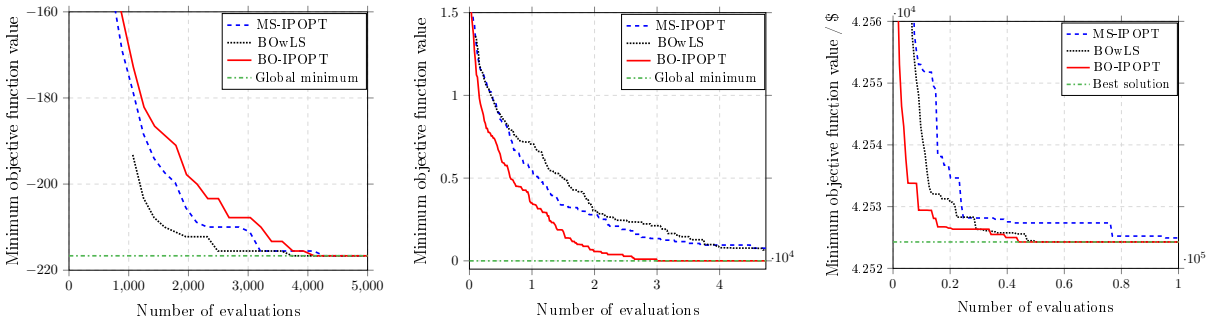

Figure A.21: Optimization results for the simple test problem (4.11)-(4.19) (**left**), the constrained Ackley function (4.20)-(4.21) (**middle**), and the DED problem (4.22)-(4.25) with 5 units (**right**): comparison of the averaged minimum objective function value over 100 and 50 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the total number of function evaluations. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

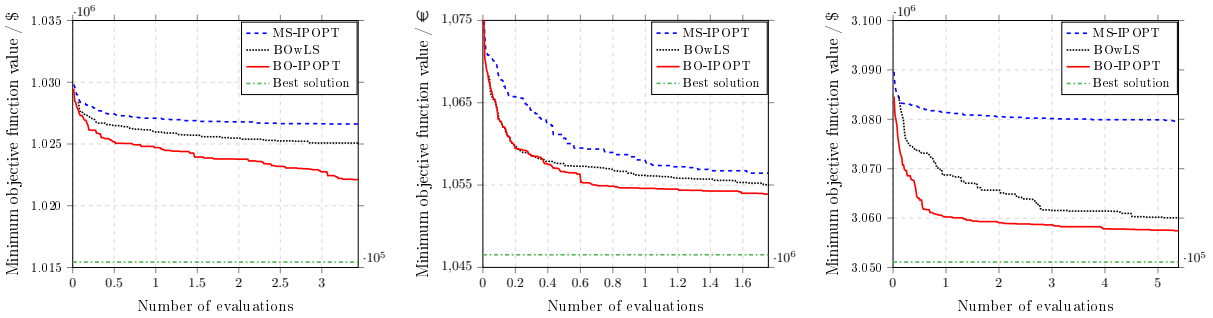

Figure A.22: Optimization results for the DED problem (4.22)-(4.25) with 10 units (**left**), for the RSG problem (4.26)-(4.35) (**middle**), and the DED problem (4.22)-(4.25) with 30 units (**right**): comparison of the averaged minimum objective function value over 50 and 20 trials using MS-IPOPT, BOwLS, and BO-IPOPT as a function of the total number of function evaluations. Currency values for the DED problem are given in USD, represented by the $ sign for brevity.

the total number of function evaluations combines those from both components. Figs. A.21 and A.22 show how the three optimization methods perform in terms of the number of function evaluations required to approximate the global minimum or best-known solution for the problems examined in this study. As clearly seen in these figures, BO-IPOPT demonstrates a clear advantage by achieving convergence toward the global optimum or the best available solution with significantly fewer function evaluations than MS-IPOPT and BOwLS.

A.3.12. BO-IPOPT: Convergence Behavior in Selected Candidates

Another aspect investigated in this study is the likelihood of the two selected candidates N_{bc} at each iteration in BO-IPOPT converging to the same local optimum. To assess this, we calculated the Euclidean distance between the two candidates after local optimization, and the results are presented in Figs. A.23-A.24. To provide further insight, the figures include confidence intervals, computed by adding and subtracting the standard deviation to/from the mean distance. It is worth noticing that the distances

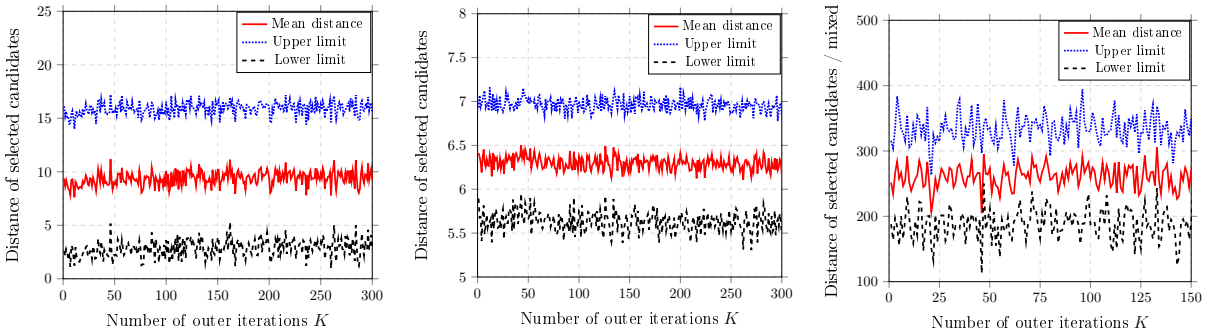

Figure A.23: Optimization results for the simple test problem (4.11)-(4.19) (**left**), the constrained ACKLEY function (4.20)-(4.21) (**middle**), and the DED problem (4.22)-(4.25) with 5 units (**right**): comparison of the averaged distance of the selected candidates ($N_{\text{bc}} = 2$) after applying IPOPT over 100 and 50 trials using BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves, which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the distance to/from the averaged distance and describe the worst and best possible case.

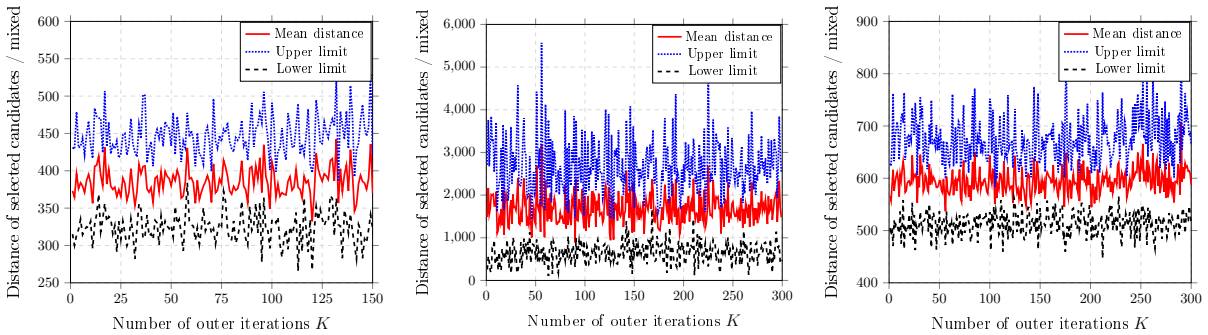

Figure A.24: Optimization results for the DED problem (4.22)-(4.25) with 10 units (**left**), for the RSG problem (4.26)-(4.35) (**middle**), and the DED problem (4.22)-(4.25) with 30 units (**right**): comparison of the averaged distance of the selected candidates ($N_{\text{bc}} = 2$) after applying IPOPT over 100 and 50 trials using BO-IPOPT as a function of the number of outer iterations K. The upper and lower curves, which form a confidence interval for each optimizer, result from adding/subtracting the standard deviation of the distance to/from the averaged distance and describe the worst and best possible case.

determined for the RSG and DED problems involve mixed units, since the optimization variables have different physical units.

The results show that, across all benchmark problems, there were no instances where both candidates converged to the same local optimum, because their distance was always greater than zero – even in the worst-case scenario represented by the lower bound of the interval. These findings suggest sufficient diversity in the selection of candidates, even when only two are used. However, in future work where more than two candidates may be selected at each outer iteration, methods such as the q-expected improvement (q-EI) acquisition function could be employed to enhance both the diversity and quality of the selected points.

A.3.13. Comparison of BO-IPOPT, BO-XPRESS, and BO-CONOPT

One key advantage of the proposed hybrid method lies in its flexibility regarding the choice of the local solver. This flexibility allows the local solver to be easily replaced by an alternative one without requiring modifications to the overall framework. In this study, IPOPT was selected due to its strong efficiency and open-source accessibility. However, this raises the question of how IPOPT's performance compares to that of commercial solvers. To investigate this, we conducted tests incorporating two popular commercial solvers – XPRESS (Fair Isaac Corporation, 2025) and CONOPT4 (Drud, 1994) – within our hybrid method.

The optimization outcomes for the 10-unit DED problem and the RSG problem, obtained using BO-IPOPT, BO-XPRESS, and BO-CONOPT, are presented in Figs. A.25 and A.26, respectively. The results highlight distinct solver behaviors. In the RSG problem, BO-XPRESS and BO-CONOPT demonstrate significantly faster convergence toward the best-known solution compared to BO-IPOPT. However, in the DED problem, BO-IPOPT outperforms both commercial solvers, exhibiting a noticeably

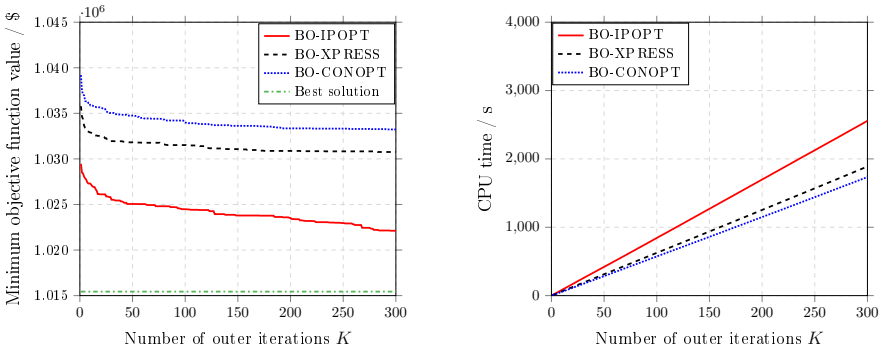

Figure A.25: Optimization results for the DED problem (4.22)-(4.25) with 10 units: comparison of the averaged minimum objective function value (**left**) and averaged CPU time (**right**) over 50 trials using BO-IPOPT, BO-XPRESS, and BO-CONOPT as a function of the number of outer iterations K. IPOPT is open-source, while XPRESS and CONOPT are commercial solvers. Currency values are given in USD, represented by the $ sign for brevity.

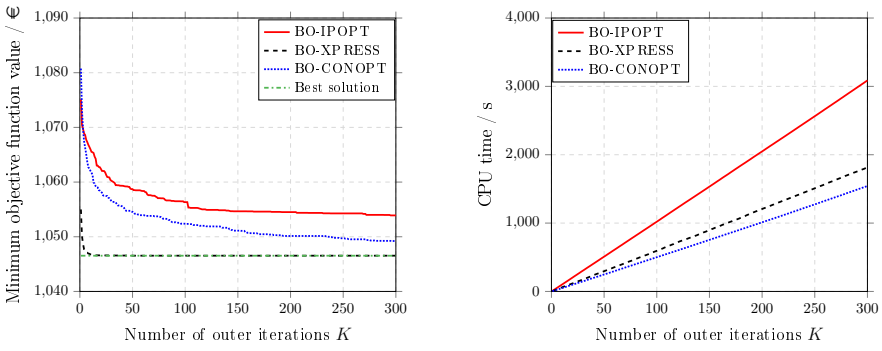

Figure A.26: Optimization results for the RSG problem (4.26)-(4.35): comparison of the averaged minimum objective function value (**left**) and averaged CPU time (**right**) over 50 trials using BO-IPOPT, BO-XPRESS, and BO-CONOPT as a function of the number of outer iterations K. IPOPT is open-source, while XPRESS and CONOPT are commercial solvers.

superior convergence rate. These differing solver performances emphasize the variability in how different solvers handle specific problem structures, which could be an important topic for future research. A deeper analysis of solver selection and problem-dependent solver effectiveness could provide additional insights and further enhance the performance of hybrid optimization approaches.